



# Gowin 原语指南

SUG283-1.5, 2018-11-15

## **版权所有©2018广东高云半导体科技股份有限公司**

未经本公司书面许可，任何单位和个人不得擅自摘抄、复制、翻译本档内容的部分或全部，并不得以任何形式传播。

### **免责声明**

本档并未授予任何知识产权的许可，并未以明示或暗示，或以禁止发言或其它方式授予任何知识产权许可。除高云半导体在其产品的销售条款和条件中声明的责任之外，高云半导体概不承担任何法律或非法律责任。高云半导体对高云半导体产品的销售和 / 或使用不作任何明示或暗示的担保，包括对产品的特定用途适用性、适销性或对任何专利权、版权或其它知识产权的侵权责任等，均不作担保。高云半导体对档中包含的文字、图片及其它内容的准确性和完整性不承担任何法律或非法律责任，高云半导体保留修改档中任何内容的权利，恕不另行通知。高云半导体不承诺对这些档进行适时的更新。

## 版本信息

日期	版本	说明
2017/04/20	1.0	初始版本
2017/09/19	1.1	<ul style="list-style-type: none"><li>● 增加支持器件系列 GW1NR-4、GW1N-6、GW1N-9、GW1NR-9;</li><li>● 增加 ELVDS_Iobuf、TLVDS_Iobuf、BUFG、BUFS、OSC、IEM;</li><li>● 更新 DSP 原语;</li><li>● 更新 ODDR/ODDRc、IDDR_MEM、IDES4_MEM、IDES8_MEM、RAM16S1、RAM16S2、RAM16S4、RAM16SDP1、RAM16SDP2、RAM16SDP4、ROM16 部分 port 名称;</li><li>● 更新 OSC、PLL、DLLDLY 部分 Attribute;</li><li>● 更新部分原语例化;</li><li>● 增加 MIPI_Iobuf_HS、MIPI_Iobuf_LP、MIPI_OBUF、IDES16、OSER16;</li><li>● 更新 CLKDIV 部分 Attribute。</li></ul>
2018/04/12	1.2	<ul style="list-style-type: none"><li>● 增加 vhdI 原语例化</li></ul>
2018/08/08	1.3	<ul style="list-style-type: none"><li>● 增加支持器件系列 GW1N-2B、GW1N-4B、GW1NR-4B、GW1N-6ES、GW1N-9ES、GW1NR-9ES、GW1NS-2、GW1NS-2C;</li><li>● 增加 I3C_Iobuf、DHCEN;</li><li>● 增加 User Flash;</li><li>● 增加 EMPU;</li><li>● 更新原语名称。</li></ul>
2018/10/26	1.4	<ul style="list-style-type: none"><li>● 增加支持器件系列 GW1NZ-1、GW1NSR-2C;</li><li>● 增加 OSCZ、FLASH96KZ。</li></ul>
2018/11/15	1.5	<ul style="list-style-type: none"><li>● 增加支持器件系列 GW1NSR-2;</li><li>● 删除器件 GW1N-6ES、GW1N-9ES、GW1NR-9ES。</li></ul>

# 目录

目录 .....	ii
图目录 .....	vi
表目录 .....	x
<b>1 IOB .....</b>	<b>1</b>
1.1 Buffer/LVDS .....	1
1.1.1 IBUF .....	1
1.1.2 OBUF .....	2
1.1.3 TBUF .....	3
1.1.4 IOBUF .....	4
1.1.5 LVDS input buffer .....	5
1.1.6 LVDS ouput buffer .....	6
1.1.7 LVDS tristate buffer .....	8
1.1.8 LVDS inout buffer .....	10
1.1.9 MIPI_IBUF_HS .....	11
1.1.10 MIPI_IBUF_LP .....	12
1.1.11 MIPI_OBUF .....	13
1.1.12 I3C_IOBUF .....	14
1.2 IOLOGIC .....	15
1.2.1 IDDR .....	15
1.2.2 ODDR .....	16
1.2.3 IDDRC .....	18
1.2.4 ODDRC .....	19
1.2.5 IDES4 .....	21
1.2.6 IDES8 .....	22
1.2.7 IDES10 .....	24
1.2.8 IVIDEO .....	26
1.2.9 IDES16 .....	29
1.2.10 OSER4 .....	31
1.2.11 OSER8 .....	33
1.2.12 OSER10 .....	36

---

1.2.13 OVIDEO .....	38
1.2.14 OSER16.....	39
1.2.15 IDDR_MEM.....	42
1.2.16 ODDR_MEM.....	43
1.2.17 IDES4_MEM.....	45
1.2.18 OSER4_MEM.....	48
1.2.19 IDES8_MEM.....	50
1.2.20 OSER8_MEM.....	52
1.2.21 IODELAY .....	55
1.2.22 IEM .....	57
<b>2 CLU .....</b>	<b>60</b>
2.1 LUT .....	60
2.1.1 LUT1 .....	61
2.1.2 LUT2 .....	62
2.1.3 LUT3 .....	63
2.1.4 LUT4 .....	65
2.1.5 Wide LUT.....	67
2.2 MUX.....	69
2.2.1 MUX2.....	69
2.2.2 MUX4.....	70
2.2.3 Wide MUX.....	72
2.3 ALU .....	74
2.4 FF.....	77
2.4.1 DFF .....	77
2.4.2 DFFE .....	79
2.4.3 DFFS .....	80
2.4.4 DFFSE .....	81
2.4.5 DFFR .....	83
2.4.6 DFFRE.....	84
2.4.7 DFFP .....	85
2.4.8 DFFPE .....	86
2.4.9 DFFC .....	88
2.4.10 DFFCE.....	89
2.4.11 DFFN.....	91
2.4.12 DFFNE.....	92
2.4.13 DFFNS.....	93
2.4.14 DFFNSE.....	94
2.4.15 DFFNR.....	96
2.4.16 DFFNRE.....	97

2.4.17 DFFNP .....	98
2.4.18 DFFNPE .....	99
2.4.19 DFFNC.....	101
2.4.20 DFFNCE .....	102
2.5 LATCH .....	104
2.5.1 DL .....	104
2.5.2 DLE .....	105
2.5.3 DLC.....	107
2.5.4 DLCE .....	108
2.5.5 DLP .....	110
2.5.6 DLPE .....	111
2.5.7 DLN.....	112
2.5.8 DLNE .....	113
2.5.9 DLNC .....	115
2.5.10 DLNCE.....	116
2.5.11 DLNP.....	118
2.5.12 DLNPE.....	119
<b>3 CFU .....</b>	<b>121</b>
3.1 SSRAM .....	121
3.1.1 RAM16S1 .....	121
3.1.2 RAM16S2 .....	123
3.1.3 RAM16S4 .....	124
3.1.4 RAM16SDP1 .....	126
3.1.5 RAM16SDP2 .....	128
3.1.6 RAM16SDP4 .....	129
3.1.7 ROM16 .....	131
<b>4 Block SRAM .....</b>	<b>133</b>
4.1 SP/SPX9.....	133
4.2 SDP/SDPX9.....	138
4.3 DP/DPX9 .....	144
4.4 ROM/ROMX9.....	151
<b>5 DSP .....</b>	<b>156</b>
5.1 Pre-adder .....	156
5.1.1 PADD18 .....	156
5.1.2 PADD9 .....	159
5.2 Multiplier .....	161
5.2.1 MULT18X18.....	162
5.2.2 MULT9X9.....	165
5.2.3 MULT36X36.....	168

---

5.3 ALU54D .....	171
5.4 MULTALU.....	174
5.4.1 MULTALU36X18 .....	174
5.4.2 MULTALU18X18 .....	179
5.5 MULTADDALU .....	184
5.5.1 MULTADDALU18X18 .....	184
<b>6 Clock.....</b>	<b>190</b>
6.1 PLL.....	190
6.2 DLL/DLLDLY .....	196
6.2.1 DLL .....	196
6.2.2 DLLDLY.....	198
6.3 CLKDIV .....	200
6.4 DQCE .....	202
6.5 DCS .....	203
6.6 DQS .....	205
6.7 OSC .....	208
6.8 OSCZ.....	209
6.9 OSCF .....	210
6.10 DHCEN .....	212
6.11 BUFG .....	213
6.12 BUFS .....	213
<b>7 User Flash .....</b>	<b>215</b>
7.1 FLASH96K.....	215
7.2 FLASH96KZ.....	217
7.3 FLASH128K.....	219
7.4 FLASH256K.....	221
7.5 FLASH608K.....	223
<b>8 EMPU .....</b>	<b>226</b>
8.1 MCU.....	226
8.2 USB20_PHY .....	237
8.3 ADC.....	245

# 图目录

图 1-1 IBUF 结构框图.....	1
图 1-2 OBUF 结构框图.....	2
图 1-3 TBUF 结构框图.....	3
图 1-4 IOBUF 结构框图.....	4
图 1-5 TLVDS_IBUF/ELVDS_IBUF 结构框图.....	5
图 1-6 TLVDS_OBUF/ELVDS_OBUF 结构框图.....	7
图 1-7 TLVDS_TBUF/ELVDS_TBUF 结构框图.....	8
图 1-8 TLVDS_IOBUF/ELVDS_IOBUF 结构框图.....	10
图 1-9 MIPI_IBUF_HS 结构框图.....	11
图 1-10 MIPI_IBUF_LP 结构框图.....	12
图 1-11 MIPI_OBUF 结构框图.....	13
图 1-12 I3C_IOBUF 结构框图.....	14
图 1-13 IDDR 结构框图.....	15
图 1-14 ODDR 结构框图.....	16
图 1-15 IDDR_C 结构框图.....	18
图 1-16 ODDR_C 结构框图.....	19
图 1-17 IDES4 结构框图.....	21
图 1-18 IDES8 结构框图.....	23
图 1-19 IDES10 结构框图.....	25
图 1-20 IVIDEO 结构框图.....	27
图 1-21 IDES16 结构框图.....	29
图 1-22 OSER4 结构框图.....	31
图 1-23 OSER8 结构框图.....	33
图 1-24 OSER10 结构框图.....	36
图 1-25 OVIDEO 结构框图.....	38
图 1-26 OSER16 结构框图.....	40
图 1-27 IDDR_MEM 结构框图.....	42
图 1-28 ODDR_MEM 结构框图.....	44
图 1-29 IDES4_MEM 结构框图.....	46



图 1-30 OSER4_MEM 结构框图 .....	48
图 1-31 IDES8_MEM 结构框图 .....	50
图 1-32 OSER8_MEM 结构框图 .....	53
图 1-33 IODELAY 结构框图 .....	55
图 1-34 IEM 结构框图 .....	57
图 2-1 CLU 结构示意图 .....	60
图 2-2 LUT1 结构框图 .....	61
图 2-3 LUT2 结构框图 .....	62
图 2-4 LUT3 结构框图 .....	63
图 2-5 LUT4 结构框图 .....	65
图 2-6 MUX2_LUT5 结构框图 .....	67
图 2-7 MUX2 结构框图 .....	69
图 2-8 MUX4 结构框图 .....	71
图 2-9 MUX2_MUX8 结构框图 .....	72
图 2-10 ALU 结构框图 .....	75
图 2-11 DFF 结构框图 .....	78
图 2-12 DFFE 结构框图 .....	79
图 2-13 DFFS 结构框图 .....	80
图 2-14 DFFSE 结构框图 .....	81
图 2-15 DFFR 结构框图 .....	83
图 2-16 DFFRE 结构框图 .....	84
图 2-17 DFFP 结构框图 .....	85
图 2-18 DFFPE 结构框图 .....	87
图 2-19 DFFC 结构框图 .....	88
图 2-20 DFFCE 结构框图 .....	89
图 2-21 DFFN 结构框图 .....	91
图 2-22 DFFNE 结构框图 .....	92
图 2-23 DFFNS 结构框图 .....	93
图 2-24 DFFNSE 结构框图 .....	94
图 2-25 DFFNR 结构框图 .....	96
图 2-26 DFFNRE 结构框图 .....	97
图 2-27 DFFNP 结构框图 .....	98
图 2-28 DFFNPE 结构框图 .....	100
图 2-29 DFFNC 结构框图 .....	101
图 2-30 DFFNCE 结构框图 .....	102
图 2-31 DL 结构框图 .....	104
图 2-32 DLE 结构框图 .....	106

图 2-33 DLC 结构框图.....	107
图 2-34 DLCE 结构框图 .....	108
图 2-35 DLP 结构框图.....	110
图 2-36 DLPE 结构框图.....	111
图 2-37 DLN 结构框图.....	112
图 2-38 DLNE 结构框图 .....	114
图 2-39 DLNC 结构框图 .....	115
图 2-40 DLNCE 结构框图.....	116
图 2-41 DLNP 结构框图 .....	118
图 2-42 DLNPE 结构框图 .....	119
图 3-1 RAM16S1 结构框图.....	122
图 3-2 RAM16S2 结构框图.....	123
图 3-3 RAM16S4 结构框图.....	125
图 3-4 RAMSDP1 结构框图.....	126
图 3-5 RAM16SDP2 结构框图.....	128
图 3-6 RAMSDP4 结构框图.....	130
图 3-7 ROM16 结构框图.....	131
图 4-1 SP/SPX9 结构框图 .....	134
图 4-2 SDP/SDPX9 结构框图.....	139
图 4-3 DP/DPX9 结构框图.....	144
图 4-4 ROM/ROMX9 结构框图.....	151
图 5-1 结构框图 .....	156
图 5-2 PADD9 结构框图 .....	159
图 5-3 MULT18X18 结构框图 .....	162
图 5-4 MULT9X9 结构框图 .....	165
图 5-5 MULT36X36 结构框图 .....	168
图 5-6 ALU54D 结构框图.....	171
图 5-7 MULTALU36X18 结构框图 .....	175
图 5-8 MULTALU18X18 结构框图 .....	179
图 5-9 MULTADDALU18X18 结构框图.....	184
图 6-1 PLL 结构框图.....	191
图 6-2 DLL 结构框图 .....	196
图 6-3 DLLDLY 结构框图.....	199
图 6-4 CLKDIV 结构框图.....	201
图 6-5 DQCE 结构框图.....	202
图 6-6 DCS 结构框图 .....	203
图 6-7 DQS 结构框图 .....	205

---

图 6-8 OSC 结构框图 .....	208
图 6-9 OSCZ 结构框图 .....	210
图 6-10 OSCF 结构框图 .....	211
图 6-11 DHCEN 结构框图.....	212
图 6-12 BUFG 结构框图 .....	213
图 6-13 BUFS 结构框图 .....	214
图 7-1 FLASH96K 结构框图 .....	215
图 7-2 FLASH96KZ 结构框图.....	218
图 7-3 FLASH128K 结构框图 .....	219
图 7-4 FLASH256K 结构框图 .....	222
图 7-5 FLASH608K 结构框图 .....	223
图 8-1 MCU 结构框图 .....	227
图 8-2 USB20_PHY 结构框图 .....	237
图 8-3 ADC 结构框图.....	245

# 表目录

表 1-1 Port 介绍.....	1
表 1-2 Port 介绍.....	2
表 1-3 Port 介绍.....	3
表 1-4 Port 介绍.....	4
表 1-5 Port 介绍.....	5
表 1-6 Port 介绍.....	7
表 1-7 Port 介绍.....	8
表 1-8 Port 介绍.....	10
表 1-9 Port 介绍.....	11
表 1-10 Port 介绍.....	12
表 1-11 Port 介绍.....	13
表 1-12 Port 介绍.....	14
表 1-13 Port 介绍.....	15
表 1-14 Port 介绍.....	17
表 1-15 Port 介绍.....	18
表 1-16 Port 介绍.....	20
表 1-17 Port 介绍.....	21
表 1-18 Port 介绍.....	23
表 1-19 Port 介绍.....	25
表 1-20 Port 介绍.....	27
表 1-21 Port 介绍.....	29
表 1-22 Port 介绍.....	32
表 1-23 Port 介绍.....	34
表 1-24 Port 介绍.....	36
表 1-25 Port 介绍.....	38
表 1-26 Port 介绍.....	40
表 1-27 Port 介绍.....	42
表 1-28 Port 介绍.....	44
表 1-29 Port 介绍.....	46

表 1-30 Port 介绍.....	48
表 1-31 Port 介绍.....	51
表 1-32 Port 介绍.....	53
表 1-33 Port 介绍.....	56
表 1-34 Attribute 介绍.....	56
表 1-35 Port 介绍.....	57
表 1-36 Attribute 介绍.....	58
表 2-1 Port 介绍.....	61
表 2-2 Attribute 介绍.....	61
表 2-3 真值表.....	61
表 2-4 Port 介绍.....	62
表 2-5 Attribute 介绍.....	62
表 2-6 真值表.....	63
表 2-7 Port 介绍.....	64
表 2-8 Attribute 介绍.....	64
表 2-9 真值表.....	64
表 2-10 Port 介绍.....	65
表 2-11 Attribute 介绍.....	65
表 2-12 真值表.....	66
表 2-13 Port 介绍.....	67
表 2-14 真值表.....	67
表 2-15 Port 介绍.....	70
表 2-16 真值表.....	70
表 2-17 Port 介绍.....	71
表 2-18 真值表.....	71
表 2-19 Port 介绍.....	73
表 2-20 真值表.....	73
表 2-21 ALU 功能.....	75
表 2-22 Port 介绍.....	75
表 2-23 Attribute 介绍.....	76
表 2-24 与 FF 相关的原语.....	77
表 2-25 Port 介绍.....	78
表 2-26 Attribute 介绍.....	78
表 2-27 Port 介绍.....	79
表 2-28 Attribute 介绍.....	79
表 2-29 Port 介绍.....	80
表 2-30 Attribute 介绍.....	81

表 2-31 Port 介绍.....	82
表 2-32 Attribute 介绍.....	82
表 2-33 Port 介绍.....	83
表 2-34 Attribute 介绍.....	83
表 2-35 Port 介绍.....	84
表 2-36 Attribute 介绍.....	84
表 2-37 Port 介绍.....	86
表 2-38 Attribute 介绍.....	86
表 2-39 Port 介绍.....	87
表 2-40 Attribute 介绍.....	87
表 2-41 Port 介绍.....	88
表 2-42 Attribute 介绍.....	88
表 2-43 Port 介绍.....	90
表 2-44 Attribute 介绍.....	90
表 2-45 Port 介绍.....	91
表 2-46 Attribute 介绍.....	91
表 2-47 Port 介绍.....	92
表 2-48 Attribute 介绍.....	92
表 2-49 Port 介绍.....	93
表 2-50 Attribute 介绍.....	93
表 2-51 Port 介绍.....	95
表 2-52 Attribute 介绍.....	95
表 2-53 Port 介绍.....	96
表 2-54 Attribute 介绍.....	96
表 2-55 Port 介绍.....	97
表 2-56 Attribute 介绍.....	97
表 2-57 Port 介绍.....	99
表 2-58 Attribute 介绍.....	99
表 2-59 Port 介绍.....	100
表 2-60 Attribute 介绍.....	100
表 2-61 Port 介绍.....	101
表 2-62 Attribute 介绍.....	101
表 2-63 Port 介绍.....	103
表 2-64 Attribute 介绍.....	103
表 2-65 与 LATCH 相关的原语.....	104
表 2-66 Port 介绍.....	105
表 2-67 Attribute 介绍.....	105

表 2-68 Port 介绍.....	106
表 2-69 Attribute 介绍.....	106
表 2-70 Port 介绍.....	107
表 2-71 Attribute 介绍.....	107
表 2-72 Port 介绍.....	109
表 2-73 Attribute 介绍.....	109
表 2-74 Port 介绍.....	110
表 2-75 Attribute 介绍.....	110
表 2-76 Port 介绍.....	111
表 2-77 Attribute 介绍.....	111
表 2-78 Port 介绍.....	113
表 2-79 Attribute 介绍.....	113
表 2-80 Port 介绍.....	114
表 2-81 Attribute 介绍.....	114
表 2-82 Port 介绍.....	115
表 2-83 Attribute 介绍.....	115
表 2-84 Port 介绍.....	117
表 2-85 Attribute 介绍.....	117
表 2-86 Port 介绍.....	118
表 2-87 Attribute 介绍.....	118
表 2-88 Port 介绍.....	119
表 2-89 Attribute 介绍.....	119
表 3-1 SSRAM.....	121
表 3-2 Port 介绍.....	122
表 3-3 Attribute 介绍.....	122
表 3-4 Port 介绍.....	123
表 3-5 Attribute 介绍.....	124
表 3-6 Port 介绍.....	125
表 3-7 Attribute 介绍.....	125
表 3-8 Port 介绍.....	127
表 3-9 Attribute 介绍.....	127
表 3-10 Port 介绍.....	128
表 3-11 Attribute 介绍.....	128
表 3-12 Port 介绍.....	130
表 3-13 Attribute 介绍.....	130
表 3-14 Port 介绍.....	132
表 3-15 Attribute 介绍.....	132

表 4-1 Port 介绍.....	134
表 4-2 Attribute 介绍.....	135
表 4-3 配置关系.....	135
表 4-4 Port 介绍.....	139
表 4-5 Attribute 介绍.....	140
表 4-6 配置关系.....	140
表 4-7 Port 介绍.....	145
表 4-8 Attribute 介绍.....	146
表 4-9 配置关系.....	146
表 4-10 Port 介绍.....	151
表 4-11 Attribute 介绍.....	152
表 4-12 配置关系.....	152
表 5-1 Port 介绍.....	157
表 5-2 Attribute 介绍.....	157
表 5-3 Port 介绍.....	159
表 5-4 Attribute 介绍.....	160
表 5-5 Port 介绍.....	162
表 5-6 Attribute 介绍.....	163
表 5-7 Port 介绍.....	165
表 5-8 Attribute 介绍.....	166
表 5-9 Port 介绍.....	169
表 5-10 Attribute 介绍.....	169
表 5-11 Port 介绍.....	171
表 5-12 Attribute 介绍.....	172
表 5-13 Port 介绍.....	175
表 5-14 Attribute 介绍.....	176
表 5-15 Port 介绍.....	180
表 5-16 Attribute 介绍.....	181
表 5-17 Port 介绍.....	185
表 5-18 Attribute 介绍.....	186
表 6-1 PLL 性能.....	190
表 6-2 Port 介绍.....	191
表 6-3 Attribute 介绍.....	192
表 6-4 Port 介绍.....	197
表 6-5 Attribute 介绍.....	197
表 6-6 Port 介绍.....	199
表 6-7 Attribute 介绍.....	199



---

表 6-8 Port 介绍.....	201
表 6-9 Attribute 介绍.....	201
表 6-10 Port 介绍.....	202
表 6-11 Port 介绍.....	204
表 6-12 Attribute 介绍.....	204
表 6-13 Port 介绍.....	205
表 6-14 Attribute 介绍.....	206
表 6-15 Port 介绍.....	209
表 6-16 Attribute 介绍.....	209
表 6-17 Port 介绍.....	210
表 6-18 Attribute 介绍.....	210
表 6-19 Port 介绍.....	211
表 6-20 Attribute 介绍.....	211
表 6-21 Port 介绍.....	212
表 6-22 Port 介绍.....	213
表 6-23 Port 介绍.....	214
表 7-1 Port 介绍.....	216
表 7-2 Port 介绍.....	218
表 7-3 Port 介绍.....	220
表 7-4 Port 介绍.....	222
表 7-5 Port 介绍.....	224
表 8-1 Port 介绍.....	228
表 8-2 Port 介绍.....	238
表 8-3 Attribute 介绍.....	240
表 8-4 Port 介绍.....	245
表 1-1 Port 介绍.....	1
表 1-2 Port 介绍.....	2
表 1-3 Port 介绍.....	3
表 1-4 Port 介绍.....	4
表 1-5 Port 介绍.....	5
表 1-6 Port 介绍.....	7
表 1-7 Port 介绍.....	8
表 1-8 Port 介绍.....	10
表 1-9 Port 介绍.....	11
表 1-10 Port 介绍.....	12
表 1-11 Port 介绍.....	13

表 1-12 Port 介绍.....	14
表 1-13 Port 介绍.....	15
表 1-14 Port 介绍.....	17
表 1-15 Port 介绍.....	18
表 1-16 Port 介绍.....	20
表 1-17 Port 介绍.....	21
表 1-18 Port 介绍.....	23
表 1-19 Port 介绍.....	25
表 1-20 Port 介绍.....	27
表 1-21 Port 介绍.....	29
表 1-22 Port 介绍.....	32
表 1-23 Port 介绍.....	34
表 1-24 Port 介绍.....	36
表 1-25 Port 介绍.....	38
表 1-26 Port 介绍.....	40
表 1-27 Port 介绍.....	42
表 1-28 Port 介绍.....	44
表 1-29 Port 介绍.....	46
表 1-30 Port 介绍.....	48
表 1-31 Port 介绍.....	51
表 1-32 Port 介绍.....	53
表 1-33 Port 介绍.....	56
表 1-34 Attribute 介绍.....	56
表 1-35 Port 介绍.....	57
表 1-36 Attribute 介绍.....	58
表 2-1 Port 介绍.....	61
表 2-2 Attribute 介绍.....	61
表 2-3 真值表.....	61
表 2-4 Port 介绍.....	62
表 2-5 Attribute 介绍.....	62
表 2-6 真值表.....	63
表 2-7 Port 介绍.....	64
表 2-8 Attribute 介绍.....	64
表 2-9 真值表.....	64
表 2-10 Port 介绍.....	65
表 2-11 Attribute 介绍.....	65
表 2-12 真值表.....	66

表 2-13 Port 介绍.....	67
表 2-14 真值表.....	67
表 2-15 Port 介绍.....	70
表 2-16 真值表.....	70
表 2-17 Port 介绍.....	71
表 2-18 真值表.....	71
表 2-19 Port 介绍.....	73
表 2-20 真值表.....	73
表 2-21 ALU 功能.....	75
表 2-22 Port 介绍.....	75
表 2-23 Attribute 介绍.....	76
表 2-24 与 FF 相关的原语.....	77
表 2-25 Port 介绍.....	78
表 2-26 Attribute 介绍.....	78
表 2-27 Port 介绍.....	79
表 2-28 Attribute 介绍.....	79
表 2-29 Port 介绍.....	80
表 2-30 Attribute 介绍.....	81
表 2-31 Port 介绍.....	82
表 2-32 Attribute 介绍.....	82
表 2-33 Port 介绍.....	83
表 2-34 Attribute 介绍.....	83
表 2-35 Port 介绍.....	84
表 2-36 Attribute 介绍.....	84
表 2-37 Port 介绍.....	86
表 2-38 Attribute 介绍.....	86
表 2-39 Port 介绍.....	87
表 2-40 Attribute 介绍.....	87
表 2-41 Port 介绍.....	88
表 2-42 Attribute 介绍.....	88
表 2-43 Port 介绍.....	90
表 2-44 Attribute 介绍.....	90
表 2-45 Port 介绍.....	91
表 2-46 Attribute 介绍.....	91
表 2-47 Port 介绍.....	92
表 2-48 Attribute 介绍.....	92
表 2-49 Port 介绍.....	93

表 2-50 Attribute 介绍 .....	93
表 2-51 Port 介绍 .....	95
表 2-52 Attribute 介绍 .....	95
表 2-53 Port 介绍 .....	96
表 2-54 Attribute 介绍 .....	96
表 2-55 Port 介绍 .....	97
表 2-56 Attribute 介绍 .....	97
表 2-57 Port 介绍 .....	99
表 2-58 Attribute 介绍 .....	99
表 2-59 Port 介绍 .....	100
表 2-60 Attribute 介绍 .....	100
表 2-61 Port 介绍 .....	101
表 2-62 Attribute 介绍 .....	101
表 2-63 Port 介绍 .....	103
表 2-64 Attribute 介绍 .....	103
表 2-65 与 LATCH 相关的原语 .....	104
表 2-66 Port 介绍 .....	105
表 2-67 Attribute 介绍 .....	105
表 2-68 Port 介绍 .....	106
表 2-69 Attribute 介绍 .....	106
表 2-70 Port 介绍 .....	107
表 2-71 Attribute 介绍 .....	107
表 2-72 Port 介绍 .....	109
表 2-73 Attribute 介绍 .....	109
表 2-74 Port 介绍 .....	110
表 2-75 Attribute 介绍 .....	110
表 2-76 Port 介绍 .....	111
表 2-77 Attribute 介绍 .....	111
表 2-78 Port 介绍 .....	113
表 2-79 Attribute 介绍 .....	113
表 2-80 Port 介绍 .....	114
表 2-81 Attribute 介绍 .....	114
表 2-82 Port 介绍 .....	115
表 2-83 Attribute 介绍 .....	115
表 2-84 Port 介绍 .....	117
表 2-85 Attribute 介绍 .....	117
表 2-86 Port 介绍 .....	118

表 2-87 Attribute 介绍 .....	118
表 2-88 Port 介绍 .....	119
表 2-89 Attribute 介绍 .....	119
表 3-1 SSRAM.....	121
表 3-2 Port 介绍 .....	122
表 3-3 Attribute 介绍 .....	122
表 3-4 Port 介绍 .....	123
表 3-5 Attribute 介绍 .....	124
表 3-6 Port 介绍 .....	125
表 3-7 Attribute 介绍 .....	125
表 3-8 Port 介绍 .....	127
表 3-9 Attribute 介绍 .....	127
表 3-10 Port 介绍 .....	128
表 3-11 Attribute 介绍 .....	128
表 3-12 Port 介绍 .....	130
表 3-13 Attribute 介绍 .....	130
表 3-14 Port 介绍 .....	132
表 3-15 Attribute 介绍 .....	132
表 4-1 Port 介绍 .....	134
表 4-2 Attribute 介绍 .....	135
表 4-3 配置关系 .....	135
表 4-4 Port 介绍 .....	139
表 4-5 Attribute 介绍 .....	140
表 4-6 配置关系 .....	140
表 4-7 Port 介绍 .....	145
表 4-8 Attribute 介绍 .....	146
表 4-9 配置关系 .....	146
表 4-10 Port 介绍 .....	151
表 4-11 Attribute 介绍 .....	152
表 4-12 配置关系 .....	152
表 5-1 Port 介绍 .....	157
表 5-2 Attribute 介绍 .....	157
表 5-3 Port 介绍 .....	159
表 5-4 Attribute 介绍 .....	160
表 5-5 Port 介绍 .....	162
表 5-6 Attribute 介绍 .....	163
表 5-7 Port 介绍 .....	165

表 5-8 Attribute 介绍 .....	166
表 5-9 Port 介绍 .....	169
表 5-10 Attribute 介绍 .....	169
表 5-11 Port 介绍 .....	171
表 5-12 Attribute 介绍 .....	172
表 5-13 Port 介绍 .....	175
表 5-14 Attribute 介绍 .....	176
表 5-15 Port 介绍 .....	180
表 5-16 Attribute 介绍 .....	181
表 5-17 Port 介绍 .....	185
表 5-18 Attribute 介绍 .....	186
表 6-1 PLL 性能 .....	190
表 6-2 Port 介绍 .....	191
表 6-3 Attribute 介绍 .....	192
表 6-4 Port 介绍 .....	197
表 6-5 Attribute 介绍 .....	197
表 6-6 Port 介绍 .....	199
表 6-7 Attribute 介绍 .....	199
表 6-8 Port 介绍 .....	201
表 6-9 Attribute 介绍 .....	201
表 6-10 Port 介绍 .....	202
表 6-11 Port 介绍 .....	204
表 6-12 Attribute 介绍 .....	204
表 6-13 Port 介绍 .....	205
表 6-14 Attribute 介绍 .....	206
表 6-15 Port 介绍 .....	209
表 6-16 Attribute 介绍 .....	209
表 6-17 Port 介绍 .....	210
表 6-18 Attribute 介绍 .....	210
表 6-19 Port 介绍 .....	211
表 6-20 Attribute 介绍 .....	211
表 6-21 Port 介绍 .....	212
表 6-22 Port 介绍 .....	213
表 6-23 Port 介绍 .....	214
表 7-1 Port 介绍 .....	216
表 7-2 Port 介绍 .....	218
表 7-3 Port 介绍 .....	220

---

表 7-4 Port 介绍.....	222
表 7-5 Port 介绍.....	224
表 8-1 Port 介绍.....	228
表 8-2 Port 介绍.....	238
表 8-3 Attribute 介绍.....	240
表 8-4 Port 介绍.....	245

# 1 IOB

## 1.1 Buffer/LVDS

Buffer，缓冲器，具有缓存功能。根据不同功能，可分为普通 buffer、模拟 LVDS（ELVDS）和真 LVDS（TLVDS）。

### 1.1.1 IBUF

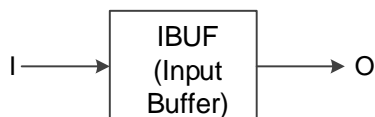
#### 原语介绍

IBUF(Input Buffer)，输入缓冲器。

支持器件：GW1N-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、GW1NR-9、GW1NZ-1、GW1NSR-2、GW1NSR-2C、GW2A-18、GW2AR-18、GW2A-55。

#### 结构框图

图 1-1 IBUF 结构框图



#### Port 介绍

表 1-1 Port 介绍

Port Name	I/O	Description
I	Input	Data Input
O	Output	Data Output

#### 原语例化

**Verilog 例化：**

```

IBUF uut(
    .O(O),
    .I(I)
  )
  
```



```

);
Vhdl 例化:
COMPONENT IBUF
  PORT (
    O:OUT std_logic;
    I:IN std_logic
  );
END COMPONENT;
 uut:IBUF
  PORT MAP(
    O=>O,
    I=>I
  );

```

## 1.1.2 OBUF

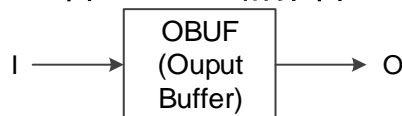
### 原语介绍

OBUF(Output Buffer), 输出缓冲器。

支持器件: GW1N-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、GW1NR-9、GW1NZ-1、GW1NSR-2、GW1NSR-2C、GW2A-18、GW2AR-18、GW2A-55。

### 结构框图

图 1-2 OBUF 结构框图



### Port 介绍

表 1-2 Port 介绍

Port Name	I/O	Description
I	Input	Data Input
O	Output	Data Output

### 原语例化

```

Verilog 例化:
  OBUF uut(
    .O(O),
    .I(I)
  );
Vhdl 例化:
COMPONENT OBUF
  PORT (
    O:OUT std_logic;
    I:IN std_logic
  );
END COMPONENT;
 uut:OBUF

```

```

PORT MAP(
    O=>O,
    I=>I
);

```

### 1.1.3 TBUF

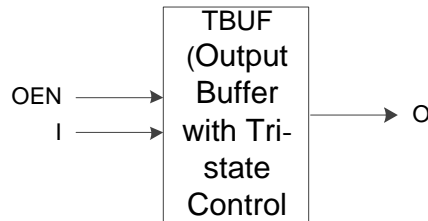
#### 原语介绍

TBUF(Output Buffer with Tri-state Control), 三态缓冲器, 低电平使能。

支持器件: GW1N-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、GW1NR-9、GW1NZ-1、GW1NSR-2、GW1NSR-2C、GW2A-18、GW2AR-18、GW2A-55。

#### 结构框图

图 1-3 TBUF 结构框图



#### Port 介绍

表 1-3 Port 介绍

Port Name	I/O	Description
I	Input	Data Input
OEN	Input	Output Enable
O	Output	Data Output

#### 原语示例例化

##### Verilog 例化:

```

TBUF uut(
    .O(O),
    .I(I),
    .OEN(OEN)
);

```

##### Vhdl 例化:

```

COMPONENT TBUF
PORT (
    O:OUT std_logic;
    I:IN std_logic;
    OEN:IN std_logic
);
END COMPONENT;
uut:TBUF
PORT MAP(

```

```

O=>O,
I=>I,
OEN=> OEN
);

```

## 1.1.4 IOBUF

### 原语介绍

IOBUF(Bi-Directional Buffer), 双向缓冲器。当 OEN 为高电平时, 作为输入缓冲器; OEN 为低电平时, 作为输出缓冲器。

支持器件: GW1N-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、GW1NR-9、GW1NZ-1、GW1NSR-2、GW1NSR-2C、GW2A-18、GW2AR-18、GW2A-55。

### 结构框图

图 1-4 IOBUF 结构框图



### PORT 介绍

表 1-4 Port 介绍

Port Name	I/O	Description
I	Input	Data Input
OEN	Input	Output Enable
IO	Inout	Inout Port
O	Output	Data Output

### 原语例化

#### Verilog 例化:

```

IOBUF uut(
    .O(O),
    .IO(IO),
    .I(I),
    .OEN(OEN)
);

```

#### Vhdl 例化:

```

COMPONENT IOBUF
PORT (
    O:OUT std_logic;
    IO:INOUT std_logic;
    I:IN std_logic;
    OEN:IN std_logic
);
END COMPONENT;

```

```

 uut:IOBUF
   PORT MAP(
     O=>O,
     IO=>IO,
     I=>I,
     OEN=> OEN
   );

```

## 1.1.5 LVDS input buffer

### 原语介绍

LVDS 差分输入分为两种：TLVDS\_IBUF 和 ELVDS\_IBUF。

TLVDS\_IBUF(True LVDS Input Buffer)，真差分输入缓冲器。

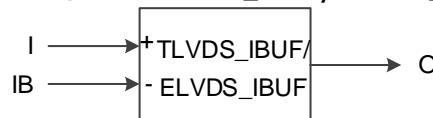
支持器件：GW1N-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、GW1NR-9、GW1NSR-2、GW1NSR-2C、GW2A-18、GW2AR-18、GW2A-55。

ELVDS\_IBUF(Emulated LVDS Input Buffer)，模拟差分输入缓冲器。

支持器件：GW1N-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、GW1NR-9、GW1NZ-1、GW1NSR-2、GW1NSR-2C、GW2A-18、GW2AR-18、GW2A-55。

### 结构框图

图 1-5 TLVDS\_IBUF/ELVDS\_IBUF 结构框图



### Port 介绍

表 1-5 Port 介绍

Port Name	I/O	Description
I	Input	Differential Input
IB	Input	Differential Input
O	Output	Data Output

### 原语例化

示例一

#### Verilog 例化:

```

 TLVDS_IBUF uut(
   .O(O),
   .I(I),
   .IB(IB)
 );

```

#### Vhdl 例化:

```

 COMPONENT TLVDS_IBUF
 PORT (
   O:OUT std_logic;

```

```

        I:IN std_logic;
        IB:IN std_logic
    );
END COMPONENT;
 uut:TLVDS_IBUF
    PORT MAP(
        O=>O,
        I=>I,
        IB=> IB
    );

```

示例二

#### Verilog 例化:

```

ELVDS_IBUF uut(
    .O(O),
    .I(I),
    .IB(IB)
);

```

#### Vhdl 例化:

```

COMPONENT ELVDS_IBUF
    PORT (
        O:OUT std_logic;
        I:IN std_logic;
        IB:IN std_logic
    );
END COMPONENT;
 uut:ELVDS_IBUF
    PORT MAP(
        O=>O,
        I=>I,
        IB=> IB
    );

```

## 1.1.6 LVDS output buffer

### 原语介绍

LVDS 差分输出分为两种: TLVDS\_OBUF 和 ELVDS\_OBUF。

TLVDS\_OBUF(True LVDS Output Buffer), 真差分输出缓冲器。

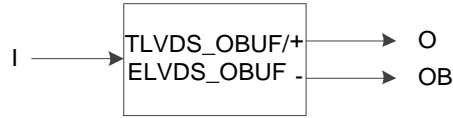
支持器件: GW1N-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、GW1NR-9、GW1NSR-2、GW1NSR-2C、GW2A-18、GW2AR-18、GW2A-55。

ELVDS\_OBUF(Emulated LVDS Output Buffer), 模拟差分输出缓冲器。

支持器件: GW1N-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、GW1NR-9、GW1NZ-1、GW1NSR-2、GW1NSR-2C、GW2A-18、GW2AR-18、GW2A-55。

## 结构框图

图 1-6 TLVDS\_OBUF/ELVDS\_OBUF 结构框图



## Port 介绍

表 1-6 Port 介绍

Port Name	I/O	Description
I	Input	Data Input
OB	Output	Differential Output
O	Output	Differential Output

## 原语例化

示例一

### Verilog 例化:

```

TLVDS_OBUF uut(
    .O(O),
    .OB(OB),
    .I(I)
);
  
```

### Vhdl 例化:

```

COMPONENT TLVDS_OBUF
  PORT (
    O:OUT std_logic;
    OB:OUT std_logic;
    I:IN std_logic
  );
END COMPONENT;
uut:TLVDS_OBUF
  PORT MAP(
    O=>O,
    OB=>OB,
    I=> I
  );
  
```

示例二

### Verilog 例化:

```

ELVDS_OBUF uut(
    .O(O),
    .OB(OB),
    .I(I)
);
  
```

### Vhdl 例化:

```

COMPONENT ELVDS_OBUF
  PORT (
    O:OUT std_logic;
  );
  
```

```

        OB:OUT std_logic;
        I:IN std_logic
    );
END COMPONENT;
 uut:ELVDS_OBUF
    PORT MAP(
        O=>O,
        OB=>OB,
        I=> I
    );

```

## 1.1.7 LVDS tristate buffer

### 原语介绍

LVDS 三态差分输出分为两种：TLVDS\_TBUF 和 ELVDS\_TBUF。

TLVDS\_TBUF(True LVDS Tristate Buffer)，真差分三态缓冲器，低电平使能。

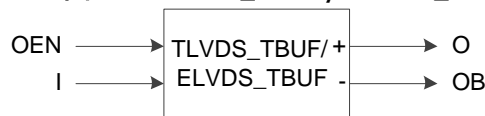
支持器件：GW1N-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、GW1NR-9、GW1NSR-2、GW1NSR-2C、GW2A-18、GW2AR-18、GW2A-55。

ELVDS\_TBUF(Emulated LVDS Tristate Buffer)，模拟差分三态缓冲器，低电平使能。

支持器件：GW1N-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、GW1NR-9、GW1NZ-1、GW1NSR-2、GW1NSR-2C、GW2A-18、GW2AR-18、GW2A-55。

### 结构框图

图 1-7 TLVDS\_TBUF/ELVDS\_TBUF 结构框图



### Port 介绍

表 1-7 Port 介绍

Port Name	I/O	Description
I	Input	Data Input
OEN	Input	Output Enable
OB	Output	Differential Output
O	Output	Differential Output

### 原语例化

示例一

**Verilog 例化:**

```
TLVDS_TBUF uut(
```

```

.O(O),
.OB(OB),
.I(I),
.OEN(OEN)
);

```

**Vhdl 例化:**

```

COMPONENT TLVDS_TBUF
PORT (
    O:OUT std_logic;
    OB:OUT std_logic;
    I:IN std_logic;
    OEN:IN std_logic
);
END COMPONENT;
uut:TLVDS_TBUF
PORT MAP(
    O=>O,
    OB=>OB,
    I=> I,
    OEN=>OEN
);

```

## 示例二

**Verilog 例化:**

```

ELVDS_TBUF uut(
.O(O),
.OB(OB),
.I(I),
.OEN(OEN)
);

```

**Vhdl 例化:**

```

COMPONENT ELVDS_TBUF
PORT (
    O:OUT std_logic;
    OB:OUT std_logic;
    I:IN std_logic;
    OEN:IN std_logic
);
END COMPONENT;
uut:ELVDS_TBUF
PORT MAP(
    O=>O,
    OB=>OB,
    I=> I,
    OEN=>OEN
);

```



## 1.1.8 LVDS inout buffer

### 原语介绍

LVDS 差分输入输出分为两种：TLVDS\_IOBUF 和 ELVDS\_IOBUF。

TLVDS\_IOBUF(True LVDS Bi-Directional Buffer)，真差分双向缓冲器，当 OEN 为高电平时，作为真差分输入缓冲器；OEN 为低电平时，作为真差分输出缓冲器

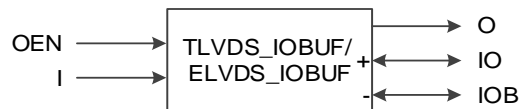
支持器件：GW1N-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、GW1NR-9、GW1NSR-2、GW1NSR-2C、GW2A-18、GW2AR-18、GW2A-55。

ELVDS\_IOBUF(Emulated LVDS Bi-Directional Buffer)，模拟差分双向缓冲器，当 OEN 为高电平时，作为模拟差分输入缓冲器；OEN 为低电平时，作为模拟差分输出缓冲器

支持器件：GW1N-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、GW1NR-9、GW1NZ-1、GW1NSR-2、GW1NSR-2C、GW2A-18、GW2AR-18、GW2A-55。

### 结构框图

图 1-8 TLVDS\_IOBUF/ELVDS\_IOBUF 结构框图



### Port 介绍

表 1-8 Port 介绍

Port Name	I/O	Description
I	Input	Data Input
OEN	Input	Output Enable
O	Output	Data Output
IOB	Inout	Differential Inout
IO	Inout	Differential Inout

### 原语例化

#### Verilog 例化:

```
ELVDS_IOBUF uut(
    .O(O),
    .IO(IO),
    .IOB(IOB),
    .I(I),
    .OEN(OEN)
);
```

#### Vhdl 例化:

```
COMPONENT ELVDS_IOBUF
PORT (
```

```

O:OUT std_logic;
IO:INOUT std_logic;
IOB:INOUT std_logic;
I:IN std_logic;
OEN:IN std_logic

);
END COMPONENT;
uut:ELVDS_IOBUF
PORT MAP(
O=>O,
IO=>IO,
IOB=>IOB,
I=> I,
OEN=>OEN
);

```

## 1.1.9 MIPI\_IBUF\_HS

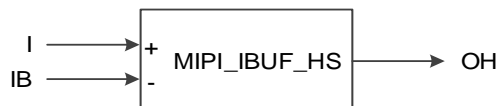
### 原语介绍

MIPI\_IBUF\_HS(MIPI High Speed Input Buffer ), MIPI 高速输入缓冲器。

支持器件: GW1NS-2、GW1NS-2C、GW1N-6、GW1N-9、GW1NR-9、GW1NSR-2、GW1NSR-2C。

### 结构框图

图 1-9 MIPI\_IBUF\_HS 结构框图



### Port 介绍

表 1-9 Port 介绍

Port Name	I/O	Description
I	Input	Differential Input
IB	Input	Differential Input
OH	Output	Data Output

### 原语例化

#### Verilog 例化:

```

MIPI_IBUF_HS uut(
.OH(OH),
.I(I),
.IB(IB)
);

```

#### Vhdl 例化:

```

COMPONENT MIPI_IBUF_HS
PORT (
OH:OUT std_logic;
I:IN std_logic;

```

```

        IB:IN std_logic
    );
END COMPONENT;
uut: MIPI_IBUF_HS
    PORT MAP(
        OH=>OH,
        I=>I,
        IB=>IB
    );

```

## 1.1.10 MIPI\_IBUF\_LP

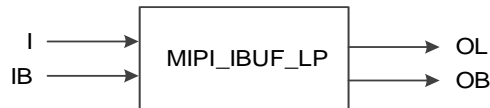
### 原语介绍

MIPI\_IBUF\_LP(MIPI Low Power Input Buffer), MIPI 低功耗输入缓冲器。

支持器件: GW1NS-2、GW1NS-2C、GW1N-6、GW1N-9、GW1NR-9、GW1NSR-2、GW1NSR-2C。

### 结构框图

图 1-10 MIPI\_IBUF\_LP 结构框图



### Port 介绍

表 1-10 Port 介绍

Port Name	I/O	Description
I	Input	Data Input
IB	Input	Data Input
OL	Output	Data Output
OB	Output	Data Output

### 原语例化

#### Verilog 例化:

```

MIPI_IBUF_LP uut(
    .OL(OL),
    .OB(OB),
    .I(I),
    .IB(IB)
);

```

#### Vhdl 例化:

```

COMPONENT MIPI_IBUF_LP
    PORT (
        OL:OUT std_logic;
        OB:OUT std_logic;
        I:IN std_logic;
        IB:IN std_logic
    );

```

```

END COMPONENT;
 uut: MIPI_IBUF_LP
   PORT MAP(
     OL=>OL,
     OB=>OB,
     I=>I,
     IB=>IB
   );

```

### 1.1.11 MIPI\_OBUF

#### 原语介绍

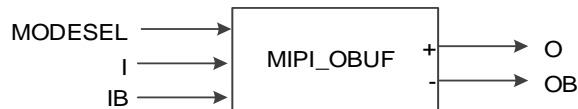
MIPI\_OBUF 有两种工作模式：HS 模式和 LP 模式。

MIPI\_OBUF(MIPI Output Buffer)，MIPI 输出缓冲器，当 MODESEL 为高电平时，作为(HS)MIPI 高速输出缓冲器；当 MODESEL 为低电平时，作为(LP)MIPI 低功耗输出缓冲器。

支持器件：GW1NS-2、GW1NS-2C、GW1N-6、GW1N-9、GW1NR-9、GW1NSR-2、GW1NSR-2C。

#### 结构框图

图 1-11 MIPI\_OBUF 结构框图



#### Port 介绍

表 1-11 Port 介绍

Port Name	I/O	Description
I	Input	Data Input
IB	Input	Data Input
MODESEL	Input	Mode Selection, HS or LP
O	Output	Data Output
OB	Output	Data Output

#### 原语例化

##### Verilog 例化:

```

MIPI_OBUF uut(
  .O(O),
  .OB(OB),
  .I(I),
  .IB(IB),
  .MODESEL(MODESEL)
);

```

##### Vhdl 例化:

```

COMPONENT MIPI_OBUF
  PORT (

```

```

O:OUT std_logic;
OB:OUT std_logic;
I:IN std_logic;
IB:IN std_logic;
MODESEL:IN std_logic
);
END COMPONENT;
 uut: MIPI_OBUF
    PORT MAP(
        O=>O,
        OB=>OB,
        I=>I,
        IB=>IB,
        MDOESEL=>MODESEL
    );

```

### 1.1.12 I3C\_IOBUF

#### 原语介绍

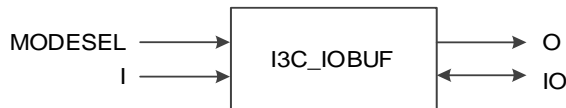
I3C\_IOBUF 有两种工作模式：Normal 模式和 I3C 模式。

I3C\_IOBUF( I3C Bi-Directional Buffer), I3C 双向缓冲器，当 MODESEL 为高电平时，作为 I3C 双向缓冲器；当 MODESEL 为低电平时，作为普通双向缓冲器。

支持器件：GW1NS-2、GW1NS-2C、GW1N-6、GW1N-9、GW1NR-9、GW1NSR-2、GW1NSR-2C。

#### 结构框图

图 1-12 I3C\_IOBUF 结构框图



#### Port 介绍

表 1-12 Port 介绍

Port Name	I/O	Description
I	Input	Data Input
IO	Inout	Inout Port
MODESEL	Input	Mode Selection, Normal or I3C
O	Output	Data Output

#### 原语例化

##### Verilog 例化:

```

I3C_IOBUF uut(
    .O(O),
    .IO(IO),
    .I(I),
    .MODESEL(MODESEL)
)

```

```

);
Vhdl 例化:
COMPONENT I3C_IOBUF
  PORT (
    O:OUT std_logic;
    IO:INOUT std_logic;
    I:IN std_logic;
    MODESEL:IN std_logic
  );
END COMPONENT;
uut: I3C_IOBUF
  PORT MAP(
    O=>O,
    IO=>IO,
    I=>I,
    MDOESEL=>MODESEL
  );

```

## 1.2 IOLOGIC

### 1.2.1 IDDR

#### 原语介绍

IDDR(Dual Data Rate Input), 实现双倍数据速率输入。

支持器件: GW1N-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、GW1NR-9、GW1NZ-1、GW1NSR-2、GW1NSR-2C、GW2A-18、GW2AR-18、GW2A-55。

#### 结构框图

图 1-13 IDDR 结构框图



#### Port 介绍

表 1-13 Port 介绍

Port Name	I/O	Description
D	Input	Data Input, from port O of input buffer or port DO of IODELAY
CLK	Input	Clock Input
Q0	Output	Data Output
Q1	Output	Data Output

IDDR 的属性 Q0\_INIT 与 Q1\_INIT 表示对 IDDR 中 Register 的初始化。

#### 原语例化

#### Verilog 例化:

```
IDDR uut(
```

```

        .Q0(Q0),
        .Q1(Q1),
        .D(D),
        .CLK(CLK)
    );
    defparam uut.Q0_INIT = 1'b0;
    defparam uut.Q1_INIT = 1'b0;
Vhdl 例化:
    COMPONENT IDDR
        GENERIC (Q0_INIT:bit:='0';
                Q1_INIT:bit:='0'
        );
        PORT(
            Q0:OUT std_logic;
            Q1:OUT std_logic;
            D:IN std_logic;
            CLK:IN std_logic
        );
    END COMPONENT;
    uut:IDDR
        GENERIC MAP (Q0_INIT=>'0',
                    Q1_INIT=>'0'
        )
        PORT MAP (
            Q0=>Q0,
            Q1=>Q1,
            D=>D,
            CLK=>CLK
        );

```

## 1.2.2 ODDR

### 原语介绍

ODDR(Dual Data Rate Output), 实现双倍数据速率输出。

支持器件: GW1N-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、GW1NR-9、GW1NZ-1、GW1NSR-2、GW1NSR-2C、GW2A-18、GW2AR-18、GW2A-55。

### 结构框图

图 1-14 ODDR 结构框图



## Port 介绍

表 1-14 Port 介绍

Port Name	I/O	Description
D0	Input	Data Input
D1	Input	Data Input
TX	Input	Data Input
CLK	Input	Clock Input
Q0	Output	Data Output, to port I of output buffer or port DI of IODELAY
Q1	Output	Tristate Enable Output, to port OEN of tristate/inout buffer(Q0 connected)or dangling

ODDR 的属性 INIT 表示对 ODDR 中 Register 的初始化。

### 原语例化

#### Verilog 例化:

```

ODDR uut(
    .Q0(Q0),
    .Q1(Q1),
    .D0(D0),
    .D1(D1),
    .TX(TX),
    .CLK(CLK)
);
defparam uut.INIT=1'b0;
defparam uut.TXCLK_POL=1'b0;

```

#### Vhdl 例化:

```

COMPONENT ODDR
    GENERIC (CONSTANT INIT:bit='0';
             TXCLK_POL:bit='0'
    );
    PORT(
        Q0:OUT std_logic;
        Q1:OUT std_logic;
        D0:IN std_logic;
        D1:IN std_logic;
        TX:IN std_logic;
        CLK:IN std_logic
    );
END COMPONENT;
uut:ODDR
    GENERIC MAP (INIT=>'0',
                TXCLK_POL=>'0'
    )
    PORT MAP (
        Q0=>Q0,
        Q1=>Q1,

```



```

D0=>D0,
D1=>D1,
TX=>TX,
CLK=>CLK
);

```

## 1.2.3 IDDRC

### 原语介绍

IDDRC(Dual Data Rate Input with Asynchronous Clear)与 IDDR 功能类似，实现双倍速率输入，同时具有异步复位功能。

支持器件：GW1N-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、GW1NR-9、GW1NZ-1、GW1NSR-2、GW1NSR-2C、GW2A-18、GW2AR-18、GW2A-55。

### 结构框图

图 1-15 IDDRC 结构框图



### Port 介绍

表 1-15 Port 介绍

Port Name	I/O	Description
D	Input	Data Input, from port O of input buffer or port DO of IODELAY
CLK	Input	Clock Input
CLEAR	Input	Asynchronous Clear Input
Q0	Output	Data Output
Q1	Output	Data Output

IDDRC 的属性 Q0\_INIT 和 Q1\_INIT 表示对 IDDRC 中 Register 的初始化。

### 原语例化

#### Verilog 例化:

```

IDDRC uut(
    .Q0(Q0),
    .Q1(Q1),
    .D(D),
    .CLK(CLK),
    .CLEAR(CLEAR)
);
defparam uut.Q0_INIT = 1'b0;
defparam uut.Q1_INIT = 1'b0;

```

#### Vhdl 例化:

```

COMPONENT IDDRC
  GENERIC (Q0_INIT:bit:='0';
          Q1_INIT:bit:='0'
  );
  PORT(
    Q0:OUT std_logic;
    Q1:OUT std_logic;
    D:IN std_logic;
    CLEAR:IN std_logic;
    CLK:IN std_logic
  );
END COMPONENT;
 uut:IDDRC
  GENERIC MAP (Q0_INIT=>'0',
              Q1_INIT=>'0'
  )
  PORT MAP (
    Q0=>Q0,
    Q1=>Q1,
    D=>D,
    CLEAR=>CLEAR,
    CLK=>CLK
  );

```

## 1.2.4 ODDRC

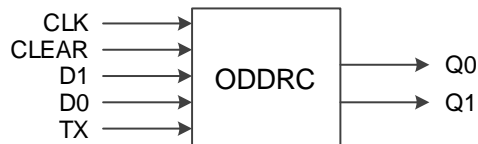
### 原语介绍

ODDRC(Dual Data Rate Output with Asynchronous Clear)与 ODDR 功能类似，实现双倍速率输出，同时具有异步复位功能。

支持器件：GW1N-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、GW1NR-9、GW1NZ-1、GW1NSR-2、GW1NSR-2C、GW2A-18、GW2AR-18、GW2A-55。

### 结构框图

图 1-16 ODDRC 结构框图



## Port 介绍

表 1-16 Port 介绍

Port Name	I/O	Description
D0	Input	Data Input
D1	Input	Data Input
TX	Input	Data Input
CLK	Input	Clock Input
CLEAR	Input	Asynchronous Clear Input
Q0	Output	Data Output, to port I of output buffer or port DI of IODELAY
Q1	Output	Tristate Enable Output, to port OEN of tristate/inout buffer(Q0 connected)or dangling

ODDRC 的属性 INIT 是对 ODDRC 中 Register 的初始化。

### 原语例化

#### Verilog 例化:

```

ODDRC uut(
    .Q0(Q0),
    .Q1(Q1),
    .D0(D0),
    .D1(D1),
    .TX(TX),
    .CLK(CLK),
    .CLEAR(CLEAR)
);
defparam uut.INIT=1'b0;
defparam uut.TXCLK_POL=1'b0;

```

#### Vhdl 例化:

```

COMPONENT ODDRC
    GENERIC (CONSTANT INIT:bit='0';
            TXCLK_POL:bit='0'
    );
    PORT(
        Q0:OUT std_logic;
        Q1:OUT std_logic;
        D0:IN std_logic;
        D1:IN std_logic;
        TX:IN std_logic;
        CLK:IN std_logic;
        CLEAR:IN std_logic
    );
END COMPONENT;
uut:ODDRC
    GENERIC MAP (INIT=>'0',
                TXCLK_POL=>'0'
    )
    PORT MAP (

```

```

Q0=>Q0,
Q1=>Q1,
D0=>D0,
D1=>D1,
TX=>TX,
CLK=>CLK,
CLEAR=>CLEAR
);

```

## 1.2.5 IDES4

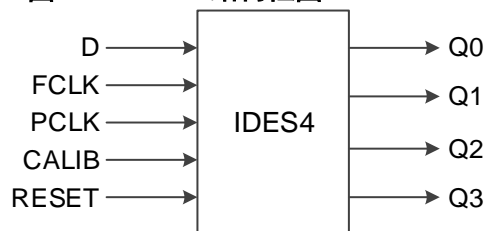
### 原语介绍

IDES4(1 to 4 Deserializer)为 1 位串行输入、4 位并行输出的解串器。

支持器件: GW1N-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、GW1NR-9、GW1NZ-1、GW1NSR-2、GW1NSR-2C、GW2A-18、GW2AR-18、GW2A-55。

### 结构框图

图 1-17 IDES4 结构框图



### Port 介绍

表 1-17 Port 介绍

Port Name	I/O	Description
D	Input	Data Input, from port O of input buffer or port DO of IODELAY
FCLK	Input	Fast Clock Input
PCLK	Input	Primary Clock Input
CALIB	Input	Calib Signal, adjust Output
RESET	Input	Asynchronous Reset Input
Q3~Q0	Output	Data Output

PCLK 通常由 FCLK 分频获得,  $f_{PCLK} = 1/2 f_{FCLK}$ 。

### 原语例化

#### Verilog 例化:

```

IDES4 uut(
    .Q0(Q0),
    .Q1(Q1),
    .Q2(Q2),
    .Q3(Q3),

```

```

        .D(D),
        .FCLK(FCLK),
        .PCLK(PCLK),
        .CALIB(CALIB),
        .RESET(RESET)
    );
    defparam uut. GSREN="false";
    defparam uut. LSREN ="true";
Vhdl 例化:
    COMPONENT IDES4
        GENERIC (GSREN:string:="false";
                LSREN:string:="true"
        );
        PORT(
            Q0:OUT std_logic;
            Q1:OUT std_logic;
            Q2:OUT std_logic;
            Q3:OUT std_logic;
            D:IN std_logic;
            FCLK:IN std_logic;
            PCLK:IN std_logic;
            CALIB:IN std_logic;
            RESET:IN std_logic
        );
    END COMPONENT;
    uut:IDES4
        GENERIC MAP (GSREN=>"false",
                    LSREN=>"true"
        )
        PORT MAP (
            Q0=>Q0,
            Q1=>Q1,
            Q2=>Q2,
            Q3=>Q3,
            D=>D,
            FCLK=>FCLK,
            PCLK=>PCLK,
            CALIB=>CALIB,
            RESET=>RESET
        );

```

## 1.2.6 IDES8

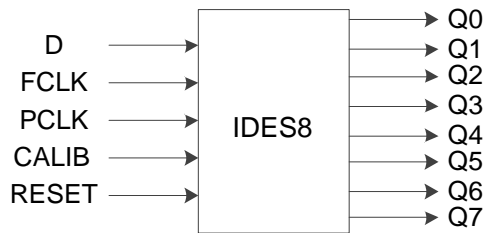
### 原语介绍

IDES8(1 to 8 Deserializer)为 1 位串行输入、8 位并行输出的解串器。

支持器件: GW1N-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、GW1NR-9、GW1NZ-1、GW1NSR-2、GW1NSR-2C、GW2A-18、GW2AR-18、GW2A-55。

## 结构框图

图 1-18 IDES8 结构框图



## Port 介绍

表 1-18 Port 介绍

Port Name	I/O	Description
D	Input	Data Input, from port O of input buffer or port DO of IODELAY
FCLK	Input	Fast Clock Input
PCLK	Input	Primary Clock Input
CALIB	Input	Calib Signal Input
RESET	Input	Asynchronous Reset Input
Q7~Q0	Output	Data Output

PCLK 通常由 FCLK 分频获得， $f_{PCLK} = 1/4 f_{FCLK}$ 。

## 原语例化

### Verilog 例化:

```

IDES8 uut(
    .Q0(Q0),
    .Q1(Q1),
    .Q2(Q2),
    .Q3(Q3),
    .Q4(Q4),
    .Q5(Q5),
    .Q6(Q6),
    .Q7(Q7),
    .D(D),
    .FCLK(FCLK),
    .PCLK(PCLK),
    .CALIB(CALIB),
    .RESET(RESET)
);
defparam uut. GSREN="false";
defparam uut. LSREN="true";

```

### Vhdl 例化:

```

COMPONENT IDES8
    GENERIC (GSREN:string:="false";
            LSREN:string:="true"

```

```

);
PORT(
    Q0:OUT std_logic;
    Q1:OUT std_logic;
    Q2:OUT std_logic;
    Q3:OUT std_logic;
    Q4:OUT std_logic;
    Q5:OUT std_logic;
    Q6:OUT std_logic;
    Q7:OUT std_logic;
    D:IN std_logic;
    FCLK:IN std_logic;
    PCLK:IN std_logic;
    CALIB:IN std_logic;
    RESET:IN std_logic
);
END COMPONENT;
 uut:IDES8
    GENERIC MAP (GSREN=>"false",
                LSREN=>"true"
    )
    PORT MAP (
        Q0=>Q0,
        Q1=>Q1,
        Q2=>Q2,
        Q3=>Q3,
        Q4=>Q4,
        Q5=>Q5,
        Q6=>Q6,
        Q7=>Q7,
        D=>D,
        FCLK=>FCLK,
        PCLK=>PCLK,
        CALIB=>CALIB,
        RESET=>RESET
    );

```

## 1.2.7 IDES10

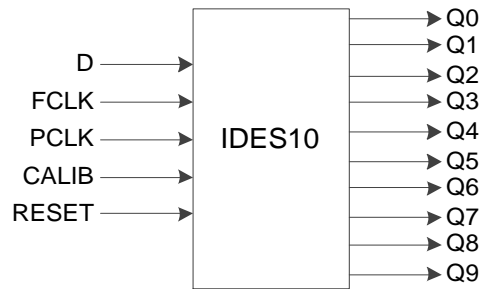
### 原语介绍

IDES10(1 to 10 Deserializer)为 1 位串行输入、10 位并行输出的解串器。

支持器件：GW1N-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、GW1NR-9、GW1NZ-1、GW1NSR-2、GW1NSR-2C、GW2A-18、GW2AR-18、GW2A-55。

## 结构框图

图 1-19 IDES10 结构框图



## Port 介绍

表 1-19 Port 介绍

Port Name	I/O	Description
D	Input	Data Input, from port O of input buffer or port DO of IODELAY
FCLK	Input	Fast Clock Input
PCLK	Input	Primary Clock Input
CALIB	Input	Calib Signal
RESET	Input	Asynchronous Reset Input
Q9~Q0	Output	Data Output

PCLK 通常由 FCLK 分频获得， $f_{PCLK} = 1/5 f_{FCLK}$ 。

## 原语例化

### Verilog 例化:

```

IDES10 uut(
    .Q0(Q0),
    .Q1(Q1),
    .Q2(Q2),
    .Q3(Q3),
    .Q4(Q4),
    .Q5(Q5),
    .Q6(Q6),
    .Q7(Q7),
    .Q8(Q8),
    .Q9(Q9),
    .D(D),
    .FCLK(FCLK),
    .PCLK(PCLK),
    .CALIB(CALIB),
    .RESET(RESET)
);
defparam uut. GSREN="false";
defparam uut. LSREN="true";

```

### Vhdl 例化:



```

COMPONENT IDES10
  GENERIC (GSREN:string=="false";
          LSREN:string=="true"
  );
  PORT(
    Q0:OUT std_logic;
    Q1:OUT std_logic;
    Q2:OUT std_logic;
    Q3:OUT std_logic;
    Q4:OUT std_logic;
    Q5:OUT std_logic;
    Q6:OUT std_logic;
    Q7:OUT std_logic;
    Q8:OUT std_logic;
    Q9:OUT std_logic;
    D:IN std_logic;
    FCLK:IN std_logic;
    PCLK:IN std_logic;
    CALIB:IN std_logic;
    RESET:IN std_logic
  );
END COMPONENT;
 uut:IDES10
  GENERIC MAP (GSREN=>"false",
              LSREN=>"true"
  )
  PORT MAP (
    Q0=>Q0,
    Q1=>Q1,
    Q2=>Q2,
    Q3=>Q3,
    Q4=>Q4,
    Q5=>Q5,
    Q6=>Q6,
    Q7=>Q7,
    Q8=>Q8,
    Q9=>Q9,
    D=>D,
    FCLK=>FCLK,
    PCLK=>PCLK,
    CALIB=>CALIB,
    RESET=>RESET
  );

```

## 1.2.8 IVIDEO

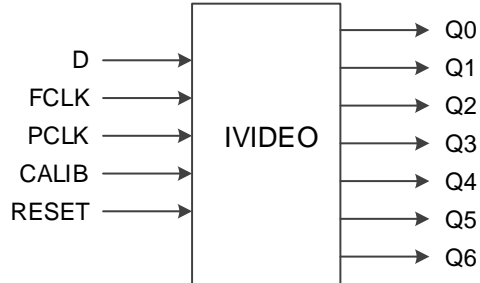
### 原语介绍

IVIDEO(1 to 7 Deserializer)为 1 位串行输入、7 位并行输出的解串器。  
支持器件：GW1N-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、

GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、  
GW1NR-9、GW1NZ-1、GW1NSR-2、GW1NSR-2C、GW2A-18、GW2AR-18、  
GW2A-55。

### 结构框图

图 1-20 IVIDEO 结构框图



### Port 介绍

表 1-20 Port 介绍

Port Name	I/O	Description
D	Input	Data Input, from port O of input buffer or port DO of IODELAY
FCLK	Input	Fast Clock Input
PCLK	Input	Primary Clock Input
CALIB	Input	Calib Signal Input
RESET	Input	Asynchronous Reset Input
Q6~Q0	Output	Data Output

PCLK 通常由 FCLK 分频而来， $f_{PCLK} = 1/3.5 f_{FCLK}$ 。

### 原语例化

#### Verilog 例化:

```
IVIDEO uut(
    .Q0(Q0),
    .Q1(Q1),
    .Q2(Q2),
    .Q3(Q3),
    .Q4(Q4),
    .Q5(Q5),
    .Q6(Q6),
    .D(D),
    .FCLK(FCLK),
    .PCLK(PCLK),
    .CALIB(CALIB),
    .RESET(RESET)
);
```

```
defparam uut. GSREN="false";
defparam uut. LSREN="true";
```

#### Vhdl 例化:

```
COMPONENT IVIDEO
  GENERIC (GSREN:string="false";
          LSREN:string="true"
  );
  PORT(
    Q0:OUT std_logic;
    Q1:OUT std_logic;
    Q2:OUT std_logic;
    Q3:OUT std_logic;
    Q4:OUT std_logic;
    Q5:OUT std_logic;
    Q6:OUT std_logic;
    D:IN std_logic;
    FCLK:IN std_logic;
    PCLK:IN std_logic;
    CALIB:IN std_logic;
    RESET:IN std_logic
  );
END COMPONENT;
 uut:IVIDEO
  GENERIC MAP (GSREN=>"false",
              LSREN=>"true"
  )
  PORT MAP (
    Q0=>Q0,
    Q1=>Q1,
    Q2=>Q2,
    Q3=>Q3,
    Q4=>Q4,
    Q5=>Q5,
    Q6=>Q6,
    D=>D,
    FCLK=>FCLK,
    PCLK=>PCLK,
    CALIB=>CALIB,
    RESET=>RESET
  );
```

## 1.2.9 IDES16

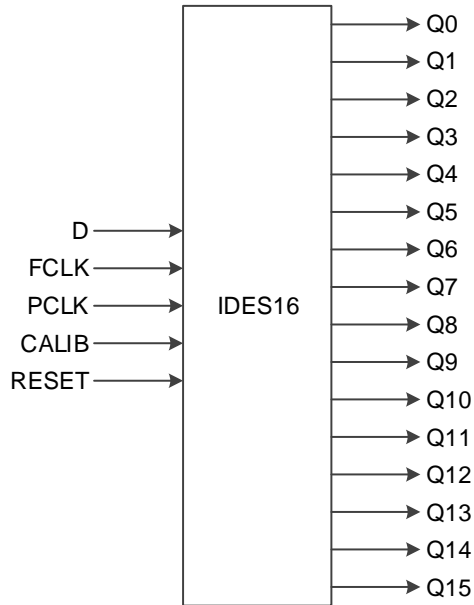
### 原语介绍

IDES16(1 to 16 Deserializer)为 1 位串行输入、16 位并行输出的解串器。

支持器件：GW1NS-2、GW1NS-2C、GW1N-6、GW1N-9、GW1NR-9、GW1NSR-2、GW1NSR-2C。

### 结构框图

图 1-21 IDES16 结构框图



### Port 介绍

表 1-21 Port 介绍

Port Name	I/O	Description
D	Input	Data Input, from port O of input buffer or port DO of IODELAY
FCLK	Input	Fast Clock Input
PCLK	Input	Primary Clock Input
CALIB	Input	Calib Signal
RESET	Input	Asynchronous Reset Input
Q15~Q0	Output	Data Output

PCLK 通常由 FCLK 分频获得， $f_{PCLK} = 1/8 f_{FCLK}$ 。

### 原语例化

#### Verilog 例化:

```
IDES16 uut(
    .Q0(Q0),
    .Q1(Q1),
    .Q2(Q2),
    .Q3(Q3),
```

```

        .Q4(Q4),
        .Q5(Q5),
        .Q6(Q6),
        .Q7(Q7),
        .Q8(Q8),
        .Q9(Q9),
        .Q10(Q10),
        .Q11(Q11),
        .Q12(Q12),
        .Q13(Q13),
        .Q14(Q14),
        .Q15(Q15),
        .D(D),
        .FCLK(FCLK),
        .PCLK(PCLK),
        .CALIB(CALIB),
        .RESET(RESET)
    );
    defparam uut. GSREN="false";
    defparam uut. LSREN ="true";
Vhdl 例化:
    COMPONENT IDES16
        GENERIC (GSREN:string:="false";
                LSREN:string:="true"
        );
        PORT(
            Q0:OUT std_logic;
            Q1:OUT std_logic;
            Q2:OUT std_logic;
            Q3:OUT std_logic;
            Q4:OUT std_logic;
            Q5:OUT std_logic;
            Q6:OUT std_logic;
            Q7:OUT std_logic;
            Q8:OUT std_logic;
            Q9:OUT std_logic;
            Q10:OUT std_logic;
            Q11:OUT std_logic;
            Q12:OUT std_logic;
            Q13:OUT std_logic;
            Q14:OUT std_logic;
            Q15:OUT std_logic;
            D:IN std_logic;
            FCLK:IN std_logic;
            PCLK:IN std_logic;
            CALIB:IN std_logic;
            RESET:IN std_logic
        );
    END COMPONENT;
    uut:IDES16

```

```

    GENERIC MAP (GSREN=>"false",
                LSREN=>"true"
    )
    PORT MAP (
        Q0=>Q0,
        Q1=>Q1,
        Q2=>Q2,
        Q3=>Q3,
        Q4=>Q4,
        Q5=>Q5,
        Q6=>Q6,
        Q7=>Q7,
        Q8=>Q8,
        Q9=>Q9,
        Q10=>Q10,
        Q11=>Q11,
        Q12=>Q12,
        Q13=>Q13,
        Q14=>Q14,
        Q15=>Q15,
        D=>D,
        FCLK=>FCLK,
        PCLK=>PCLK,
        CALIB=>CALIB,
        RESET=>RESET
    );

```

## 1.2.10 OSER4

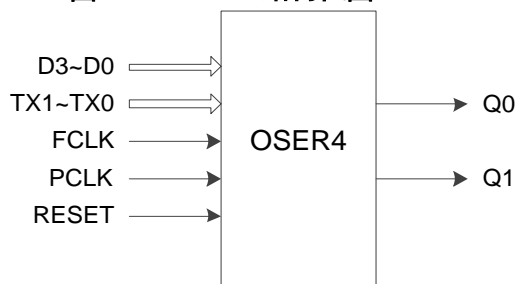
### 原语介绍

OSER4(4 to 1 Serializer)为 4 位并行输入、1 位串行输出的串化器。

支持器件: GW1N-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、GW1NR-9、GW1NZ-1、GW1NSR-2、GW1NSR-2C、GW2A-18、GW2AR-18、GW2A-55。

### 结构框图

图 1-22 OSER4 结构框图



## Port 介绍

表 1-22 Port 介绍

Port name	I/O	Description
D3~D0	Input	Data Input
TX1~TX0	Input	Data Input
FCLK	Input	Fast Clock Input
PCLK	Input	Primary Clock Input
RESET	Input	Asynchronous Reset Input
Q0	Output	Data Output, to port I of output buffer or port DI of IODELAY
Q1	Output	Tristate Enable Output, to port OEN of tristate/inout buffer(Q0 connected)or dangling

PCLK 通常由 FCLK 分频而获得,  $f_{PCLK} = 1/2 f_{FCLK}$ 。

### 原语例化

#### Verilog 例化:

```
OSER4 uut(
    .Q0(Q0),
    .Q1(Q1),
    .D0(D0),
    .D1(D1),
    .D2(D2),
    .D3(D3),
    .TX0(TX0),
    .TX1(TX1),
    .PCLK(PCLK),
    .FCLK(FCLK),
    .RESET(RESET)
);
defparam uut. GSREN="false";
defparam uut. LSREN="true";
defparam uut. HWL="false";
defparam uut. TXCLK_POL=1'b0;
```

#### Vhdl 例化:

```
COMPONENT OSER4
    GENERIC (GSREN:string:="false";
            LSREN:string:="true";
            HWL:string:="false";
            TXCLK_POL:bit:='0'
    );
    PORT(
        Q0:OUT std_logic;
        Q1:OUT std_logic;
        D0:IN std_logic;
        D1:IN std_logic;
        D2:IN std_logic;
```

```

        D3:IN std_logic;
        TX0:IN std_logic;
        TX1:IN std_logic;
        FCLK:IN std_logic;
        PCLK:IN std_logic;
        RESET:IN std_logic
    );
END COMPONENT;
 uut:OSER4
    GENERIC MAP (GSREN=>"false",
                 LSREN=>"true",
                 HWL=>"false",
                 TXCLK_POL=>'0'
    )
    PORT MAP (
        Q0=>Q0,
        Q1=>Q1,
        D0=>D0,
        D1=>D1,
        D2=>D2,
        D3=>D3,
        TX0=>TX0,
        TX1=>TX1,
        FCLK=>FCLK,
        PCLK=>PCLK,
        RESET=>RESET
    );

```

## 1.2.11 OSER8

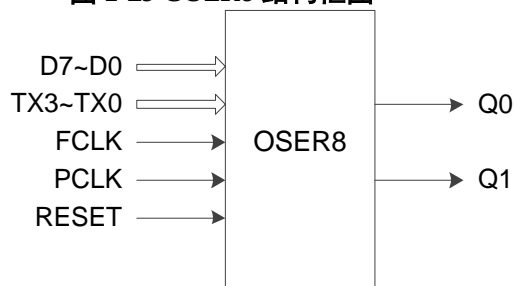
### 原语介绍

OSER8(8 to 1 Serializer)为 8 位并行输入、1 位串行输出的串化器。

支持器件：GW1N-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、GW1NR-9、GW1NZ-1、GW1NSR-2、GW1NSR-2C、GW2A-18、GW2AR-18、GW2A-55。

### 结构框图

图 1-23 OSER8 结构框图





## Port 介绍

表 1-23 Port 介绍

Port name	I/O	Description
D7~D0	Input	Data Input
TX3~TX0	Input	Data Input
FCLK	Input	Fast Clock Input
PCLK	Input	Primary Clock Input
RESET	Input	Asynchronous Reset Input
Q0	Output	Data Output, to port I of output buffer or port DI of IODELAY
Q1	Output	Tristate Enable Output, to port OEN of tristate/inout buffer(Q0 connected)or dangling

PCLK 通常由 FCLK 分频而得来,  $f_{PCLK} = 1/4 f_{FCLK}$ 。

### 原语例化

#### Verilog 例化:

```
OSER8 uut(
    .Q0(Q0),
    .Q1(Q1),
    .D0(D0),
    .D1(D1),
    .D2(D2),
    .D3(D3),
    .D4(D4),
    .D5(D5),
    .D6(D6),
    .D7(D7),
    .TX0(TX0),
    .TX1(TX1),
    .TX2(TX2),
    .TX3(TX3),
    .PCLK(PCLK),
    .FCLK(FCLK),
    .RESET(RESET)
);
defparam uut. GSREN="false";
defparam uut. LSREN="true";
defparam uut. HWL="false";
defparam uut. TXCLK_POL=1'b0;
```

#### Vhdl 例化:

```
COMPONENT OSER8
    GENERIC (GSREN:string:="false";
            LSREN:string:="true";
            HWL:string:="false";
            TXCLK_POL:bit:='0')
```

```
);
PORT(
    Q0:OUT std_logic;
    Q1:OUT std_logic;
    D0:IN std_logic;
    D1:IN std_logic;
    D2:IN std_logic;
    D3:IN std_logic;
    D4:IN std_logic;
    D5:IN std_logic;
    D6:IN std_logic;
    D7:IN std_logic;
    TX0:IN std_logic;
    TX1:IN std_logic;
    TX2:IN std_logic;
    TX3:IN std_logic;
    FCLK:IN std_logic;
    PCLK:IN std_logic;
    RESET:IN std_logic
);
END COMPONENT;
uut:OSER8
    GENERIC MAP (GSREN=>"false",
                LSREN=>"true",
                HWL=>"false",
                TXCLK_POL=>'0'
    )
    PORT MAP (
        Q0=>Q0,
        Q1=>Q1,
        D0=>D0,
        D1=>D1,
        D2=>D2,
        D3=>D3,
        D4=>D4,
        D5=>D5,
        D6=>D6,
        D7=>D7,
        TX0=>TX0,
        TX1=>TX1,
        TX2=>TX2,
        TX3=>TX3,
        FCLK=>FCLK,
        PCLK=>PCLK,
        RESET=>RESET
    );
```

## 1.2.12 OSER10

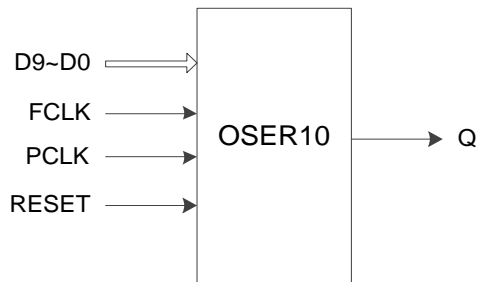
### 原语介绍

OSER10(10 to 1 Serializer)为 10 位并行输入、1 位串行输出的串化器。

支持器件：GW1N-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、GW1NR-9、GW1NZ-1、GW1NSR-2、GW1NSR-2C、GW2A-18、GW2AR-18、GW2A-55。

### 结构框图

图 1-24 OSER10 结构框图



### Port 介绍

表 1-24 Port 介绍

Port name	I/O	Description
D9~D0	Input	Data Input
FCLK	Input	Fast Clock Input
PCLK	Input	Primary Clock Input
RESET	Input	Asynchronous Reset Input
Q	Output	Data Output, to port I of output buffer or port DI of IODELAY

PCLK 通常由 FCLK 分频而得来， $f_{PCLK} = 1/5 f_{FCLK}$ 。

### 原语例化

#### Verilog 例化:

```
OSER10 uut(
    .Q(Q),
    .D0(D0),
    .D1(D1),
    .D2(D2),
    .D3(D3),
    .D4(D4),
    .D5(D5),
    .D6(D6),
    .D7(D7),
    .D8(D8),
    .D9(D9),
    .PCLK(PCLK),
```

```

        .FCLK(FCLK),
        .RESET(RESET)
    );
    defparam uut. GSREN="false";
    defparam uut. LSREN="true";
Vhdl 例化:
    COMPONENT OSER10
        GENERIC (GSREN:string:="false";
                LSREN:string:="true"
        );
        PORT(
            Q:OUT std_logic;
            D0:IN std_logic;
            D1:IN std_logic;
            D2:IN std_logic;
            D3:IN std_logic;
            D4:IN std_logic;
            D5:IN std_logic;
            D6:IN std_logic;
            D7:IN std_logic;
            D8:IN std_logic;
            D9:IN std_logic;
            FCLK:IN std_logic;
            PCLK:IN std_logic;
            RESET:IN std_logic
        );
    END COMPONENT;
    uut:OSER10
        GENERIC MAP (GSREN=>"false",
                    LSREN=>"true"
        )
        PORT MAP (
            Q=>Q,
            D0=>D0,
            D1=>D1,
            D2=>D2,
            D3=>D3,
            D4=>D4,
            D5=>D5,
            D6=>D6,
            D7=>D7,
            D8=>D8,
            D9=>D9,
            FCLK=>FCLK,
            PCLK=>PCLK,
            RESET=>RESET
        );

```

## 1.2.13 OVIDEO

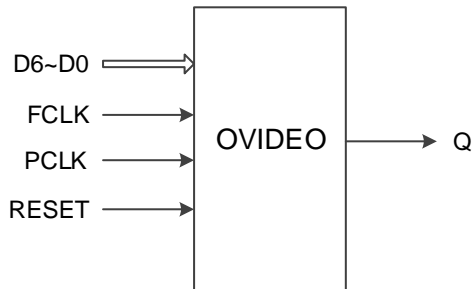
### 原语介绍

OVIDEO(7 to 1 Serializer)为 7 位并行输入、1 位串行输出的串化器。

支持器件: GW1N-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、GW1NR-9、GW1NZ-1、GW1NSR-2、GW1NSR-2C、GW2A-18、GW2AR-18、GW2A-55。

### 结构框图

图 1-25 OVIDEO 结构框图



### Port 介绍

表 1-25 Port 介绍

Port name	I/O	Description
D6~D0	Input	Data Input
FCLK	Input	Fast Clock Input
PCLK	Input	Primary Clock Input
RESET	Input	Asynchronous Reset Input
Q	Output	Data Output, to port I of output buffer or port DI of IODELAY

PCLK 通常由 FCLK 分频而得来,  $f_{PCLK} = 1/3.5 f_{FCLK}$ 。

### 原语例化

#### Verilog 例化:

```
OVIDEO uut(
    .Q(Q),
    .D0(D0),
    .D1(D1),
    .D2(D2),
    .D3(D3),
    .D4(D4),
    .D5(D5),
    .D6(D6),
    .PCLK(PCLK),
    .FCLK(FCLK),
    .RESET(RESET)
);
```

```

defparam uut. GSREN="false";
defparam uut. LSREN ="true";
Vhdl 例化:
COMPONENT OVIDEO
  GENERIC (GSREN:string:="false";
           LSREN:string:="true"
  );
  PORT(
    Q:OUT std_logic;
    D0:IN std_logic;
    D1:IN std_logic;
    D2:IN std_logic;
    D3:IN std_logic;
    D4:IN std_logic;
    D5:IN std_logic;
    D6:IN std_logic;
    FCLK:IN std_logic;
    PCLK:IN std_logic;
    RESET:IN std_logic
  );
END COMPONENT;
uut:OVIDEO
  GENERIC MAP (GSREN=>"false",
              LSREN=>"true"
  )
  PORT MAP (
    Q=>Q,
    D0=>D0,
    D1=>D1,
    D2=>D2,
    D3=>D3,
    D4=>D4,
    D5=>D5,
    D6=>D6,
    FCLK=>FCLK,
    PCLK=>PCLK,
    RESET=>RESET
  );

```

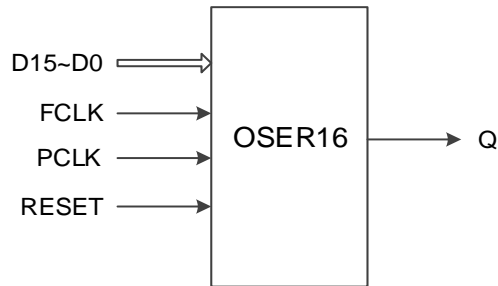
## 1.2.14 OSER16

### 原语介绍

OSER16(16 to 1 Serializer)为 16 位并行输入、1 位串行输出的串化器。  
 支持器件: GW1NS-2、GW1NS-2C、GW1N-6、GW1N-9、GW1NR-9、  
 GW1NSR-2、GW1NSR-2C。

## 结构框图

图 1-26 OSER16 结构框图



## Port 介绍

表 1-26 Port 介绍

Port name	I/O	Description
D15~D0	Input	Data Input
FCLK	Input	Fast Clock Input
PCLK	Input	Primary Clock Input
RESET	Input	Asynchronous Reset Input
Q	Output	Data Output, to port I of output buffer or port DI of IODELAY

PCLK 通常由 FCLK 分频而得来， $f_{PCLK} = 1/8 f_{FCLK}$ 。

## 原语例化

### Verilog 例化:

```
OSER16 uut(
    .Q(Q),
    .D0(D0),
    .D1(D1),
    .D2(D2),
    .D3(D3),
    .D4(D4),
    .D5(D5),
    .D6(D6),
    .D7(D7),
    .D8(D8),
    .D9(D9),
    .D10(D10),
    .D11(D11),
    .D12(D12),
    .D13(D13),
    .D14(D14),
    .D15(D15),
    .PCLK(PCLK),
    .FCLK(FCLK),
    .RESET(RESET)
);
```

```

defparam uut. GSREN="false";
defparam uut. LSREN ="true";
Vhdl 例化:
COMPONENT OSER16
    GENERIC (GSREN:string:="false";
            LSREN:string:="true"
    );
    PORT(
        Q:OUT std_logic;
        D0:IN std_logic;
        D1:IN std_logic;
        D2:IN std_logic;
        D3:IN std_logic;
        D4:IN std_logic;
        D5:IN std_logic;
        D6:IN std_logic;
        D7:IN std_logic;
        D8:IN std_logic;
        D9:IN std_logic;
        D10:IN std_logic;
        D11:IN std_logic;
        D12:IN std_logic;
        D13:IN std_logic;
        D14:IN std_logic;
        D15:IN std_logic;
        FCLK:IN std_logic;
        PCLK:IN std_logic;
        RESET:IN std_logic
    );
END COMPONENT;
uut:OSER16
    GENERIC MAP (GSREN=>"false",
                LSREN=>"true"
    )
    PORT MAP (
        Q=>Q,
        D0=>D0,
        D1=>D1,
        D2=>D2,
        D3=>D3,
        D4=>D4,
        D5=>D5,
        D6=>D6,
        D7=>D7,
        D8=>D8,
        D9=>D9,
        D10=>D10,
        D11=>D11,
        D12=>D12,
        D13=>D13,

```



```

D14=>D14,
D15=>D15,
FCLK=>FCLK,
PCLK=>PCLK,
RESET=>RESET
);

```

## 1.2.15 IDDR\_MEM

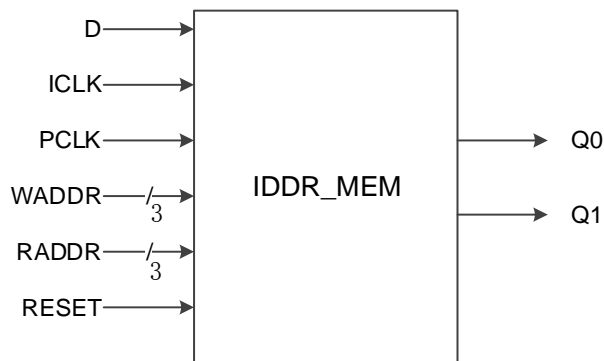
### 原语介绍

IDDR\_MEM(Dual Data Rate Input with Memory), 实现带 memory 的双倍数据速率输入。IDDR\_MEM 需要配合 DQS 使用, 其中, ICLK 连接 DQS 的输出信号 DQSR90, 且根据 ICLK 的时钟沿将数据送入 IDDR\_MEM; WADDR[2:0]连接 DQS 的输出信号 WPOINT; RADDR[2:0]连接 DQS 的输出信号 RPOINT。

支持器件: GW2A-18、GW2AR-18、GW2A-55。

### 结构框图

图 1-27 IDDR\_MEM 结构框图



### Port 介绍

表 1-27 Port 介绍

Port Name	I/O	Description
D	Input	Data Input, from port O of input buffer or port DO of IODELAY
ICLK	Input	Clock Input, from DQSR90 port of DQS
PCLK	Input	Primary Clock Input
WADDR[2:0]	Input	Write Address, from port WPOINT of DQS
RADDR[2:0]	Input	Read Address, from port RPOINT of DQS
RESET	Input	Asynchronous Reset Input
Q1~Q0	Output	Data Output

PCLK 和 ICLK 的频率关系为:  $f_{PCLK} = f_{ICLK}$ 。

PCLK 和 ICLK 之间存在一定的相位关系, 可根据 DQS 的 DLLSTEP 值确定相位关系。

## 原语例化

### Verilog 例化:

```
IDDR_MEM iddr_mem_inst(
    .Q0(q0),
    .Q1(q1),
    .D(d),
    .ICLK (iclk),
    .PCLK(pclk),
    .WADDR(waddr[2:0]),
    .RADDR(raddr[2:0]),
    .RESET(reset)
);
```

```
defparam uut. GSREN="false";
```

```
defparam uut. LSREN ="true";
```

### Vhdl 例化:

```
COMPONENT IDDR_MEM
    GENERIC (GSREN:string:="false";
            LSREN:string:="true"
    );
    PORT(
        Q0:OUT std_logic;
        Q1:OUT std_logic;
        D:IN std_logic;
        ICLK:IN std_logic;
        PCLK:IN std_logic;
        WADDR:IN std_logic_vector(2 downto 0);
        RADDR:IN std_logic_vector(2 downto 0);
        RESET:IN std_logic
    );
END COMPONENT;
uut:IDDR_MEM
    GENERIC MAP (GSREN=>"false",
                LSREN=>"true"
    )
    PORT MAP (
        Q0=>q0,
        Q1=>q1,
        D=>d,
        ICLK=>iclk,
        PCLK=>pclk,
        WADDR=>waddr,
        RADDR=>raddr,
        RESET=>reset
    );
```

## 1.2.16 ODDR\_MEM

### 原语介绍

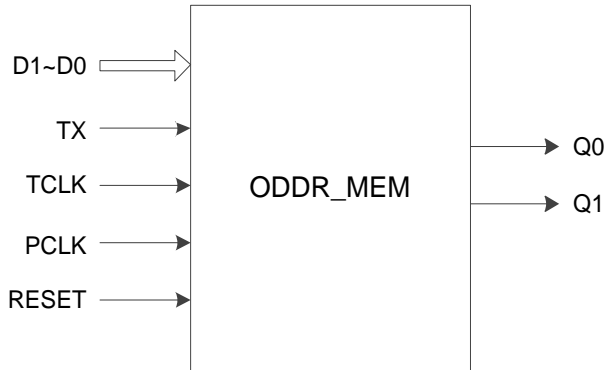
ODDR\_MEM(Dual Data Rate Output with Memory), 实现带 memory

的双倍数据速率输出。与 ODDR 不同，ODDR\_MEM 需要配合 DQS 使用，TCLK 连接 DQS 的输出信号 DQSW0 或 DQSW270，且根据 TCLK 的时钟沿将数据从 ODDR\_MEM 输出。

支持器件：GW2A-18、GW2AR-18、GW2A-55。

### 结构框图

图 1-28 ODDR\_MEM 结构框图



### Port 介绍

表 1-28 Port 介绍

Port Name	I/O	Description
D1~D0	Input	Data Input
TX	Input	Data Input
TCLK	Input	Clock Input, from port DQSW0 or DQSW270 of DQS
PCLK	Input	Primary Clock Input
RESET	Input	Asynchronous Reset Input
Q0	Output	Data Output, to port I of output buffer or port DI of IODELAY
Q1	Output	Tristate Enable Output, to port OEN of tristate/inout buffer(Q0 connected)or dangling

PCLK 和 TCLK 的频率关系为： $f_{PCLK} = f_{TCLK}$ 。

PCLK 和 TCLK 之间存在一定的相位关系，可根据 DQS 的 DLLSTEP 值和 WSTEP 值确定该相位关系。

### 原语例化

#### Verilog 例化:

```
ODDR_MEM oddr_mem_inst(
    .Q0(q0),
    .Q1(q1),
    .D0(d0),
    .D1(d1),
    .TX(tx),
    .TCLK(tclk),
    .PCLK(pclk),
```

```

        .RESET(reset)
    );
    defparam uut. GSREN="false";
    defparam uut. LSREN="true";
    defparam uut. TCLK_SOURCE="DQSW";
    defparam uut. TXCLK_POL=1'b0;
Vhdl 例化:
    COMPONENT ODDR_MEM
        GENERIC (GSREN:string="false";
                LSREN:string="true";
                TXCLK_POL:bit='0';
                TCLK_SOURCE:string="DQSW"
        );
        PORT(
            Q0:OUT std_logic;
            Q1:OUT std_logic;
            D0:IN std_logic;
            D1:IN std_logic;
            TX:IN std_logic;
            TCLK:IN std_logic;
            PCLK:IN std_logic;
            RESET:IN std_logic
        );
    END COMPONENT;
    uut:ODDR_MEM
        GENERIC MAP (GSREN=>"false",
                    LSREN=>"true",
                    TXCLK_POL=>'0',
                    TCLK_SOURCE=>"DQSW"
        )
        PORT MAP (
            Q0=>q0,
            Q1=>q1,
            D0=>d0,
            D1=>d1,
            TX=>tx,
            TCLK=>tclk,
            PCLK=>pclk,
            RESET=>reset
        );

```

## 1.2.17 IDES4\_MEM

### 原语介绍

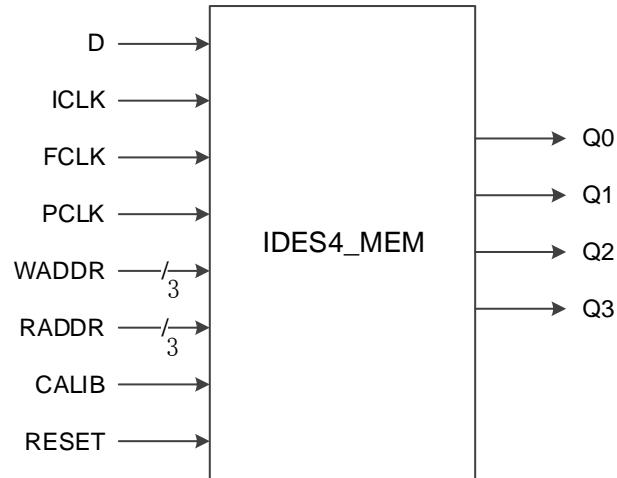
IDES4\_MEM(4 to 1 Deserializer with Memory) 带存储功能的 1:4 串并转换器，可实现 1 位串行转 4 位并行。与 IDES4 不同，IDES4\_MEM 需要配合 DQS 使用，其中，ICLK 连接 DQS 的输出信号 DQSR90，且根据 ICLK 的时钟沿将数据送入 IDES4\_MEM；WADDR[2:0]连接 DQS 的输出信号

WPOINT; RADDR[2:0]连接 DQS 的输出信号 RPOINT。

支持器件：GW2A-18、GW2AR-18、GW2A-55。

### 结构框图

图 1-29 IDES4\_MEM 结构框图



### Port 介绍

表 1-29 Port 介绍

Port Name	I/O	Description
D	Input	Data Input, from port O of input buffer or port DO of IODELAY
ICLK	Input	Clock Input , from DQSR90 port of DQS
FCLK	Input	Fast Clock Input
PCLK	Input	Primary Clock Input
WADDR[2:0]	Input	Write Address, from port WPOINT of DQS
RADDR[2:0]	Input	Read Address, from port RPOINT of DQS
CALIB	Input	Calib Signal Input
RESET	Input	Asynchronous Reset Input
Q3~Q0	Output	Data Output

PCLK、FCLK 和 ICLK 的频率关系为： $f_{PCLK} = 1/2 f_{FCLK} = 1/2 f_{ICLK}$ 。

FCLK 和 ICLK 之间存在一定的相位关系，可根据 DQS 的 DLLSTEP 值确定相位关系。

### 原语例化

#### Verilog 例化:

```

IDES4_MEM ides4_mem_inst(
    .Q0(q0),
    .Q1(q1),
    .Q2(q2),
    .Q3(q3),
    .D(d),
    .ICLK(iclk),

```

```

        .FCLK(fclk),
        .PCLK(pclk),
        .WADDR(waddr[2:0]),
        .RADDR(raddr[2:0]),
        .CALIB(calib),
        .RESET(reset)
    );
    defparam uut. GSREN="false";
    defparam uut. LSREN ="true";
Vhdl 例化:
    COMPONENT IDES4_MEM
        GENERIC (GSREN:string:="false";
                LSREN:string:="true"
        );
        PORT(
            Q0:OUT std_logic;
            Q1:OUT std_logic;
            Q2:OUT std_logic;
            Q3:OUT std_logic;
            D:IN std_logic;
            ICLK:IN std_logic;
            FCLK:IN std_logic;
            PCLK:IN std_logic;
            WADDR:IN std_logic_vector(2 downto 0);
            RADDR:IN std_logic_vector(2 downto 0);
            CALIB:IN std_logic;
            RESET:IN std_logic
        );
    END COMPONENT;
    uut:IDES4_MEM
        GENERIC MAP (GSREN=>"false",
                    LSREN=>"true"
        )
        PORT MAP (
            Q0=>q0,
            Q1=>q1,
            Q2=>q2,
            Q3=>q3,
            D=>d,
            ICLK=>iclk,
            FCLK=>fclk,
            PCLK=>pclk,
            WADDR=>waddr,
            RADDR=>raddr,
            CALIB=>calib,
            RESET=>reset
        );

```

## 1.2.18 OSER4\_MEM

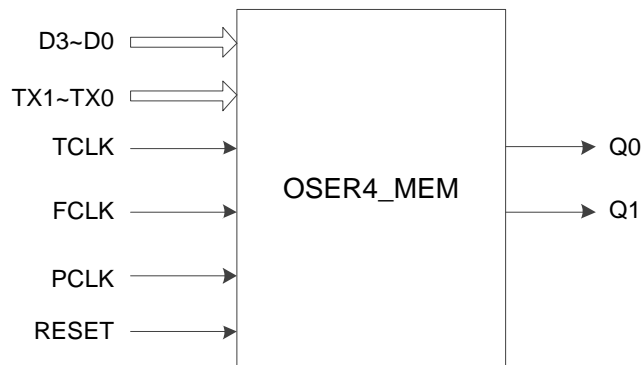
### 原语介绍

OSER4\_MEM(4 to 1 Serializer with Memory) 带存储功能的 4:1 并串转换器，可实现 4 位并行转 1 位串行。与 OSER4 不同，OSER4\_MEM 需要配合 DQS 使用，TCLK 连接 DQS 的输出信号 DQSW0 或 DQSW270，且根据 TCLK 的时钟沿将数据从 OSER4\_MEM 输出。

支持器件：GW2A-18、GW2AR-18、GW2A-55。

### 结构框图

图 1-30 OSER4\_MEM 结构框图



### Port 介绍

表 1-30 Port 介绍

Port Name	I/O	Description
D3~D0	Input	Data Input
TX1~TX0	Input	Data Input
TCLK	Input	Clock Input, from port DQSW0 or DQSW270 of DQS
FCLK	Input	Fast Clock Input
PCLK	Input	Primary Clock Input
RESET	Input	Asynchronous Reset Input
Q0	Output	Data Output, to port I of output buffer or port DI of IODELAY
Q1	Output	Tristate Enable Output, to port OEN of tristate/inout buffer(Q0 connected)or dangling

PCLK、FCLK 和 TCLK 的频率关系为： $f_{PCLK} = 1/2 f_{FCLK} = 1/2 f_{TCLK}$ 。

FCLK 和 TCLK 之间存在一定的相位关系，可根据 DQS 的 DLLSTEP 值和 WSTEP 值确定该相位关系。

### 原语例化

#### Verilog 例化：

```
OSER4_MEM oser4_mem_inst(
    .Q0(q0),
    .Q1(q1),
```

```

        .D0(d0),
        .D1(d1),
        .D2(d2),
        .D3(d3),
        .TX0(tx0),
        .TX1(tx1),
        .TCLK(tclk),
        .FCLK(fclk),
        .PCLK(pclk),
        .RESET(reset)
    );
    defparam uut. GSREN="false";
    defparam uut. LSREN="true";
    defparam uut. HWL="false";
    defparam uut. TCLK_SOURCE="DQSW";
    defparam uut. TXCLK_POL=1'b0;
Vhdl 例化:
    COMPONENT OSER4_MEM
        GENERIC (GSREN:string="false";
                LSREN:string="true";
                HWL:string="false";
                TXCLK_POL:bit='0';
                TCLK_SOURCE:string="DQSW"
        );
        PORT(
            Q0:OUT std_logic;
            Q1:OUT std_logic;
            D0:IN std_logic;
            D1:IN std_logic;
            D2:IN std_logic;
            D3:IN std_logic;
            TX0:IN std_logic;
            TX1:IN std_logic;
            TCLK:IN std_logic;
            FCLK:IN std_logic;
            PCLK:IN std_logic;
            RESET:IN std_logic
        );
    END COMPONENT;
    uut:OSER4_MEM
        GENERIC MAP (GSREN=>"false",
                    LSREN=>"true",
                    HWL=>"false",
                    TXCLK_POL=>'0',
                    TCLK_SOURCE=>"DQSW"
        )
        PORT MAP (
            Q0=>q0,
            Q1=>q1,
            D0=>d0,

```



```

D1=>d1,
D2=>d2,
D3=>d3,
TX0=>tx0,
TX1=>tx1,
TCLK=>tclk,
FCLK=>fclk,
PCLK=>pclk,
RESET=>reset
);

```

## 1.2.19 IDES8\_MEM

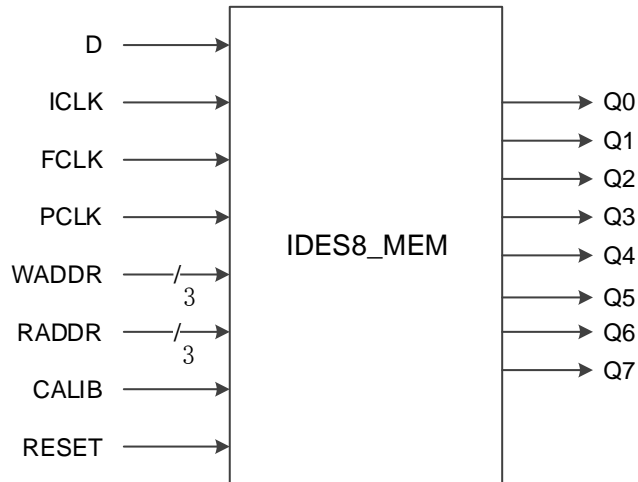
### 原语介绍

IDES8\_MEM(8 to 1 Deserializer with Memory) 带存储功能的 1:8 串并转换器，可实现 1 位串行转 8 位并行。与 IDES8 不同，IDES8\_MEM 需要配合 DQS 使用，其中，ICLK 连接 DQS 的输出信号 DQSR90，且根据 ICLK 的时钟沿将数据送入 IDES8\_MEM；WADDR[2:0]连接 DQS 的输出信号 WPOINT；RADDR[2:0]连接 DQS 的输出信号 RPOINT。

支持器件：GW2A-18、GW2AR-18、GW2A-55。

### 结构框图

图 1-31 IDES8\_MEM 结构框图



## Port 介绍

表 1-31 Port 介绍

Port Name	I/O	Description
D	Input	Data Input, from port O of input buffer or port DO of IODELAY
ICLK	Input	Clock Input , from DQSR90 port of DQS
FCLK	Input	Fast Clock Input
PCLK	Input	Primary Clock Input
WADDR[2:0]	Input	Write Address, from port WPOINT of DQS
RADDR[2:0]	Input	Read Address, from port RPOINT of DQS
CALIB	Input	Calib Signal Input
RESET	Input	Asynchronous Reset Input
Q7~Q0	Output	Data Output

PCLK、FCLK 和 ICLK 的频率关系为： $f_{PCLK} = 1/4 f_{FCLK} = 1/4 f_{ICLK}$ 。

FCLK 和 ICLK 之间存在一定的相位关系，可根据 DQS 的 DLLSTEP 值确定相位关系。

### 原语例化

#### Verilog 例化:

```

IDES8_MEM ides8_mem_inst(
    .Q0(q0),
    .Q1(q1),
    .Q2(q2),
    .Q3(q3),
    .Q4(q4),
    .Q5(q5),
    .Q6(q6),
    .Q7(q7),
    .D(d),
    .ICLK(iclk),
    .FCLK(fclk),
    .PCLK(pclk),
    .WADDR(waddr[2:0]),
    .RADDR(raddr[2:0]),
    .CALIB(calib),
    .RESET(reset)
);
defparam uut. GSREN="false";
defparam uut. LSREN ="true";

```

#### Vhdl 例化:

```

COMPONENT IDES8_MEM
    GENERIC (GSREN:string:="false";
            LSREN:string:="true"
    );
PORT(

```

```

        Q0:OUT std_logic;
        Q1:OUT std_logic;
        Q2:OUT std_logic;
        Q3:OUT std_logic;
        Q4:OUT std_logic;
        Q5:OUT std_logic;
        Q6:OUT std_logic;
        Q7:OUT std_logic;
        D:IN std_logic;
        ICLK:IN std_logic;
        FCLK:IN std_logic;
        PCLK:IN std_logic;
        WADDR:IN std_logic_vector(2 downto 0);
        RADDR:IN std_logic_vector(2 downto 0);
        CALIB:IN std_logic;
        RESET:IN std_logic
    );
END COMPONENT;
 uut:IDES8_MEM
    GENERIC MAP (GSREN=>"false",
                LSREN=>"true"
    )
    PORT MAP (
        Q0=>q0,
        Q1=>q1,
        Q2=>q2,
        Q3=>q3,
        Q4=>q4,
        Q5=>q5,
        Q6=>q6,
        Q7=>q7,
        D=>d,
        ICLK=>iclk,
        FCLK=>fclk,
        PCLK=>pclk,
        WADDR=>waddr,
        RADDR=>raddr,
        CALIB=>calib,
        RESET=>reset
    );

```

## 1.2.20 OSER8\_MEM

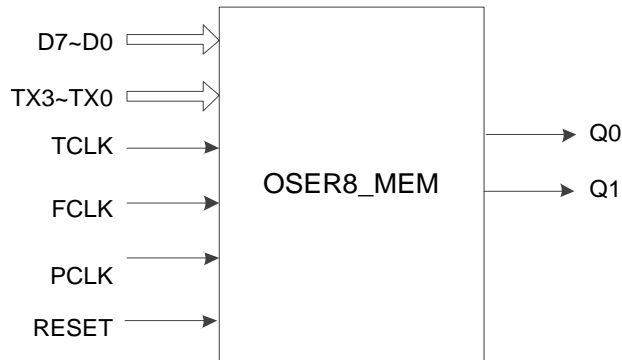
### 原语介绍

OSER8\_MEM(8 to 1 Serializer with Memory) 带存储功能的 8:1 并串转换器，可实现 8 位并行转 1 位串行。与 OSER8 不同，OSER8\_MEM 需要配合 DQS 使用，TCLK 连接 DQS 的输出信号 DQSW0 或 DQSW270，且根据 TCLK 的时钟沿将数据从 OSER8\_MEM 输出。

支持器件：GW2A-18、GW2AR-18、GW2A-55。

## 结构框图

图 1-32 OSER8\_MEM 结构框图



## Port 介绍

表 1-32 Port 介绍

Port Name	I/O	Description
D7~D0	Input	Data Input
TX3~TX0	Input	Data Input
TCLK	Input	Clock Input, from port DQSW0 or DQSW270 of DQS
FCLK	Input	Fast Clock Input
PCLK	Input	Primary Clock Input
RESET	Input	Asynchronous Reset Input
Q0	Output	Data Output
Q1	Output	Data Output

PCLK、FCLK 和 TCLK 的频率关系为： $f_{PCLK} = 1/4 f_{FCLK} = 1/4 f_{TCLK}$ 。

FCLK 和 TCLK 之间存在一定的相位关系，可根据 DQS 的 DLLSTEP 值和 WSTEP 值确定相位关系。

## 原语例化

### Verilog 例化:

```
OSER8_MEM oser8_mem_inst(
    .Q0(q0),
    .Q1(q1),
    .D0(d0),
    .D1(d1),
    .D2(d2),
    .D3(d3),
    .D4 (d4),
    .D5 (d5),
    .D6 (d6),
    .D7 (d7),
    .TX0 (tx0),
    .TX1 (tx1),
```

```

        .TX2 (tx2),
        .TX3 (tx3),
        .TCLK (tclk),
        .FCLK (fclk),
        .PCLK (pclk),
        .RESET(reset)
    );
    defparam uut. GSREN="false";
    defparam uut. LSREN = "true";
    defparam uut. HWL = "false";
    defparam uut. TCLK_SOURCE = "DQSW";
    defparam uut. TXCLK_POL=1'b0;
Vhdl 例化:
    COMPONENT OSER8_MEM
        GENERIC (GSREN:string:="false";
                LSREN:string:="true";
                HWL:string:="false";
                TXCLK_POL:bit='0';
                TCLK_SOURCE:string:="DQSW"
        );
        PORT(
            Q0:OUT std_logic;
            Q1:OUT std_logic;
            D0:IN std_logic;
            D1:IN std_logic;
            D2:IN std_logic;
            D3:IN std_logic;
            D4:IN std_logic;
            D5:IN std_logic;
            D6:IN std_logic;
            D7:IN std_logic;
            TX0:IN std_logic;
            TX1:IN std_logic;
            TX2:IN std_logic;
            TX3:IN std_logic;
            TCLK:IN std_logic;
            FCLK:IN std_logic;
            PCLK:IN std_logic;
            RESET:IN std_logic
        );
    END COMPONENT;
    uut:OSER8_MEM
        GENERIC MAP (GSREN=>"false",
                    LSREN=>"true",
                    HWL=>"false",
                    TXCLK_POL=>'0',
                    TCLK_SOURCE=>"DQSW"
        )
        PORT MAP (
            Q0=>q0,

```

```

Q1=>q1,
D0=>d0,
D1=>d1,
D2=>d2,
D3=>d3,
D4=>d4,
D5=>d5,
D6=>d6,
D7=>d7,
TX0=>tx0,
TX1=>tx1,
TX2=>tx2,
TX3=>tx3,
TCLK=>tclk,
FCLK=>fclk,
PCLK=>pclk,
RESET=>reset
);

```

## 1.2.21 IODELAY

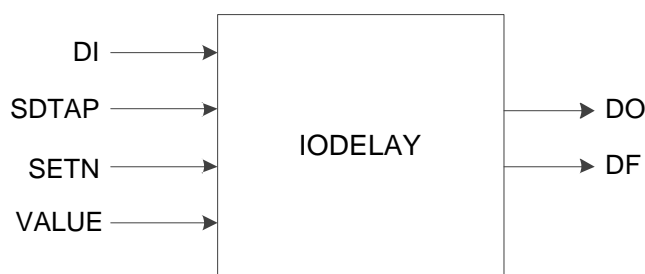
### 原语介绍

IODELAY(Input/Output delay)输入输出延时，是 IO 模块所包含的一个可编程绝对延时单元。

支持器件：GW1N-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、GW1NR-9、GW1NZ-1、GW1NSR-2、GW1NSR-2C、GW2A-18、GW2AR-18、GW2A-55。

### 结构框图

图 1-33 IODELAY 结构框图



## Port 介绍

表 1-33 Port 介绍

Port Name	I/O	Description
DI	Input	Data Input
SDTAP	Input	Control Delay Code's Download
SETN	Input	Direction can be Selected to Decide Delay, Increase or Decrease
VALUE	Input	Adjust Delay Value
DO	Output	Data Output
DF	Output	Margin test output flag for DELAY to indicate the under-flow or over-flow

## Attribute 介绍

表 1-34 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
C_STATIC_DLY	0~127	0	Delay Control

## 原语例化

### Verilog 例化:

```

IODELAY iodelay_inst(
    .DO(dout),
    .DF(df),
    .DI(di),
    .SDTAP(sdtap),
    .SETN(setn),
    .VALUE(value)
);
defparam iodelay_inst.C_STATIC_DLY=0;

```

### Vhdl 例化:

```

COMPONENT IODELAY
    GENERIC (C_STATIC_DLY:integer:=0
    );
    PORT(
        DO:OUT std_logic;
        DF:OUT std_logic;
        DI:IN std_logic;
        SDTAP:IN std_logic;
        SETN:IN std_logic;
        VALUE:IN std_logic
    );
END COMPONENT;
 uut:IODELAY
    GENERIC MAP (C_STATIC_DLY=>0
    )
    PORT MAP (
        DO=>dout,

```

```

DF=>df,
DI=>di,
SDTAP=>sdtap,
SETN=>setn,
VALUE=>value
);

```

## 1.2.22 IEM

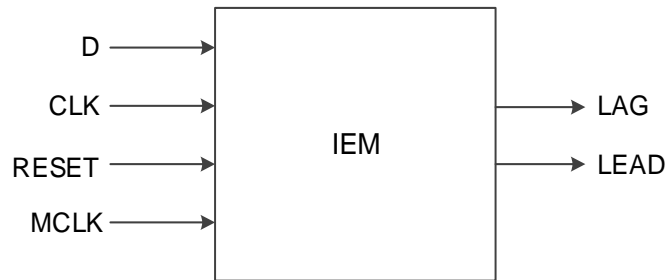
### 原语介绍

IEM(Input Edge Monitor)输入边沿监测，是 IO 模块所包含的一个取样模块，用来取样数据边沿，用于通用 DDR 模式。

支持器件：GW1N-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、GW1NR-9、GW1NZ-1、GW1NSR-2、GW1NSR-2C、GW2A-18、GW2AR-18、GW2A-55。

### 结构框图

图 1-34 IEM 结构框图



### Port 介绍

表 1-35 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLK	Input	Control Delay Code's Download
RESET	Input	Direction can be Selected to Decide Delay, Increase or Decrease
MCLK	Input	Adjust Delay Value
LAG	Output	Data Output
LEAD	Output	Margin test output flag for DELAY to indicate the under-flow or over-flow



## Attribute 介绍

表 1-36 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
WINSIZE	SMALL,MIDSMALL, MIDLARGE, LARGE	SMALL	Delay Control
GSREN	false, true	false	Global reset
LSREN	false, true	true	Enable reset

## 原语例化

### Verilog 例化:

```
IEM iem_inst(
    .LAG(lag),
    .LEAD(lead),
    .D(d),
    .CLK(clk),
    .MCLK(mclk),
    .RESET(reset)
);
defparam iodelay_inst.WINSIZE = "SMALL";
defparam iodelay_inst.GSREN = "false";
defparam iodelay_inst.LSREN = "true";
```

### Vhdl 例化:

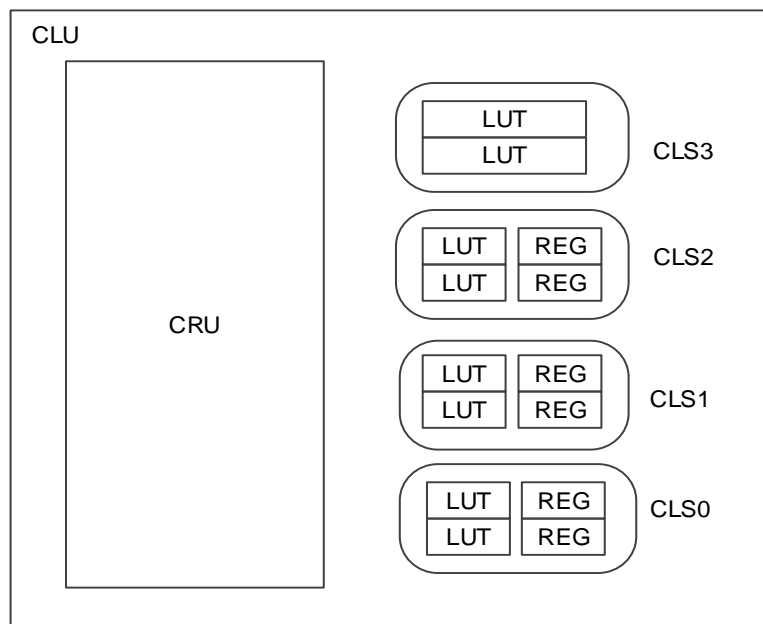
```
COMPONENT IEM
    GENERIC (WINSIZE:string:="SMALL";
            GSREN:string:="false";
            LSREN:string:="true"
    );
    PORT(
        LAG:OUT std_logic;
        LEAD:OUT std_logic;
        D:IN std_logic;
        CLK:IN std_logic;
        MCLK:IN std_logic;
        RESET:IN std_logic
    );
END COMPONENT;
uut:IEM
    GENERIC MAP (WINSIZE=>"SMALL",
                GSREN=>"false",
                LSREN=>"true"
    )
    PORT MAP (
        LAG=>lag,
        LEAD=>lead,
        D=>d,
        CLK=>clk,
        MCLK=>mclk,
```

```
RESET=>reset  
);
```

# 2<sub>CLU</sub>

可配置逻辑单元 CLU(Configurable Logic Unit)是构成 FPGA 产品的基本单元，每个 CLU 由四个可配置功能部分 CLS(Configurable Logic Section)和一个可配置绕线单元 CRU(Configurable Routing Unit)组成，CLU 的结构示意图如图 2-1 所示。其中可配置功能部分可配置查找表 LUT、2 输入算术逻辑单元 ALU 和寄存器 REG。CLU 模块可实现 MUX/LUT/ALU/FF/LATCH 等模块的功能。

图 2-1 CLU 结构示意图



## 2.1 LUT

输入查找表 LUT，常用的 LUT 结构有 LUT1、LUT2、LUT3、LUT4，其区别在于查找表输入位宽的不同。

支持器件：GW1N-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、

GW1NR-9、GW1NZ-1、GW1NSR-2、GW1NSR-2C、GW2A-18、GW2AR-18、GW2A-55。

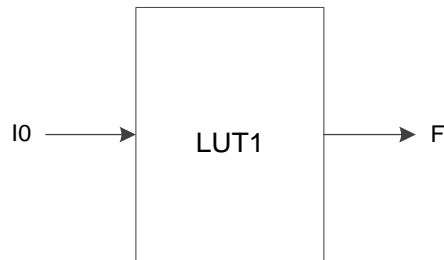
## 2.1.1 LUT1

### 原语介绍

LUT1(1-input Look-up Table)是其中最简单的一种，常用于实现缓冲器和反相器。LUT1 为 1 输入的查找表，通过 parameter 给 INIT 赋初值后，根据输入的地址查找对应的数据并输出结果。

### 结构框图

图 2-2 LUT1 结构框图



### Port 介绍

表 2-1 Port 介绍

Port Name	I/O	Description
I0	Input	Data Input
F	Output	Data Output

### Attribute 介绍

表 2-2 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	2'h0~2'h3	2'h0	Initial value for LUT1

### 真值表

表 2-3 真值表

Input(I0)	Output(F)
0	INIT[0]
1	INIT[1]

### 原语例化

#### Verilog 例化:

```

LUT1 instName (
    .I0(I0),
    .F(F)
);
defparam instName.INIT=2'h1;
  
```

#### Vhdl 例化:

```

COMPONENT LUT1
  GENERIC (INIT:bit_vector:=X"0");
  PORT(
    F:OUT std_logic;
    I0:IN std_logic
  );
END COMPONENT;
 uut:LUT1
  GENERIC MAP(INIT=>X"0")
  PORT MAP (
    F=>F,
    I0=>I0
  );

```

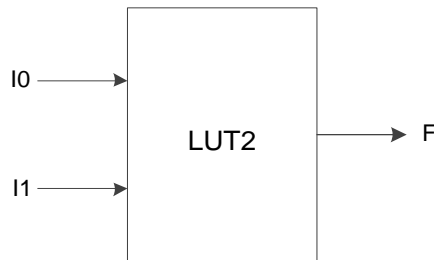
## 2.1.2 LUT2

### 原语介绍

LUT2(2-input Look-up Table)为 2 输入的查找表,通过 parameter 给 INIT 赋初值后, 根据输入的地址查找对应的数据并输出结果。

### 结构框图

图 2-3 LUT2 结构框图



### Port 介绍

表 2-4 Port 介绍

Port Name	I/O	Description
I0	Input	Data Input
I1	Input	Data Input
F	Output	Data Output

### Attribute 介绍

表 2-5 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	4'h0~4'hf	4'h0	Initial value for LUT2

## 真值表

表 2-6 真值表

Input(I1)	Input(I0)	Output(F)
0	0	INIT[0]
0	1	INIT[1]
1	0	INIT[2]
1	1	INIT[3]

## 原语例化

### Verilog 例化:

```
LUT2 instName (
    .I0(I0),
    .I1(I1),
    .F(F)
);
defparam instName.INIT=4'h1;
```

### Vhdl 例化:

```
COMPONENT LUT2
    GENERIC (INIT:bit_vector:=X"0");
    PORT(
        F:OUT std_logic;
        I0:IN std_logic;
        I1:IN std_logic
    );
END COMPONENT;
 uut:LUT2
    GENERIC MAP(INIT=>X"0")
    PORT MAP (
        F=>F,
        I0=>I0,
        I1=>I1
    );
```

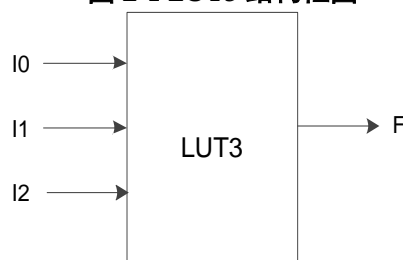
## 2.1.3 LUT3

### 原语介绍

LUT3(3-input Look-up Table)为 3 输入的查找表,通过 parameter 给 INIT 赋初值后,根据输入的地址查找对应的数据并输出结果。

### 结构框图

图 2-4 LUT3 结构框图



## Port 介绍

表 2-7 Port 介绍

Port Name	I/O	Description
I0	Input	Data Input
I1	Input	Data Input
I2	Input	Data Input
F	Output	Data Output

## Attribute 介绍

表 2-8 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	8'h00~8'hff	8'h00	Initial value for LUT3

## 真值表

表 2-9 真值表

Input(I2)	Input(I1)	Input(I0)	Output(F)
0	0	0	INIT[0]
0	0	1	INIT[1]
0	1	0	INIT[2]
0	1	1	INIT[3]
1	0	0	INIT[4]
1	0	1	INIT[5]
1	1	0	INIT[6]
1	1	1	INIT[7]

## 原语例化

### Verilog 例化:

```
LUT3 instName (
    .I0(I0),
    .I1(I1),
    .I2(I2),
    .F(F)
);
defparam instName.INIT=8'h10;
```

### Vhdl 例化

```
COMPONENT LUT3
    GENERIC (INIT:bit_vector:=X"00");
    PORT(
        F:OUT std_logic;
        I0:IN std_logic;
        I1:IN std_logic;
        I2:IN std_logic
```

```

);
END COMPONENT;
 uut:LUT3
   GENERIC MAP(INIT=>X"00")
   PORT MAP (
     F=>F,
     I0=>I0,
     I1=>I1,
     I2=>I2
   );

```

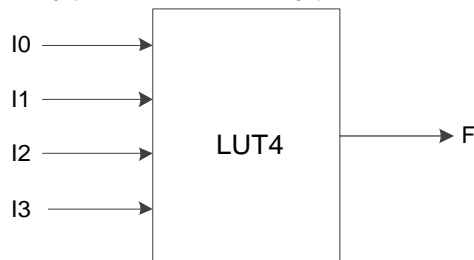
## 2.1.4 LUT4

### 原语介绍

LUT4(4-input Look-up Table)为 4 输入的查找表,通过 parameter 给 INIT 赋初值后, 根据输入的地址查找对应的数据并输出结果。

### 结构框图

图 2-5 LUT4 结构框图



### Port 介绍

表 2-10 Port 介绍

Port Name	I/O	Description
I0	Input	Data Input
I1	Input	Data Input
I2	Input	Data Input
I3	Input	Data Input
F	Output	Data Output

### Attribute 介绍

表 2-11 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	16'h0000~16'hffff	16'h0000	Initial value for LUT4



## 真值表

表 2-12 真值表

Input(I3)	Input(I2)	Input(I1)	Input(I0)	Output(F)
0	0	0	0	INIT[0]
0	0	0	1	INIT[1]
0	0	1	0	INIT[2]
0	0	1	1	INIT[3]
0	1	0	0	INIT[4]
0	1	0	1	INIT[5]
0	1	1	0	INIT[6]
0	1	1	1	INIT[7]
1	0	0	0	INIT[8]
1	0	0	1	INIT[9]
1	0	1	0	INIT[10]
1	0	1	1	INIT[11]
1	1	0	0	INIT[12]
1	1	0	1	INIT[13]
1	1	1	0	INIT[14]
1	1	1	1	INIT[15]

## 原语例化

### Verilog 例化:

```
LUT4 instName (
    .I0(I0),
    .I1(I1),
    .I2(I2),
    .I3(I3),
    .F(F)
);
defparam instName.INIT=16'h1011;
```

### Vhdl 例化:

```
COMPONENT LUT4
    GENERIC (INIT:bit_vector:=X"0000");
    PORT(
        F:OUT std_logic;
        I0:IN std_logic;
        I1:IN std_logic;
        I2:IN std_logic;
        I3:IN std_logic
    );
END COMPONENT;
 uut:LUT4
    GENERIC MAP(INIT=>X"0000")
    PORT MAP (
```

```

F=>F,
I0=>I0,
I1=>I1,
I2=>I2,
I3=>I3
);

```

## 2.1.5 Wide LUT

### 原语介绍

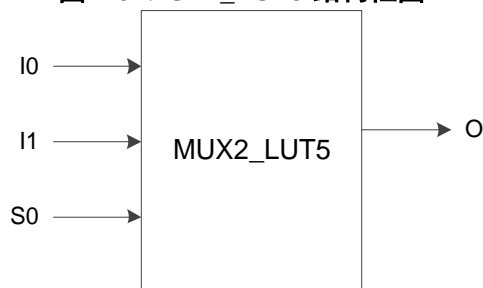
Wide LUT 是通过 LUT4 和 MUX2 构造高阶 LUT，高云 FPGA 目前支持的构造高阶 LUT 的 MUX2 有 MUX2\_LUT5/ MUX2\_LUT6/ MUX2\_LUT7/ MUX2\_LUT8。

高阶 LUT 的构造方式如下：两个 LUT4 和 MUX2\_LUT5 可组合实现 LUT5，两个组合实现的 LUT5 和 MUX2\_LUT6 可组合实现 LUT6，两个组合实现的 LUT6 和 MUX2\_LUT7 可组合实现 LUT7，两个组合实现的 LUT7 和 MUX2\_LUT8 可组合实现 LUT8。

以 MUX2\_LUT5 为例介绍 Wide LUT 的使用。

### 结构框图

图 2-6 MUX2\_LUT5 结构框图



### Port 介绍

表 2-13 Port 介绍

Port Name	I/O	Description
I0	Input	Data Input
I1	Input	Data Input
S0	Input	Select Signal Input
O	Output	Data Output

### 真值表

表 2-14 真值表

Input(S0)	Output(O)
0	I0
1	I1

## 原语例化

### Verilog 例化:

```

MUX2_LUT5 instName (
    .I0(f0),
    .I1(f1),
    .S0(i5),
    .O(o)
);
LUT4 lut_0 (
    .I0(i0),
    .I1(i1),
    .I2(i2),
    .I3(i3),
    .F(f0)
);
defparam lut_0.INIT=16'h184A;
LUT4 lut_1 (
    .I0(i0),
    .I1(i1),
    .I2(i2),
    .I3(i3),
    .F(f1)
);
defparam lut_1.INIT=16'h184A;

```

### Vhdl 例化:

```

COMPONENT MUX2_LUT5
    PORT(
        O:OUT std_logic;
        I0:IN std_logic;
        I1:IN std_logic;
        S0:IN std_logic
    );
END COMPONENT;
COMPONENT LUT4
    PORT(
        F:OUT std_logic;
        I0:IN std_logic;
        I1:IN std_logic;
        I2:IN std_logic;
        I3:IN std_logic
    );
END COMPONENT;
uut0: MUX2_LUT5
    PORT MAP (
        O=>o,
        I0=>f0,
        I1=>f1,
        S0=>i5
    );

```

```

 uut1:LUT4
   GENERIC MAP(INIT=>X"0000")
   PORT MAP (
     F=>f0,
     I0=>i0,
     I1=>i1,
     I2=>i2,
     I3=>i3
   );
 uut2:LUT4
   GENERIC MAP(INIT=>X"0000")
   PORT MAP (
     F=>f1,
     I0=>i0,
     I1=>i1,
     I2=>i2,
     I3=>i3
   );

```

## 2.2 MUX

MUX 是多路复用器，拥有多路输入，通过通道选择信号确定其中一路数据传送到输出端。高云原语中有 2 选 1 和 4 选 1 两种多路复用器。

支持器件：GW1N-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、GW1NR-9、GW1NZ-1、GW1NSR-2、GW1NSR-2C、GW2A-18、GW2AR-18、GW2A-55。

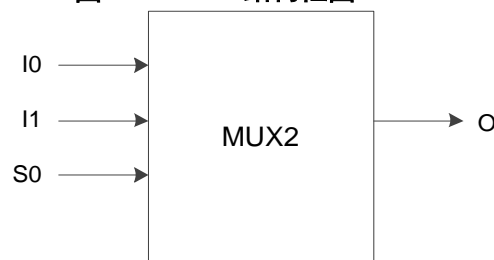
### 2.2.1 MUX2

#### 原语介绍

MUX2(2-to-1 Multiplexer)是 2 选 1 的复用器，根据选择信号，从两个输入中选择其中一个作为输出。

#### 结构框图

图 2-7 MUX2 结构框图



## Port 介绍

表 2-15 Port 介绍

Port Name	I/O	Description
I0	Input	Data Input
I1	Input	Data Input
S0	Input	Select Signal Input
O	Output	Data Output

## 真值表

表 2-16 真值表

Input(S0)	Output(O)
0	I0
1	I1

## 原语例化

### Verilog 例化:

```
MUX2 instName (
    .I0(I0),
    .I1(I1),
    .S0(S0),
    .O(O)
);
```

### Vhdl 例化:

```
COMPONENT MUX2
    PORT(
        O:OUT std_logic;
        I0:IN std_logic;
        I1:IN std_logic;
        S0:IN std_logic
    );
END COMPONENT;
 uut:MUX2
    PORT MAP (
        O=>O,
        I0=>I0,
        I1=>I1,
        S0=>S0
    );
```

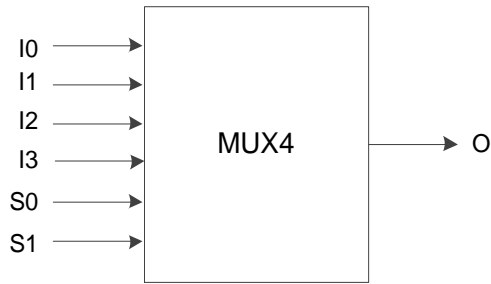
## 2.2.2 MUX4

### 原语介绍

MUX4(4-to-1 Multiplexer)是 4 选 1 的多路复用器，根据选择信号，从四个输入中选择其中一个作为输出。

## 结构框图

图 2-8 MUX4 结构框图



## Port 介绍

表 2-17 Port 介绍

Port Name	I/O	Description
I0	Input	Data Input
I1	Input	Data Input
I2	Input	Data Input
I3	Input	Data Input
S0	Input	Select Signal Input
S1	Input	Select Signal Input
O	Output	Data Output

## 真值表

表 2-18 真值表

Input(S1)	Input(S0)	Output(O)
0	0	I0
0	1	I1
1	0	I2
1	1	I3

## 原语例化

### Verilog 例化:

```
MUX4 instName (
    .I0(I0),
    .I1(I1),
    .I2(I2),
    .I3(I3),
    .S0(S0),
    .S1(S1),
    .O(O)
);
```

### Vhdl 例化:

```
COMPONENT MUX4
```

```

        PORT(
            O:OUT std_logic;
            I0:IN std_logic;
            I1:IN std_logic;
            I2:IN std_logic;
            I3:IN std_logic;
            S0:IN std_logic;
            S1:IN std_logic
        );
    END COMPONENT;
    uut:MUX4
        PORT MAP (
            O=>O,
            I0=>I0,
            I1=>I1,
            I2=>I2,
            I3=>I3,
            S0=>S0,
            S1=>S1
        );

```

## 2.2.3 Wide MUX

### 原语介绍

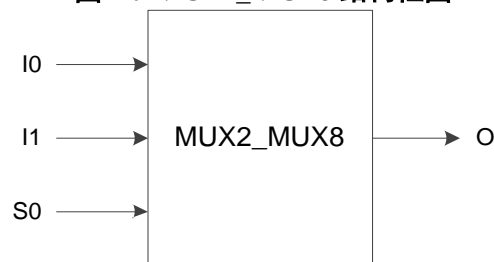
Wide MUX 是通过 MUX4 和 MUX2 构造高阶 MUX，高云 FPGA 目前支持的构造高阶 MUX 的 MUX2 有 MUX2\_MUX8/ MUX2\_MUX16/ MUX2\_MUX32。

高阶 MUX 的构造方式如下：两个 MUX4 和 MUX2\_MUX8 可组合实现 MUX8，两个组合实现的 MUX8 和 MUX2\_MUX16 可组合实现 MUX16，两个组合实现的 MUX16 和 MUX2\_MUX32 可组合实现 MUX32。

以 MUX2\_MUX8 为例介绍 Wide MUX 的使用。

### 结构框图

图 2-9 MUX2\_MUX8 结构框图



## Port 介绍

表 2-19 Port 介绍

Port Name	I/O	Description
I0	Input	Data Input
I1	Input	Data Input
S0	Input	Select Signal Input
O	Output	Data Output

## 真值表

表 2-20 真值表

Input(S0)	Output(O)
0	I0
1	I1

## 原语例化

### Verilog 例化:

```

MUX2_MUX8 instName (
    .I0(o0),
    .I1(o1),
    .S0(S2),
    .O(O)
);
MUX4 mux_0 (
    .I0(i0),
    .I1(i1),
    .I2(i2),
    .I3(i3),
    .S0(s0),
    .S1(s1),
    .O(o0)
);
MUX4 mux_1 (
    .I0(i4),
    .I1(i5),
    .I2(i6),
    .I3(i7),
    .S0(s0),
    .S1(s1),
    .O(o1)
);

```

### Vhdl 例化:

```

COMPONENT MUX2_MUX8
    PORT(
        O:OUT std_logic;
        I0:IN std_logic;

```



```

        I1:IN std_logic;
        S0:IN std_logic
    );
END COMPONENT;
COMPONENT MUX4
    PORT(
        O:OUT std_logic;
        I0:IN std_logic;
        I1:IN std_logic;
        I2:IN std_logic;
        I3:IN std_logic;
        S0:IN std_logic;
        S1:IN std_logic
    );
END COMPONENT;
uut1:MUX2_MUX8
    PORT MAP (
        O=>O,
        I0=>o0,
        I1=>o1,
        S0=>S2
    );
uut2:MUX4
    PORT MAP (
        O=>o0,
        I0=>I0,
        I1=>I1,
        I2=>I2,
        I3=>I3,
        S0=>S0,
        S1=>S1
    );
uut3:MUX4sss
    PORT MAP (
        O=>o1,
        I0=>I4,
        I1=>I5,
        I2=>I6,
        I3=>I7,
        S0=>S0,
        S1=>S1
    );

```

## 2.3 ALU

### 原语介绍

ALU(2-input Arithmetic Logic Unit)2 输入算术逻辑单元，实现了 ADD/SUB/ADDSUB 等功能。

支持器件：GW1N-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、

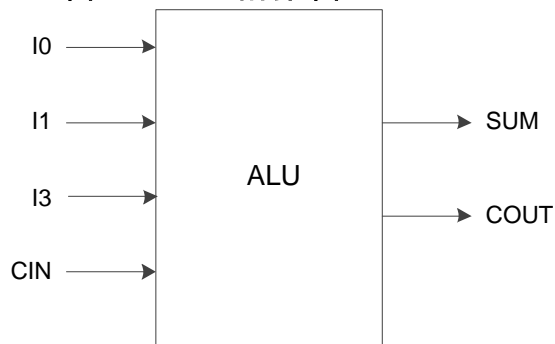
GW1NR-9、GW1NZ-1、GW1NSR-2、GW1NSR-2C、GW2A-18、GW2AR-18、GW2A-55。具体功能如表 2-21 所示。

表 2-21 ALU 功能

项目	描述
ADD	加法运算
SUB	减法运算
ADDSUB	加/减法运算
CUP	加计数器
CDN	减计数器
CUPCDN	加/减计数器
GE	大于比较器
NE	不等于比较器
LE	小于比较器
MULT	乘法器

### 结构框图

图 2-10 ALU 结构框图



### Port 介绍

表 2-22 Port 介绍

Port Name	Input/Output	Description
I0	Input	Data Input
I1	Input	Data Input
I3	Input	Data Input
CIN	Input	Carry Input
COUT	Output	Carry Output
SUM	Output	Data Output

## Attribute 介绍

表 2-23 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
ALU_MODE	0,1,2,3,4,5,6,7,8,9	0	Select the function of arithmetic. 0:ADD; 1:SUB; 2:ADDSUB; 3:NE; 4:GE; 5:LE; 6:CUP; 7:CDN; 8:CUPCDN; 9:MULT

## 原语例化

### Verilog 例化:

```

ALU instName (
    .I0(I0),
    .I1(I1),
    .I3(I3),
    .CIN(CIN),
    .COUT(COUT),
    .SUM(SUM)
);
defparam instName.ALU_MODE=1;

```

### Vhdl 例化:

```

COMPONENT ALU
    GENERIC (ALU_MODE:integer:=0);
    PORT(
        COUT:OUT std_logic;
        SUM:OUT std_logic;
        I0:IN std_logic;
        I1:IN std_logic;
        I3:IN std_logic;
        CIN:IN std_logic
    );
END COMPONENT;
 uut:ALU
    GENERIC MAP(ALU_MODE=>1)
    PORT MAP (
        COUT=>COUT,
        SUM=>SUM,
        I0=>I0,
        I1=>I1,
        I3=>I3,
        CIN=>CIN
    );

```

## 2.4 FF

触发器是时序电路中常用的基本元件，FPGA 内部的时序逻辑都可通过 FF 结构实现，常用的 FF 有 DFF、DFFE、DFFS、DFFSE 等，其区别在于复位方式、触发方式等方面。

支持器件：GW1N-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、GW1NR-9、GW1NZ-1、GW1NSR-2、GW1NSR-2C、GW2A-18、GW2AR-18、GW2A-55。与 FF 相关的原语有 20 个，如表 2-24 所示。

**表 2-24 与 FF 相关的原语**

原语	描述
DFF	D 触发器
DFFE	带时钟使能 D 触发器
DFFS	带同步置位 D 触发器
DFFSE	带时钟使能、同步置位 D 触发器
DFFR	带同步复位 D 触发器
DFFRE	带时钟使能、同步复位 D 触发器
DFFP	带异步置位 D 触发器
DFFPE	带时钟使能、异步置位 D 触发器
DFFC	带异步复位 D 触发器
DFFCE	带时钟使能、异步复位 D 触发器
DFFN	下降沿 D 触发器
DFFNE	下降沿带时钟使能 D 触发器
DFFNS	下降沿带同步置位 D 触发器
DFFNSE	下降沿带时钟使能、同步置位 D 触发器
DFFNR	下降沿带同步复位 D 触发器
DFFNRE	下降沿带时钟使能、同步复位 D 触发器
DFFNP	下降沿带异步置位 D 触发器
DFFNPE	下降沿带时钟使能、异步置位 D 触发器
DFFNC	下降沿带异步复位 D 触发器
DFFNCE	下降沿带时钟使能、异步复位 D 触发器

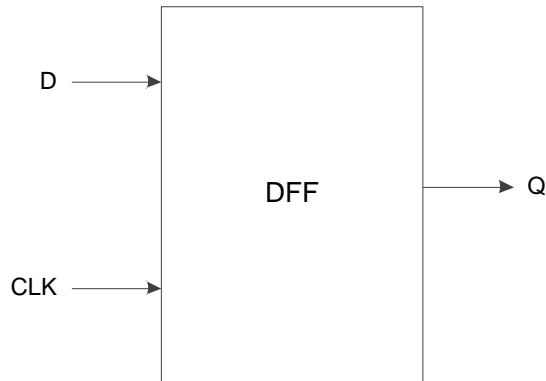
### 2.4.1 DFF

#### 原语介绍

DFF(D Flip-Flop)是最简单常用的一种触发器，常用于信号采样和处理，是上升沿触发的 D 触发器。

## 结构框图

图 2-11 DFF 结构框图



## Port 介绍

表 2-25 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLK	Input	Clock Input
Q	Output	Data Output

## Attribute 介绍

表 2-26 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b0	Initial value for DFF

## 原语例化

### Verilog 例化:

```
DFF instName (
    .D(D),
    .CLK(CLK),
    .Q(Q)
);
defparam instName.INIT=1'b0;
```

### Vhdl 例化:

```
COMPONENT DFF
    GENERIC (INIT:bit:= '0');
    PORT(
        Q:OUT std_logic;
        D:IN std_logic;
        CLK:IN std_logic
    );
END COMPONENT;
 uut:DFF
    GENERIC MAP(INIT=>'0')
    PORT MAP (
```

```

        Q=>Q,
        D=>D,
        CLK=>CLK
    );

```

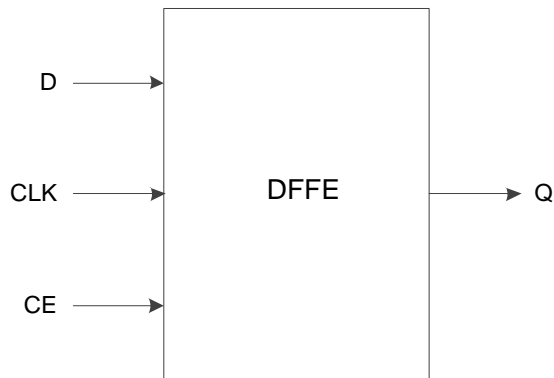
## 2.4.2 DFFE

### 原语介绍

DFFE(D Flip-Flop with Clock Enable)是上升沿触发的 D 触发器，具有时钟使能功能。

### 结构框图

图 2-12 DFFE 结构框图



### Port 介绍

表 2-27 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLK	Input	Clock Input
CE	Input	Clock Enable
Q	Output	Data Output

### Attribute 介绍

表 2-28 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b0	Initial value for DFFE

### 原语例化

#### Verilog 例化:

```

    DFFE instName (
        .D(D),
        .CLK(CLK),
        .CE(CE),
        .Q(Q)
    );
    defparam instName.INIT=1'b0;

```

**Vhdl 例化:**

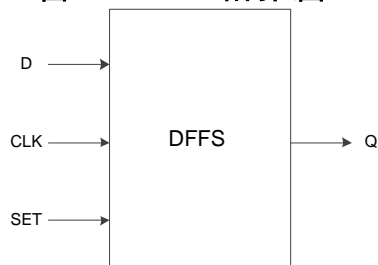
```

COMPONENT DFFE
  GENERIC (INIT:bit:='0');
  PORT(
    Q:OUT std_logic;
    D:IN std_logic;
    CLK:IN std_logic;
    CE:IN std_logic
  );
END COMPONENT;
 uut:DFFE
  GENERIC MAP(INIT=>'0')
  PORT MAP (
    Q=>Q,
    D=>D,
    CLK=>CLK,
    CE=>CE
  );

```

**2.4.3 DFFS****原语介绍**

DFFS(D Flip-Flop with Synchronous Set)是上升沿触发的 D 触发器，具有同步置位功能。

**结构框图****图 2-13 DFFS 结构框图****Port 介绍****表 2-29 Port 介绍**

Port Name	I/O	Description
D	Input	Data Input
CLK	Input	Clock Input
SET	Input	Synchronous Set Input
Q	Output	Data Output

## Attribute 介绍

表 2-30 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b1	Initial value for DFFS

## 原语例化

### Verilog 例化:

```

DFFS instName (
    .D(D),
    .CLK(CLK),
    .SET(SET),
    .Q(Q)
);
defparam instName.INIT=1'b1;

```

### Vhdl 例化:

```

COMPONENT DFFS
    GENERIC (INIT:bit:= '1');
    PORT(
        Q:OUT std_logic;
        D:IN std_logic;
        CLK:IN std_logic;
        SET:IN std_logic
    );
END COMPONENT;
 uut:DFFS
    GENERIC MAP(INIT=>'1')
    PORT MAP (
        Q=>Q,
        D=>D,
        CLK=>CLK,
        SET=>SET
    );

```

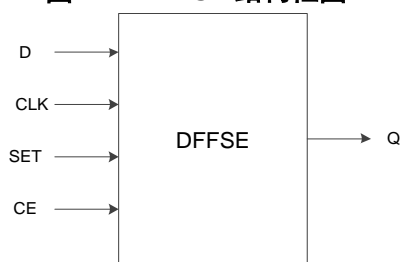
## 2.4.4 DFFSE

### 原语介绍

DFFSE(D Flip-Flop with Clock Enable and Synchronous Set)是上升沿触发的 D 触发器，具有同步置位和时钟使能功能。

### 结构框图

图 2-14 DFFSE 结构框图





## Port 介绍

表 2-31 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLK	Input	Clock Input
SET	Input	Synchronous Set Input
CE	Input	Clock Enable
Q	Output	Data Output

## Attribute 介绍

表 2-32 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b1	Initial value for DFFSE

## 原语例化

### Verilog 例化:

```
DFFSE instName (
    .D(D),
    .CLK(CLK),
    .SET(SET),
    .CE(CE),
    .Q(Q)
);
defparam instName.INIT=1'b1;
```

### Vhdl 例化:

```
COMPONENT DFFSE
    GENERIC (INIT:bit:= '1');
    PORT(
        Q:OUT std_logic;
        D:IN std_logic;
        CLK:IN std_logic;
        SET:IN std_logic;
        CE:IN std_logic
    );
END COMPONENT;
 uut:DFFSE
    GENERIC MAP(INIT=>'1')
    PORT MAP (
        Q=>Q,
        D=>D,
        CLK=>CLK,
        SET=>SET,
        CE=>CE
    );
```

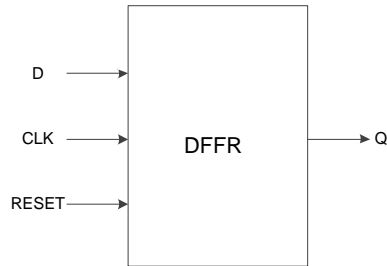
## 2.4.5 DFFR

### 原语介绍

DFFR(D Flip-Flop with Synchronous Reset)是上升沿触发的 D 触发器，具有同步复位功能。

### 结构框图

图 2-15 DFFR 结构框图



### Port 介绍

表 2-33 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLK	Input	Clock Input
RESET	Input	Synchronous Reset Input
Q	Output	Data Output

### Attribute 介绍

表 2-34 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b0	Initial value for DFFR

### 原语例化

#### Verilog 例化:

```

DFFR instName (
    .D(D),
    .CLK(CLK),
    .RESET(RESET),
    .Q(q)
);
defparam instName.INIT=1'b0;
  
```

#### Vhdl 例化:

```

COMPONENT DFFR
  GENERIC (INIT:bit:= '0');
  PORT(
    Q:OUT std_logic;
    D:IN std_logic;
  
```

```

        CLK:IN std_logic;
        RESET:IN std_logic
    );
END COMPONENT;
 uut:DFFR
    GENERIC MAP(INIT=>'0')
    PORT MAP (
        Q=>Q,
        D=>D,
        CLK=>CLK,
        RESET=>RESET
    );

```

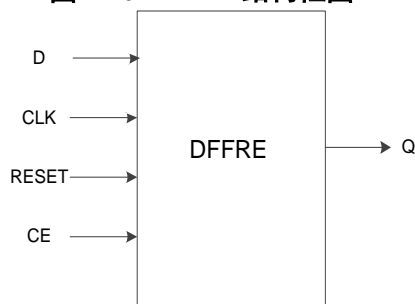
## 2.4.6 DFFRE

### 原语介绍

DFFRE(D Flip-Flop with Clock Enable and Synchronous Reset)是上升沿触发的 D 触发器，具有同步复位和时钟使能功能。

### 结构框图

图 2-16 DFFRE 结构框图



### Port 介绍

表 2-35 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLK	Input	Clock Input
RESET	Input	Synchronous Reset Input
CE	Input	Clock Enable
Q	Output	Data Output

### Attribute 介绍

表 2-36 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b0	Initial value for DFFRE

### 原语例化

**Verilog 例化:**

```

DFFRE instName (
    .D(D),
    .CLK(CLK),
    .RESET(RESET),
    .CE(CE),
    .Q(Q)
);
defparam instName.INIT=1'b0;
Vhdl 例化:
COMPONENT DFFRE
    GENERIC (INIT:bit:= '0');
    PORT(
        Q:OUT std_logic;
        D:IN std_logic;
        CLK:IN std_logic;
        RESET:IN std_logic;
        CE:IN std_logic
    );
END COMPONENT;
 uut:DFFRE
    GENERIC MAP(INIT=>'0')
    PORT MAP (
        Q=>Q,
        D=>D,
        CLK=>CLK,
        RESET=>RESET,
        CE=>CE
    );

```

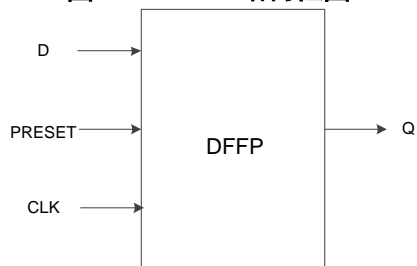
## 2.4.7 DFFP

### 原语介绍

DFFP(D Flip-Flop with Asynchronous Preset)是上升沿触发的 D 触发器，具有异步置位功能。

### 结构框图

图 2-17 DFFP 结构框图



## Port 介绍

表 2-37 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLK	Input	Clock Input
PRESET	Input	Asynchronous Preset Input
Q	Output	Data Output

## Attribute 介绍

表 2-38 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b1	Initial value for DFFP

## 原语例化

### Verilog 例化:

```
DFFP instName (
    .D(D),
    .CLK(CLK),
    .PRESET(PRESET),
    .Q(Q)
);
defparam instName.INIT=1'b1;
```

### Vhdl 例化:

```
COMPONENT DFFP
    GENERIC (INIT:bit:= '1');
    PORT(
        Q:OUT std_logic;
        D:IN std_logic;
        CLK:IN std_logic;
        PRESET:IN std_logic
    );
END COMPONENT;
 uut:DFFP
    GENERIC MAP(INIT=>'1')
    PORT MAP (
        Q=>Q,
        D=>D,
        CLK=>CLK,
        PRESET=>PRESET
    );
```

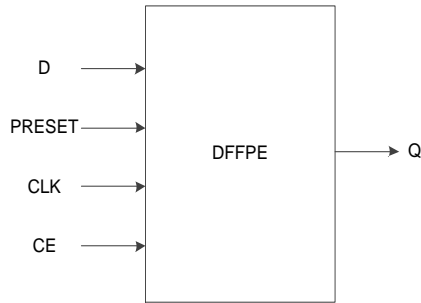
## 2.4.8 DFFPE

### 原语介绍

DFFPE(D Flip-Flop with Clock Enable and Asynchronous Preset)是上升沿触发的 D 触发器，具有异步置位和时钟使能功能。

## 结构框图

图 2-18 DFFPE 结构框图



## Port 介绍

表 2-39 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLK	Input	Clock Input
PRESET	Input	Asynchronous Preset Input
CE	Input	Clock Enable
Q	Output	Data Output

## Attribute 介绍

表 2-40 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b1	Initial value for DFFPE

## 原语例化

### Verilog 例化:

```
DFFPE instName (
    .D(D),
    .CLK(CLK),
    .PRESET(PRESET),
    .CE(CE),
    .Q(Q)
);
defparam instName.INIT=1'b1;
```

### Vhdl 例化:

```
COMPONENT DFFPE
    GENERIC (INIT:bit:= '1');
    PORT(
        Q:OUT std_logic;
        D:IN std_logic;
        CLK:IN std_logic;
        PRESET:IN std_logic;
```

```

        CE:IN std_logic
    );
END COMPONENT;
 uut:DFFPE
    GENERIC MAP(INIT=>'1')
    PORT MAP (
        Q=>Q,
        D=>D,
        CLK=>CLK,
        PRESET=>PRESET,
        CE=>CE
    );

```

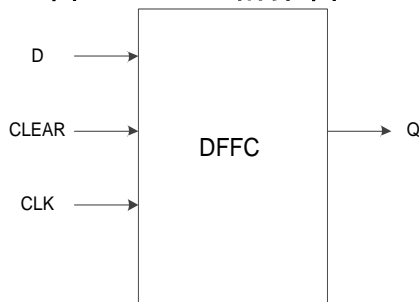
## 2.4.9 DFFC

### 原语介绍

DFFC(D Flip-Flop with Asynchronous Clear)是上升沿触发的 D 触发器，具有异步复位功能。

### 结构框图

图 2-19 DFFC 结构框图



### Port 介绍

表 2-41 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLK	Input	Clock Input
CLEAR	Input	Asynchronous Clear Input
Q	Output	Data Output

### Attribute 介绍

表 2-42 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b0	Initial value for DFFC

### 原语例化

**Verilog 例化:**  
DFFC instName (

```

        .D(D),
        .CLK(CLK),
        .CLEAR(CLEAR),
        .Q(Q)
    );
    defparam instName.INIT=1'b0;
Vhdl 例化:
    COMPONENT DFFC
        GENERIC (INIT:bit:=0');
        PORT(
            Q:OUT std_logic;
            D:IN std_logic;
            CLK:IN std_logic;
            CLEAR:IN std_logic
        );
    END COMPONENT;
    uut:DFFC
        GENERIC MAP(INIT=>'0')
        PORT MAP (
            Q=>Q,
            D=>D,
            CLK=>CLK,
            CLEAR=>CLEAR
        );

```

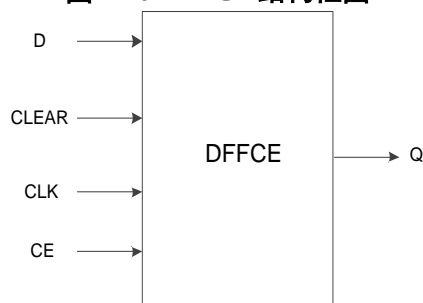
## 2.4.10 DFFCE

### 原语介绍

DFFCE(D Flip-Flop with Clock Enable and Asynchronous Clear)是上升沿触发的 D 触发器，具有异步复位和时钟使能功能。

### 结构框图

图 2-20 DFFCE 结构框图





## Port 介绍

表 2-43 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLK	Input	Clock Input
CLEAR	Input	Asynchronous Clear Input
CE	Input	Clock Enable
Q	Output	Data Output

## Attribute 介绍

表 2-44 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b0	Initial value for DFFCE

## 原语例化

### Verilog 例化:

```
DFFCE instName (
    .D(D),
    .CLK(CLK),
    .CLEAR(CLEAR),
    .CE(CE),
    .Q(Q)
);
defparam instName.INIT=1'b0;
```

### Vhdl 例化:

```
COMPONENT DFFCE
    GENERIC (INIT:bit:= '0');
    PORT(
        Q:OUT std_logic;
        D:IN std_logic;
        CLK:IN std_logic;
        CLEAR:IN std_logic;
        CE:IN std_logic
    );
END COMPONENT;
 uut:DFFCE
    GENERIC MAP(INIT=>'0')
    PORT MAP (
        Q=>Q,
        D=>D,
        CLK=>CLK,
        CLEAR=>CLEAR,
        CE=>CE
    );
```

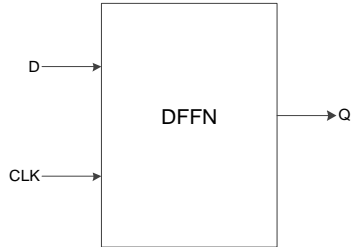
## 2.4.11 DFFN

### 原语介绍

DFFN(D Flip-Flop with Negative-Edge Clock)是下降沿触发的 D 触发器。

### 结构框图

图 2-21 DFFN 结构框图



### Port 介绍

表 2-45 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLK	Input	Clock Input
Q	Output	Data Output

### Attribute 介绍

表 2-46 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b0	Initial value for DFFN

### 原语例化

#### Verilog 例化:

```
DFFN instName (
    .D(D),
    .CLK(CLK),
    .Q(Q)
);
defparam instName.INIT=1'b0;
```

#### Vhdl 例化:

```
COMPONENT DFFN
    GENERIC (INIT:bit:= '0');
    PORT(
        Q:OUT std_logic;
        D:IN std_logic;
        CLK:IN std_logic
    );
END COMPONENT;
 uut:DFFN
    GENERIC MAP(INIT=>'0')
```

```

PORT MAP (
    Q=>Q,
    D=>D,
    CLK=>CLK
);

```

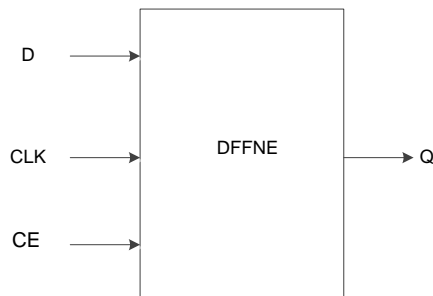
## 2.4.12 DFFNE

### 原语介绍

DFFNE(D Flip-Flop with Negative-Edge Clock and Clock Enable)是下降沿触发的 D 触发器，具有时钟使能功能。

### 结构框图

图 2-22 DFFNE 结构框图



### Port 介绍

表 2-47 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLK	Input	Clock Input
CE	Input	Clock Enable
Q	Output	Data Output

### Attribute 介绍

表 2-48 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b0	Initial value for DFFNE

### 原语例化

#### Verilog 例化:

```

DFFNE instName (
    .D(D),
    .CLK(CLK),
    .CE(CE),
    .Q(Q)
);
defparam instName.INIT=1'b0;

```

#### Vhdl 例化:

```

COMPONENT DFFNE
  GENERIC (INIT:bit:= '0');
  PORT(
    Q:OUT std_logic;
    D:IN std_logic;
    CLK:IN std_logic;
    CE:IN std_logic
  );
END COMPONENT;
 uut:DFFNE
  GENERIC MAP(INIT=>'0')
  PORT MAP (
    Q=>Q,
    D=>D,
    CLK=>CLK,
    CE=>CE
  );

```

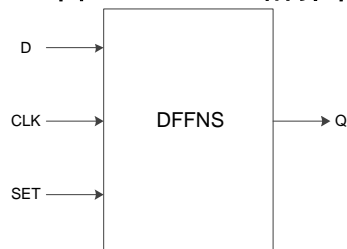
## 2.4.13 DFFNS

### 原语介绍

DFFNS(D Flip-Flop with Negative-Edge Clock and Synchronous Set) 是下降沿触发的 D 触发器，具有同步置位功能。

### 结构框图

图 2-23 DFFNS 结构框图



### Port 介绍

表 2-49 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLK	Input	Clock Input
SET	Input	Synchronous Set Input
Q	Output	Data Output

### Attribute 介绍

表 2-50 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0, 1'b1	1'b1	Initial value for DFFNS

## 原语例化

### Verilog 例化:

```
DFFNS instName (
    .D(D),
    .CLK(CLK),
    .SET(SET),
    .Q(Q)
);
defparam instName.INIT=1'b1;
```

### Vhdl 例化:

```
COMPONENT DFFNS
    GENERIC (INIT:bit:= '1');
    PORT(
        Q:OUT std_logic;
        D:IN std_logic;
        CLK:IN std_logic;
        SET:IN std_logic
    );
END COMPONENT;
 uut:DFFNS
    GENERIC MAP(INIT=>'1')
    PORT MAP (
        Q=>Q,
        D=>D,
        CLK=>CLK,
        SET=>SET
    );
```

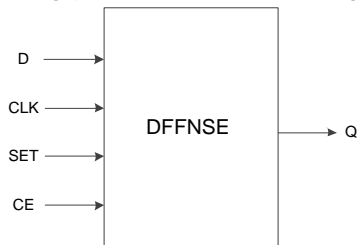
## 2.4.14 DFFNSE

### 原语介绍

DFFNSE(D Flip-Flop with Negative-Edge Clock,Clock Enable,and Synchronous Set)是下降沿触发的 D 触发器,具有同步置位和时钟使能功能。

### 结构框图

图 2-24 DFFNSE 结构框图



## Port 介绍

表 2-51 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLK	Input	Clock Input
SET	Input	Synchronous Set Input
CE	Input	Clock Enable
Q	Output	Data Output

## Attribute 介绍

表 2-52 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b1	Initial value for DFFNSE

## 原语例化

### Verilog 例化:

```
DFFNSE instName (
    .D(D),
    .CLK(CLK),
    .SET(SET),
    .CE(CE),
    .Q(Q)
);
defparam instName.INIT=1'b1;
```

### Vhdl 例化:

```
COMPONENT DFFNSE
    GENERIC (INIT:bit:= '1');
    PORT(
        Q:OUT std_logic;
        D:IN std_logic;
        CLK:IN std_logic;
        SET:IN std_logic;
        CE:IN std_logic
    );
END COMPONENT;
 uut:DFFNSE
    GENERIC MAP(INIT=>'1')
    PORT MAP (
        Q=>Q,
        D=>D,
        CLK=>CLK,
        SET=>SET,
        CE=>CE
    );
```

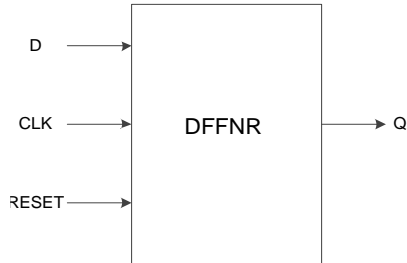
## 2.4.15 DFFNR

### 原语介绍

DFFNR(D Flip-Flop with Negative-Edge Clock and Synchronous Reset)是下降沿触发的 D 触发器，具有同步复位功能。

### 结构框图

图 2-25 DFFNR 结构框图



### Port 介绍

表 2-53 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLK	Input	Clock Input
RESET	Input	Synchronous Reset Input
Q	Output	Data Output

### Attribute 介绍

表 2-54 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b0	Initial value for DFFNR

### 原语例化

#### Verilog 例化:

```
DFFNR instName (
    .D(D),
    .CLK(CLK),
    .RESET(RESET),
    .Q(Q)
);
defparam instName.INIT=1'b0;
```

#### Vhdl 例化:

```
COMPONENT DFFNR
    GENERIC (INIT:bit:= '0');
    PORT(
        Q:OUT std_logic;
        D:IN std_logic;
        CLK:IN std_logic;
```

```

        RESET:IN std_logic
    );
END COMPONENT;
 uut:DFFNR
    GENERIC MAP(INIT=>'0')
    PORT MAP (
        Q=>Q,
        D=>D,
        CLK=>CLK,
        RESET=>RESET
    );

```

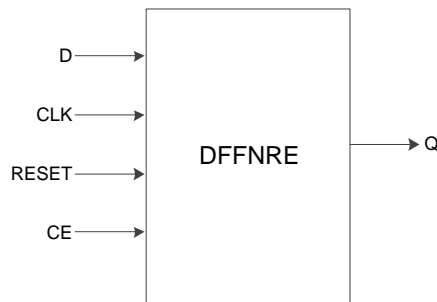
## 2.4.16 DFFNRE

### 原语介绍

DFFNRE(D Flip-Flop with Negative-Edge Clock,Clock Enable, and Synchronous Reset)是下降沿触发的 D 触发器，具有同步复位和时钟使能功能。

### 结构框图

图 2-26 DFFNRE 结构框图



### Port 介绍

表 2-55 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLK	Input	Clock Input
RESET	Input	Synchronous Reset Input
CE	Input	Clock Enable
Q	Output	Data Output

### Attribute 介绍

表 2-56 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b0	Initial value for DFFNRE

### 原语例化

**Verilog 例化:**



```

DFFNRE instName (
    .D(D),
    .CLK(CLK),
    .RESET(RESET),
    .CE(CE),
    .Q(Q)
);
defparam instName.INIT=1'b0;
Vhdl 例化:
COMPONENT DFFNRE
    GENERIC (INIT:bit:='0');
    PORT(
        Q:OUT std_logic;
        D:IN std_logic;
        CLK:IN std_logic;
        RESET:IN std_logic;
        CE:IN std_logic
    );
END COMPONENT;
 uut:DFFNRE
    GENERIC MAP(INIT=>'0')
    PORT MAP (
        Q=>Q,
        D=>D,
        CLK=>CLK,
        RESET=>RESET,
        CE=>CE
    );

```

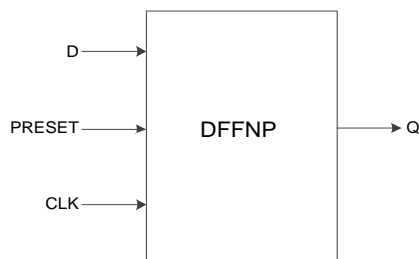
## 2.4.17 DFFNP

### 原语介绍

DFFNP(D Flip-Flop with Negative-Edge Clock and Asynchronous Preset)是下降沿触发的 D 触发器，具有异步置位功能。

### 结构框图

图 2-27 DFFNP 结构框图



## Port 介绍

表 2-57 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLK	Input	Clock Input
PRESET	Input	Asynchronous Preset Input
Q	Output	Data Output

## Attribute 介绍

表 2-58 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b1	Initial value for DFFNP

## 原语例化

### Verilog 例化:

```
DFFNP instName (
    .D(D),
    .CLK(CLK),
    .PRESET(PRESET),
    .Q(Q)
);
defparam instName.INIT=1'b1;
```

### Vhdl 例化:

```
COMPONENT DFFNP
    GENERIC (INIT:bit:= '1');
    PORT(
        Q:OUT std_logic;
        D:IN std_logic;
        CLK:IN std_logic;
        PRESET:IN std_logic
    );
END COMPONENT;
 uut:DFFNP
    GENERIC MAP(INIT=>'1')
    PORT MAP (
        Q=>Q,
        D=>D,
        CLK=>CLK,
        PRESET=>PRESET
    );
```

## 2.4.18 DFFNPE

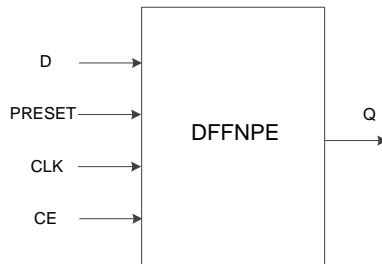
### 原语介绍

DFFNPE(D Flip-Flop with Negative-Edge Clock, Clock Enable, and

Asynchronous Preset)是下降沿触发的 D 触发器，具有异步置位和时钟使能功能。

### 结构框图

图 2-28 DFFNPE 结构框图



### Port 介绍

表 2-59 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLK	Input	Clock Input
PRESET	Input	Asynchronous Preset Input
CE	Input	Clock Enable
Q	Output	Data Output

### Attribute 介绍

表 2-60 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b1	Initial value for DFFNPE

### 原语例化

#### Verilog 例化:

```
DFFNPE instName (
    .D(D),
    .CLK(CLK),
    .PRESET(PRESET),
    .CE(CE),
    .Q(Q)
);
defparam instName.INIT=1'b1;
```

#### Vhdl 例化:

```
COMPONENT DFFNPE
    GENERIC (INIT:bit:=1);
    PORT(
        Q:OUT std_logic;
        D:IN std_logic;
        CLK:IN std_logic;
        PRESET:IN std_logic;
```

```

        CE:IN std_logic
    );
END COMPONENT;
 uut:DFFNPE
    GENERIC MAP(INIT=>'1')
    PORT MAP (
        Q=>Q,
        D=>D,
        CLK=>CLK,
        PRESET=>PRESET,
        CE=>CE
    );

```

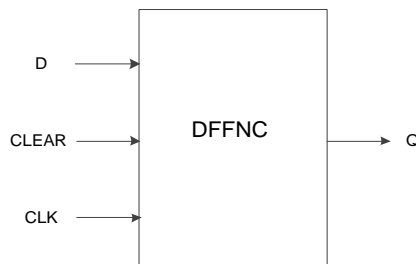
## 2.4.19 DFFNC

### 原语介绍

DFFNC(D Flip-Flop with Negative-Edge Clock and Asynchronous Clear)是下降沿触发的D触发器，具有异步复位功能。

### 结构框图

图 2-29 DFFNC 结构框图



### Port 介绍

表 2-61 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLK	Input	Clock Input
CLEAR	Input	Asynchronous Clear Input
Q	Output	Data Output

### Attribute 介绍

表 2-62 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b0	Initial value for DFFNC

### 原语例化

**Verilog 例化:**  
DFFNC instName (  
.D(D),  
.CLK(CLK),

```

        .CLEAR(CLEAR),
        .Q(Q)
    );
    defparam instName.INIT=1'b0;
Vhdl 例化:
    COMPONENT DFFNC
        GENERIC (INIT:bit:= '0');
        PORT(
            Q:OUT std_logic;
            D:IN std_logic;
            CLK:IN std_logic;
            CLEAR:IN std_logic
        );
    END COMPONENT;
    uut:DFFNC
        GENERIC MAP(INIT=>'0')
        PORT MAP (
            Q=>Q,
            D=>D,
            CLK=>CLK,
            CLEAR=>CLEAR
        );

```

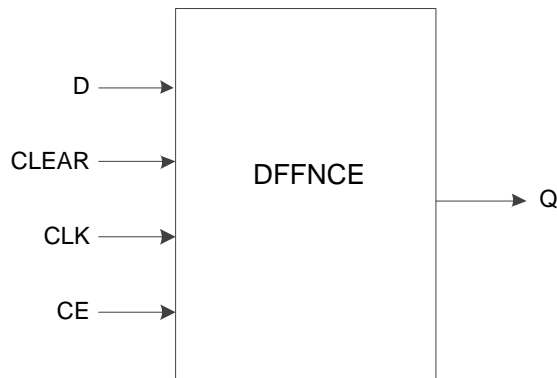
## 2.4.20 DFFNCE

### 原语介绍

DFFNCE(D Flip-Flop with Negative-Edge Clock,Clock Enable and Asynchronous Clear)是下降沿触发的 D 触发器,具有异步复位和时钟使能功能。

### 结构框图

图 2-30 DFFNCE 结构框图



## Port 介绍

表 2-63 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLK	Input	Clock Input
CLEAR	Input	Asynchronous Clear Input
CE	Input	Clock Enable
Q	Output	Data Output

## Attribute 介绍

表 2-64 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b0	Initial value for DFFNCE

## 原语例化

### Verilog 例化:

```
DFFNCE instName (
    .D(D),
    .CLK(CLK),
    .CLEAR(CLEAR),
    .CE(CE),
    .Q(Q)
);
defparam instName.INIT=1'b0;
```

### Vhdl 例化:

```
COMPONENT DFFNCE
    GENERIC (INIT:bit:= '0');
    PORT(
        Q:OUT std_logic;
        D:IN std_logic;
        CLK:IN std_logic;
        CLEAR:IN std_logic;
        CE:IN std_logic
    );
END COMPONENT;
 uut:DFFNCE
    GENERIC MAP(INIT=>'0')
    PORT MAP (
        Q=>Q,
        D=>D,
        CLK=>CLK,
        CLEAR=>CLEAR,
        CE=>CE
    );
```

## 2.5 LATCH

锁存器是一种对电平触发的存储单元电路，其可在特定输入电平作用下改变状态。

支持器件：GW1N-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、GW1NR-9、GW1NZ-1、GW1NSR-2、GW1NSR-2C、GW2A-18、GW2AR-18、GW2A-55。与 LATCH 相关的原语有 12 个，如表 2-65 所示。

表 2-65 与 LATCH 相关的原语

原语	描述
DL	数据锁存器
DLE	带锁存使能的数据锁存器
DLC	带异步清零的数据锁存器
DLCE	带异步清零和锁存使能的数据锁存器
DLP	带异步预置位的数据锁存器
DLPE	带异步预置位和锁存使能的数据锁存器
DLN	低电平有效的数据锁存器
DLNE	带锁存使能的低电平有效的数据锁存器
DLNC	带异步清零的低电平有效的数据锁存器
DLNCE	带异步清零和锁存使能的低电平有效的数据锁存器
DLNP	带异步预置位的低电平有效的数据锁存器
DLNPE	带异步预置位和锁存使能的低电平有效的数据锁存器

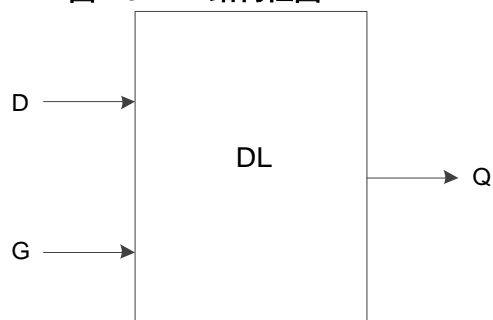
### 2.5.1 DL

#### 原语介绍

DL(Data Latch)是其中最简单常用的一种锁存器，控制信号 G 高电平有效。

#### 结构框图

图 2-31 DL 结构框图



## Port 介绍

表 2-66 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
G	Input	Control Signal Input
Q	Output	Data Output

## Attribute 介绍

表 2-67 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b0	Initial value for initial DL

## 原语例化

### Verilog 例化:

```
DL instName (
    .D(D),
    .G(G),
    .Q(Q)
);
defparam instName.INIT=1'b0;
```

### Vhdl 例化:

```
COMPONENT DL
    GENERIC (INIT:bit:= '0');
    PORT(
        Q:OUT std_logic;
        D:IN std_logic;
        G:IN std_logic
    );
END COMPONENT;
 uut:DL
    GENERIC MAP(INIT=>'0')
    PORT MAP (
        Q=>Q,
        D=>D,
        G=>G
    );
```

## 2.5.2 DLE

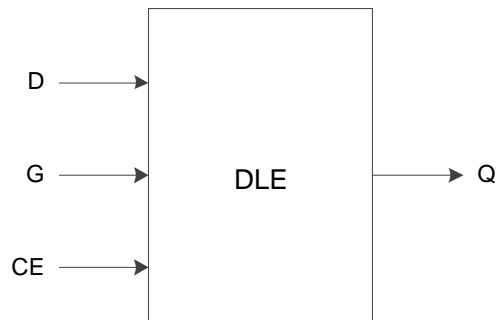
### 原语介绍

DLE(Data Latch with Latch Enable)是具有使能控制的一种锁存器，控制信号 G 高电平有效。



## 结构框图

图 2-32 DLE 结构框图



## Port 介绍

表 2-68 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
G	Input	Control Signal Input
CE	Input	Clock Enable
Q	Output	Data Output

## Attribute 介绍

表 2-69 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b0	Initial value for initial DLE

## 原语例化

### Verilog 例化:

```
DLE instName (
    .D(D),
    .G(G),
    .CE(CE),
    .Q(Q)
);
defparam instName.INIT=1'b0;
```

### Vhdl 例化:

```
COMPONENT DLE
    GENERIC (INIT:bit:=0');
    PORT(
        Q:OUT std_logic;
        D:IN std_logic;
        G:IN std_logic;
        CE:IN std_logic
    );
END COMPONENT;
 uut:DLE
```

```

GENERIC MAP(INIT=>'0')
PORT MAP (
    Q=>Q,
    D=>D,
    G=>G,
    CE=>CE
);

```

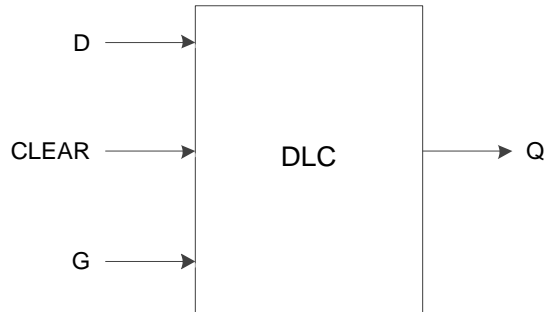
## 2.5.3 DLC

### 原语介绍

DLC(Data Latch with Asynchronous Clear)是具有复位功能的一种锁存器，控制信号 G 高电平有效。

### 结构框图

图 2-33 DLC 结构框图



### Port 介绍

表 2-70 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLEAR	Input	Asynchronous Clear Input
G	Input	Control Signal Input
Q	Output	Data Output

### Attribute 介绍

表 2-71 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b0	Initial value for initial DLC

### 原语例化

#### Verilog 例化:

```

DLC instName (
    .D(D),
    .G(G),
    .CLEAR(CLEAR),

```

```

        .Q(Q)
    );
    defparam instName.INIT=1'b0;
Vhdl 例化:
    COMPONENT DLC
        GENERIC (INIT:bit:= '0');
        PORT(
            Q:OUT std_logic;
            D:IN std_logic;
            G:IN std_logic;
            CLEAR:IN std_logic
        );
    END COMPONENT;
    uut:DLC
        GENERIC MAP(INIT=>'0')
        PORT MAP (
            Q=>Q,
            D=>D,
            G=>G,
            CLEAR=>CLEAR
        );

```

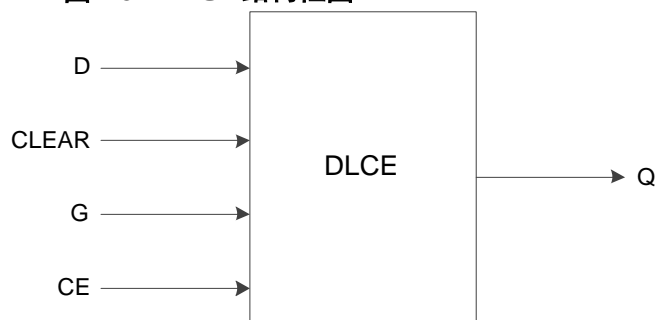
## 2.5.4 DLCE

### 原语介绍

DLCE(Data Latch with Asynchronous Clear and Latch Enable)是具有使能控制和复位功能的一种锁存器，控制信号 G 高电平有效。

### 结构框图

图 2-34 DLCE 结构框图



## Port 介绍

表 2-72 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLEAR	Input	Asynchronous Clear Input
G	Input	Control Signal Input
CE	Input	Clock Enable
Q	Output	Data Output

## Attribute 介绍

表 2-73 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b0	Initial value for initial DLCE

## 原语例化

### Verilog 例化:

```
DLCE instName (
    .D(D),
    .CLEAR(CLEAR),
    .G(G),
    .CE(CE),
    .Q(Q)
);
defparam instName.INIT=1'b0;
```

### Vhdl 例化:

```
COMPONENT DLCE
    GENERIC (INIT:bit:= '0');
    PORT(
        Q:OUT std_logic;
        D:IN std_logic;
        G:IN std_logic;
        CE:IN std_logic;
        CLEAR:IN std_logic
    );
END COMPONENT;
 uut:DLCE
    GENERIC MAP(INIT=>'0')
    PORT MAP (
        Q=>Q,
        D=>D,
        G=>G,
        CE=>CE,
        CLEAR=>CLEAR
    );
```

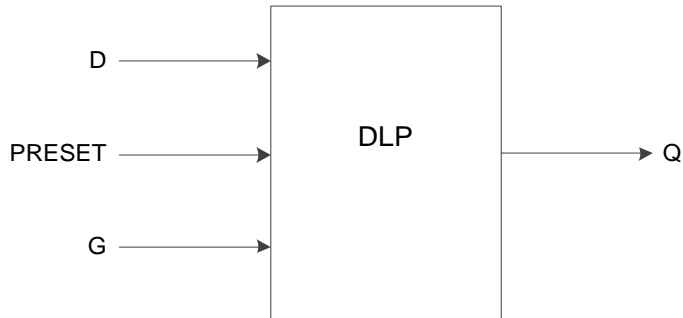
## 2.5.5 DLP

### 原语介绍

DLP(Data Latch with Asynchronous Preset)是具有置位功能的一种锁存器，控制信号 G 高电平有效。

### 结构框图

图 2-35 DLP 结构框图



### Port 介绍

表 2-74 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
PRESET	Input	Asynchronous Preset Input
G	Input	Control Signal Input
Q	Output	Data Output

### Attribute 介绍

表 2-75 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b1	Initial value for initial DLP

### 原语例化

#### Verilog 例化:

```
DLP instName (
    .D(D),
    .G(G),
    .PRESET(PRESET),
    .Q(Q)
);
defparam instName.INIT=1'b1;
```

#### Vhdl 例化:

```
COMPONENT DLP
    GENERIC (INIT:bit:= '1');
    PORT(
        Q:OUT std_logic;
```

```

        D:IN std_logic;
        G:IN std_logic;
        PRESET:IN std_logic
    );
END COMPONENT;
 uut:DLP
    GENERIC MAP(INIT=>'1')
    PORT MAP (
        Q=>Q,
        D=>D,
        G=>G,
        PRESET => PRESET
    );

```

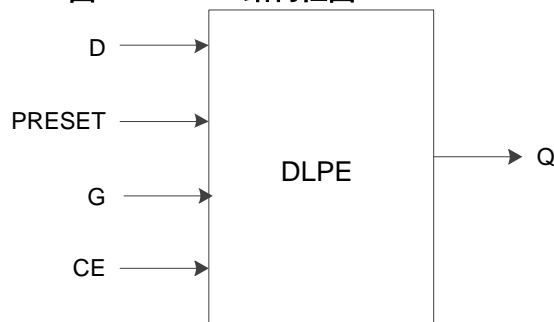
## 2.5.6 DLPE

### 原语介绍

DLPE(Data Latch with Asynchronous Preset and Latch Enable)是具有使能控制和置位功能的一种锁存器，控制信号 G 高电平有效。

### 结构框图

图 2-36 DLPE 结构框图



### Port 介绍

表 2-76 Port 介绍

Port Name	I/O	Description
D	Input	Data Output
PRESET	Input	Asynchronous Preset Input
G	Input	Control Signal Input
CE	Input	Clock Enable
Q	Output	Data Output

### Attribute 介绍

表 2-77 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b1	Initial value for initial DLPE

## 原语例化

### Verilog 例化:

```
DLPE instName (
    .D(D),
    .PRESET(PRESET),
    .G(G),
    .CE(CE),
    .Q(Q)
);
defparam instName.INIT=1'b1;
```

### Vhdl 例化:

```
COMPONENT DLPE
    GENERIC (INIT:bit:= '1');
    PORT(
        Q:OUT std_logic;
        D:IN std_logic;
        G:IN std_logic;
        CE:IN std_logic;
        PRESET:IN std_logic
    );
END COMPONENT;
 uut:DLPE
    GENERIC MAP(INIT=>'1')
    PORT MAP (
        Q=>Q,
        D=>D,
        G=>G,
        CE=>CE
        PRESET =>PRESET
    );
```

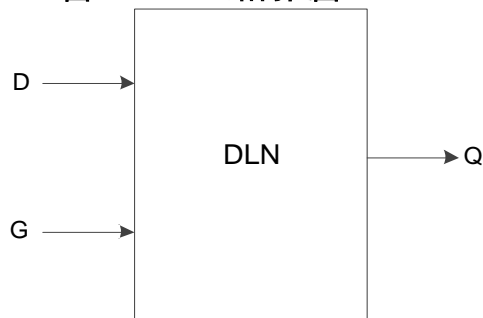
## 2.5.7 DLN

### 原语介绍

DLN(Data Latch with Inverted Gate)是控制信号低电平有效的锁存器。

### 结构框图

图 2-37 DLN 结构框图



## Port 介绍

表 2-78 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
G	Input	Control Signal Input
Q	Output	Data Output

## Attribute 介绍

表 2-79 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b0	Initial value for initial DLN

## 原语例化

### Verilog 例化:

```
DLN instName (
    .D(D),
    .G(G),
    .Q(Q)
);
defparam instName.INIT=1'b0;
```

### Vhdl 例化:

```
COMPONENT DLN
    GENERIC (INIT:bit:= '0');
    PORT(
        Q:OUT std_logic;
        D:IN std_logic;
        G:IN std_logic
    );
END COMPONENT;
uut:DLN
    GENERIC MAP(INIT=>'0')
    PORT MAP (
        Q=>Q,
        D=>D,
        G=>G
    );
```

## 2.5.8 DLNE

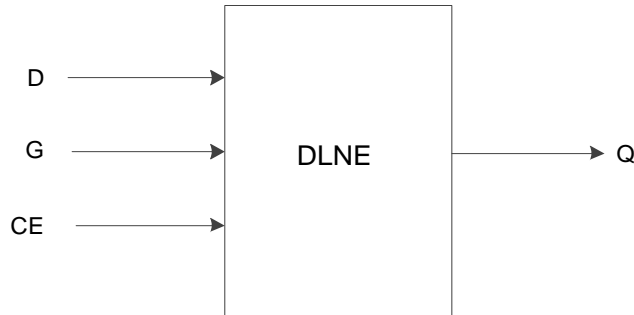
### 原语介绍

DLNE(Data Latch with Latch Enable and Inverted Gate)是一种具有使能控制的锁存器，控制信号 G 低电平有效。



## 结构框图

图 2-38 DLNE 结构框图



## Port 介绍

表 2-80 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
G	Input	Control Signal Input
CE	Input	Clock Enable
Q	Output	Data Output

## Attribute 介绍

表 2-81 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b0	Initial value for initial DLNE

## 原语例化

### Verilog 例化:

```

DLNE instName (
    .D(D),
    .G(G),
    .CE(CE),
    .Q(Q)
);
defparam instName.INIT=1'b0;
  
```

### Vhdl 例化:

```

COMPONENT DLNE
  GENERIC (INIT:bit:= '0');
  PORT(
    Q:OUT std_logic;
    D:IN std_logic;
    G:IN std_logic;
    CE:IN std_logic
  );
END COMPONENT;
 uut:DLNE
  
```

```

GENERIC MAP(INIT=>'0')
PORT MAP (
    Q=>Q,
    D=>D,
    G=>G,
    CE => CE
);

```

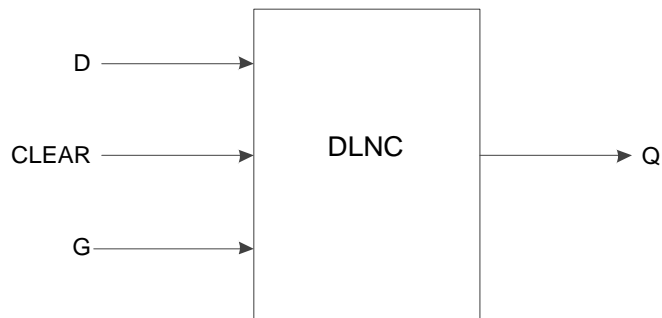
## 2.5.9 DLNC

### 原语介绍

DLNC(Data Latch with Asynchronous Clear and Inverted Gate)是一种具有复位功能的锁存器，控制信号 G 低电平有效。

### 结构框图

图 2-39 DLNC 结构框图



### Port 介绍

表 2-82 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLEAR	Input	Asynchronous Clear Input
G	Input	Control Signal Input
Q	Output	Data Output

### Attribute 介绍

表 2-83 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b0	Initial value for initial DLNC

### 原语例化

#### Verilog 例化:

```

DLNC instName (
    .D(D),
    .G(G),

```

```

        .CLEAR(CLEAR),
        .Q(Q)
    );
    defparam instName.INIT=1'b0;
Vhdl 例化:
    COMPONENT DLNC
        GENERIC (INIT:bit:= '0');
        PORT(
            Q:OUT std_logic;
            D:IN std_logic;
            G:IN std_logic;
            CLEAR:IN std_logic
        );
    END COMPONENT;
    uut:DLNC
        GENERIC MAP(INIT=>'0')
        PORT MAP (
            Q=>Q,
            D=>D,
            G=>G,
            CLEAR => CLEAR
        );

```

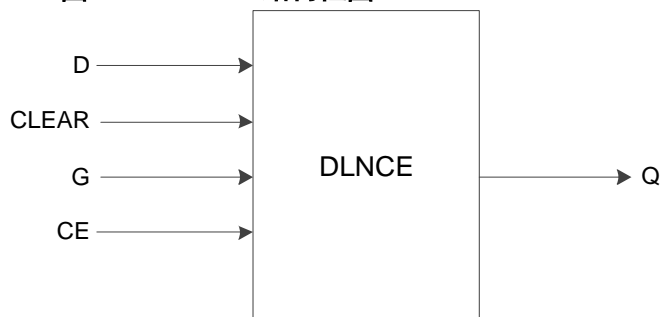
## 2.5.10 DLNCE

### 原语介绍

DLNCE(Data Latch with Asynchronous Clear, Latch Enable, and Inverted Gate)是具有使能控制和复位功能的一种锁存器，控制信号 G 低电平有效。

### 结构框图

图 2-40 DLNCE 结构框图



## Port 介绍

表 2-84 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLEAR	Input	Asynchronous Clear Input
G	Input	Control Signal Input
CE	Input	Clock Enable
Q	Output	Data Output

## Attribute 介绍

表 2-85 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b0	Initial value for initial DLNCE

## 原语例化

### Verilog 例化:

```
DLNCE instName (
    .D(D),
    .CLEAR(CLEAR),
    .G(G),
    .CE(CE),
    .Q(Q)
);
defparam instName.INIT=1'b0;
```

### Vhdl 例化:

```
COMPONENT DLNCE
    GENERIC (INIT:bit:= '0');
    PORT(
        Q:OUT std_logic;
        D:IN std_logic;
        G:IN std_logic;
        CE:IN std_logic;
        CLEAR:IN std_logic
    );
END COMPONENT;
 uut:DLNCE
    GENERIC MAP(INIT=>'0'
    )
    PORT MAP (
        Q=>Q,
        D=>D,
        G=>G,
        CE=>CE,
        CLEAR=>CLEAR
    );
```

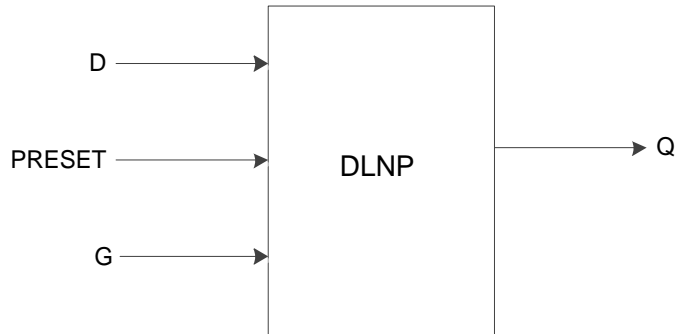
## 2.5.11 DLNP

### 原语介绍

DLNP(Data Latch with Asynchronous Clear and Inverted Gate)是具有置位功能的一种锁存器，控制信号 G 低电平有效。

### 结构框图

图 2-41 DLNP 结构框图



### Port 介绍

表 2-86 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
PRESET	Input	Asynchronous Preset Input
G	Input	Control Signal Input
Q	Output	Data Output

### Attribute 介绍

表 2-87 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b1	Initial value for initial DLNPE

### 原语例化

#### Verilog 例化:

```

DLNP instName (
    .D(D),
    .G(G),
    .PRESET(PRESET),
    .Q(Q)
);
defparam instName.INIT=1'b1;
  
```

#### Vhdl 例化:

```

COMPONENT DLNP
  GENERIC (INIT:bit:= '1');
  PORT(
  
```

```

        Q:OUT std_logic;
        D:IN std_logic;
        G:IN std_logic;
        PRESET:IN std_logic
    );
END COMPONENT;
 uut:DLNP
    GENERIC MAP(INIT=>'1')
    PORT MAP (
        Q=>Q,
        D=>D,
        G=>G,
        PRESET => PRESET
    );

```

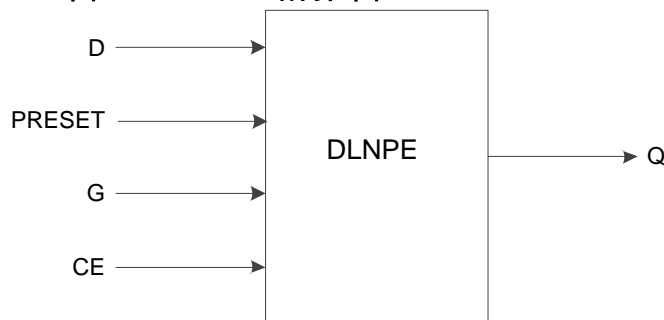
## 2.5.12 DLNPE

### 原语介绍

DLNPE(Data Latch with Asynchronous Preset,Latch Enable and Inverted Gate)是具有使能控制和置位功能的一种锁存器，控制信号 G 低电平有效。

### 结构框图

图 2-42 DLNPE 结构框图



### Port 介绍

表 2-88 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
PRESET	Input	Asynchronous Preset Input
G	Input	Control Signal Input
CE	Input	Clock Enable
Q	Output	Data Output

### Attribute 介绍

表 2-89 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b1	Initial value for initial DLNPE

## 原语例化

### Verilog 例化:

```
DLNPE instName (  
    .D(D),  
    .PRESET(PRESET),  
    .G(G),  
    .CE(CE),  
    .Q(Q)  
);  
defparam instName.INIT=1'b1;
```

### Vhdl 例化:

```
COMPONENT DLNPE  
    GENERIC (INIT:bit:= '1');  
    PORT(  
        Q:OUT std_logic;  
        D:IN std_logic;  
        G:IN std_logic;  
        CE:IN std_logic;  
        PRESET:IN std_logic  
    );  
END COMPONENT;  
uut:DLNPE  
    GENERIC MAP(INIT=>'1')  
    PORT MAP (  
        Q=>Q,  
        D=>D,  
        G=>G,  
        CE=>CE,  
        PRESET => PRESET  
    );
```

# 3<sub>CFU</sub>

CFU(Configurable Fuction Unit) 是可配置功能单元。与 CLU 不同的是，CFU 可配置为 SSRAM 模式。

## 3.1 SSRAM

SSRAM 是分布式静态随机存储器，可配置成单端口模式，半双端口模式和只读模式，如表 3-1 所示。

支持器件：GW1NS-2、GW1NS-2C、GW1N-6、GW1N-9、GW1NR-9、GW1NZ-1、GW1NSR-2、GW1NSR-2C、GW2A-18、GW2AR-18、GW2A-55。

表 3-1 SSRAM

原语	描述
RAM16S1	地址深度 16，数据宽度为 1 的单端口 SSRAM
RAM16S2	地址深度 16，数据宽度为 2 的单端口 SSRAM
RAM16S4	地址深度 16，数据宽度为 4 的单端口 SSRAM
RAM16SDP1	地址深度 16，数据宽度为 1 的半双端口 SSRAM
RAM16SDP2	地址深度 16，数据宽度为 2 的半双端口 SSRAM
RAM16SDP4	地址深度 16，数据宽度为 4 的半双端口 SSRAM
ROM16	地址深度 16，数据宽度为 1 的只读 ROM

### 3.1.1 RAM16S1

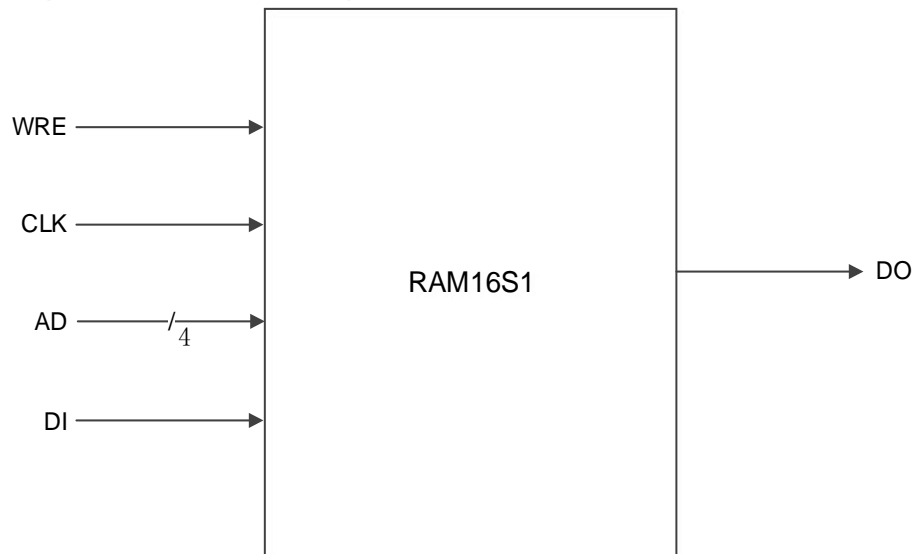
#### 原语介绍

RAM16S1(16-Deep by 1-Wide Single-port SSRAM)是地址深度为 16，数据位宽为 1 的单端口 SSRAM。



## 结构框图

图 3-1 RAM16S1 结构框图



## Port 介绍

表 3-2 Port 介绍

Port Name	I/O	Description
DI	Input	Data Input
CLK	Input	Clock Input
WRE	Input	Write Enable Input
AD[3:0]	Input	Address Input
DO	Output	Data Output

## Attribute 介绍

表 3-3 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT_0	16'h0000~16'hffff	16'h0000	Specifies Initial Contents of the RAM

## 原语例化

### Verilog 例化:

```
RAM16S1 instName(
    .DI(DI),
    .WRE(WRE),
    .CLK(CLK),
    .AD(AD[3:0]),
    .DO(DOUT)
);
defparam instName.INIT_0=16'h1100;
```

### Vhdl 例化:

```

COMPONENT RAM16S1
  GENERIC (INIT:bit_vector:=X"0000");
  PORT(
    DO:OUT std_logic;
    DI:IN std_logic;
    CLK:IN std_logic;
    WRE:IN std_logic;
    AD:IN std_logic_vector(3 downto 0)
  );
END COMPONENT;
 uut:RAM16S1
  GENERIC MAP(INIT=>X"0000")
  PORT MAP (
    DO=>DOUT,
    DI=>DI,
    CLK=>CLK,
    WRE=>WRE,
    AD=>AD
  );

```

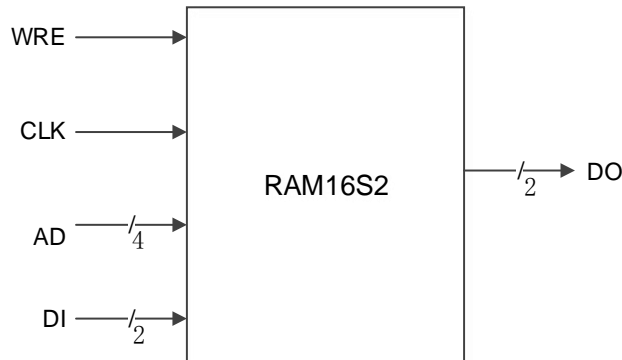
### 3.1.2 RAM16S2

#### 原语介绍

RAM16S2(16-Deep by 2-Wide Single-port SSRAM)是地址深度为 16, 数据位宽为 2 的单端口 SSRAM。

#### 结构框图

图 3-2 RAM16S2 结构框图



#### Port 介绍

表 3-4 Port 介绍

Port Name	I/O	Description
DI[1:0]	Input	Data Input
CLK	Input	Clock Input
WRE	Input	Write Enable Input
AD[3:0]	Input	Address Input
DO[1:0]	Output	Data Output

## Attribute 介绍

表 3-5 Attribute 介绍

Attribute Name	AllowedValues	Default	Description
INIT_0~ INIT_1	16'h0000~16'hffff	16'h0000	Specifies Initial Contents of the RAM

## 原语例化

### Verilog 例化:

```
RAM16S2 instName(
    .DI(DI[1:0]),
    .WRE(WRE),
    .CLK(CLK),
    .AD(AD[3:0]),
    .DO(DOUT[1:0])
);
defparam instName.INIT_0=16'h0790;
defparam instName.INIT_1=16'h0f00;
```

### Vhdl 例化:

```
COMPONENT RAM16S2
    GENERIC (INIT_0:bit_vector:=X"0000";
            INIT_1:bit_vector:=X"0000"
    );
    PORT(
        DO:OUT std_logic_vector(1 downto 0);
        DI:IN std_logic_vector(1 downto 0);
        CLK:IN std_logic;
        WRE:IN std_logic;
        AD:IN std_logic_vector(3 downto 0)
    );
END COMPONENT;
 uut:RAM16S2
    GENERIC MAP(INIT_0=>X"0000",
                INIT_1=>X"0000"
    )
    PORT MAP (
        DO=>DOUT,
        DI=>DI,
        CLK=>CLK,
        WRE=>WRE,
        AD=>AD
    );
```

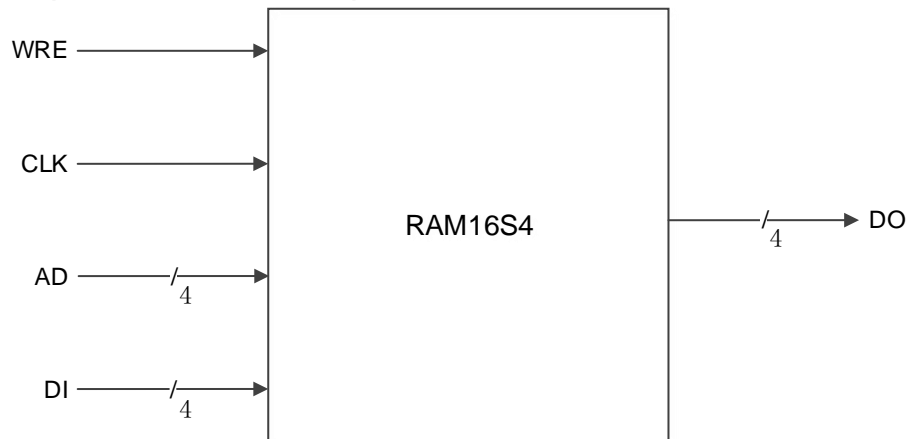
## 3.1.3 RAM16S4

### 原语介绍

RAM16S4(16-Deep by 4-Wide Single-port SSRAM)是地址深度为 16 ，数据位宽为 4 的单端口 SSRAM。

## 结构框图

图 3-3 RAM16S4 结构框图



## Port 介绍

表 3-6 Port 介绍

Port Name	I/O	Description
DI[3:0]	Input	Data Input
CLK	Input	Clock Input
WRE	Input	Write Enable Input
AD[3:0]	Input	Address Input
DO[3:0]	Output	Data Output

## Attribute 介绍

表 3-7 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT_0~ INIT_3	16'h0000~16'hffff	16'h0000	Specifies Initial Contents of the RAM

## 原语例化

### Verilog 例化:

```
RAM16S4 instName(
    .DI(DI[3:0]),
    .WRE(WRE),
    .CLK(CLK),
    .AD(AD[3:0]),
    .DO(DOUT[3:0])
);
defparam instName.INIT_0=16'h0450;
defparam instName.INIT_1=16'h1ac3;
defparam instName.INIT_2=16'h1240;
defparam instName.INIT_3=16'h045c;
```

### Vhdl 例化:

```
COMPONENT RAM16S4
```

```

        GENERIC (INIT_0:bit_vector:=X"0000";
                 INIT_1:bit_vector:=X"0000";
                 INIT_2:bit_vector:=X"0000";
                 INIT_3:bit_vector:=X"0000"
        );
        PORT(
            DO:OUT std_logic_vector(3 downto 0);
            DI:IN std_logic_vector(3 downto 0);
            CLK:IN std_logic;
            WRE:IN std_logic;
            AD:IN std_logic_vector(3 downto 0)
        );
    END COMPONENT;
    uut:RAM16S4
        GENERIC MAP(INIT_0=>X"0000",
                   INIT_1=>X"0000",
                   INIT_2=>X"0000",
                   INIT_3=>X"0000"
        )
        PORT MAP (
            DO=>DOUT,
            DI=>DI,
            CLK=>CLK,
            WRE=>WRE,
            AD=>AD
        );

```

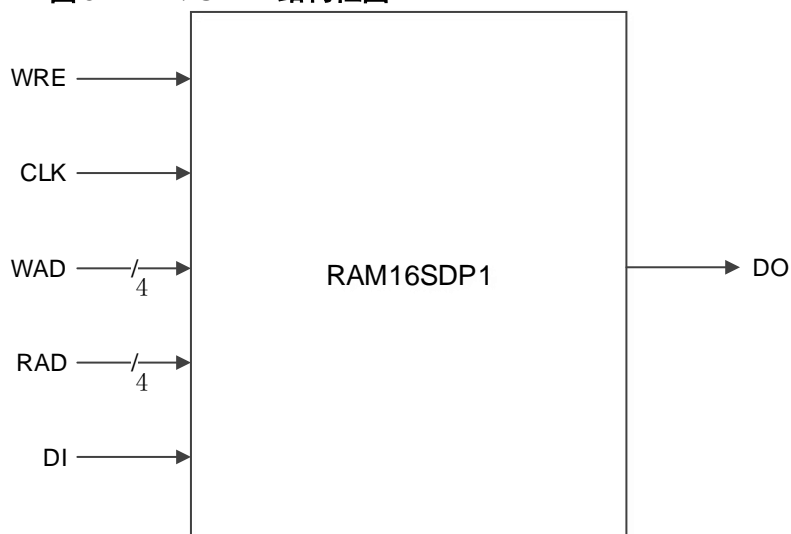
### 3.1.4 RAM16SDP1

#### 原语介绍

RAM16SDP1(16-Deep by 1-Wide Semi Dual-port SSRAM)是地址深度为 16 ，数据位宽为 1 的半双端口 SSRAM。

#### 结构框图

图 3-4 RAMSDP1 结构框图



## Port 介绍

表 3-8 Port 介绍

Port Name	I/O	Description
DI	Input	Data Input
CLK	Input	Clock Input
WRE	Input	Write Enable Input
WAD[3:0]	Input	Write Address
RAD[3:0]	Input	Read Address
DO	Output	Data Output

## Attribute 介绍

表 3-9 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT_0	16'h0000~16'hffff	16'h0000	Specifies Initial Contents of the RAM

## 原语例化

### Verilog 例化:

```
RAM16SDP1 instName(
    .DI(DI),
    .WRE(WRE),
    .CLK(CLK),
    .WAD(WAD[3:0]),
    .RAD(RAD[3:0]),
    .DO(DOUT)
);
defparam instName.INIT_0=16'h0100;
```

### Vhdl 例化:

```
COMPONENT RAM16SDP1
    GENERIC (INIT_0:bit_vector:=X"0000");
    PORT(
        DO:OUT std_logic;
        DI:IN std_logic;
        CLK:IN std_logic;
        WRE:IN std_logic;
        WAD:IN std_logic_vector(3 downto 0);
        RAD:IN std_logic_vector(3 downto 0)
    );
END COMPONENT;
 uut:RAM16SDP1
    GENERIC MAP(INIT_0=>X"0000")
    PORT MAP (
        DO=>DOUT,
        DI=>DI,
```

```

        CLK=>CLK,
        WRE=>WRE,
        WAD=>WAD,
        RAD=>RAD
    );

```

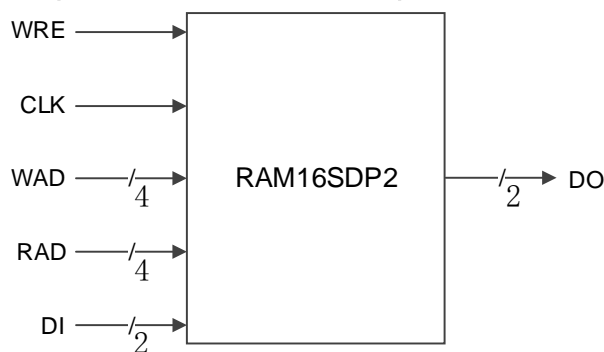
### 3.1.5 RAM16SDP2

#### 原语介绍

RAM16SDP2(16-Deep by 2-Wide Semi Dual-port SSRAM)是地址深度为 16 ，数据位宽为 2 的半双端口 SSRAM。

#### 结构框图

图 3-5 RAM16SDP2 结构框图



#### Port 介绍

表 3-10 Port 介绍

Port Name	I/O	Description
DI[1:0]	Input	Data Input
CLK	Input	Clock Input
WRE	Input	Write Enable Input
WAD[3:0]	Input	Write Address
RAD[3:0]	Input	Read Address
DO[1:0]	Output	Data Output

#### Attribute 介绍

表 3-11 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT_0~ INIT_1	16'h0000~ 16'hffff	16'h0000	Specifies Initial Contents of the RAM

#### 原语例化

**Verilog 例化:**  
RAM16SDP2 instName(

```

        .DI(DI[1:0]),
        .WRE(WRE),
        .CLK(CLK),
        .WAD(WAD[3:0]),
        .RAD(RAD[3:0]),
        .DO(DOUT[1:0])
    );
    defparam instName.INIT_0=16'h5600;
    defparam instName.INIT_1=16'h0af0;
Vhdl 例化:
    COMPONENT RAM16SDP2
        GENERIC (INIT_0:bit_vector:=X"0000";
                INIT_1:bit_vector:=X"0000"
        );
        PORT(
            DO:OUT std_logic_vector(1 downto 0);
            DI:IN std_logic_vector(1 downto 0);
            CLK:IN std_logic;
            WRE:IN std_logic;
            WAD:IN std_logic_vector(3 downto 0);
            RAD:IN std_logic_vector(3 downto 0)
        );
    END COMPONENT;
    uut:RAM16SDP2
        GENERIC MAP(INIT_0=>X"0000",
                   INIT_1=>X"0000"
        )
        PORT MAP (
            DO=>DOUT,
            DI=>DI,
            CLK=>CLK,
            WRE=>WRE,
            WAD=>WAD,
            RAD=>RAD
        );

```

### 3.1.6 RAM16SDP4

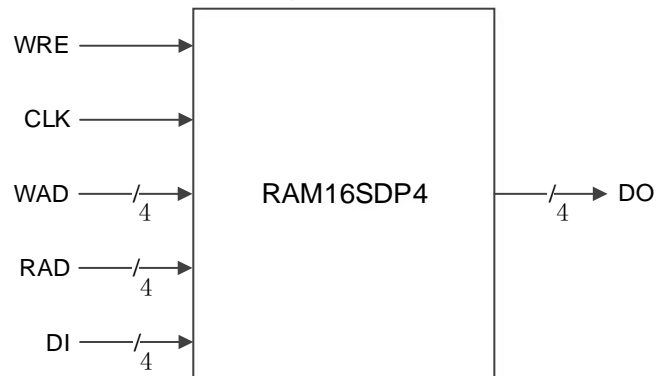
#### 原语介绍

RAM16SDP4(16-Deep by 4-Wide Semi Dual-port SSRAM)是地址深度为 16 ，数据位宽为 4 的半双端口 SSRAM。



## 结构框图

图 3-6 RAMSDP4 结构框图



## Port 介绍

表 3-12 Port 介绍

Port Name	I/O	Description
DI[3:0]	Input	Data Input
CLK	Input	Clcok Input
WRE	Input	Write Enable Input
WAD[3:0]	Input	Write Address
RAD[3:0]	Input	Read Address
DO[3:0]	Output	Data Output

## Attribute 介绍

表 3-13 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT_0~ INIT_3	16'h0000~16'hffff	16'h0000	Specifies Initial Contents of the RAM

## 原语例化

### Verilog 例化:

```
RAM16SDP4 instName(
    .DI(DI[3:0]),
    .WRE(WRE),
    .CLK(CLK),
    .WAD(WAD[3:0]),
    .RAD(RAD[3:0]),
    .DO(DOUT[3:0])
);
defparam instName.INIT_0=16'h0340;
defparam instName.INIT_1=16'h9065;
defparam instName.INIT_2=16'hac12;
defparam instName.INIT_3=16'h034c;
```

### Vhdl 例化:

```

COMPONENT RAM16SDP2
  GENERIC (INIT_0:bit_vector:=X"0000";
          INIT_1:bit_vector:=X"0000";
          INIT_2:bit_vector:=X"0000";
          INIT_3:bit_vector:=X"0000";
  );
  PORT(
    DO:OUT std_logic_vector(3 downto 0);
    DI:IN std_logic_vector(3 downto 0);
    CLK:IN std_logic;
    WRE:IN std_logic;
    WAD:IN std_logic_vector(3 downto 0);
    RAD:IN std_logic_vector(3 downto 0)
  );
END COMPONENT;
 uut:RAM16SDP2
  GENERIC MAP(INIT_0=>X"0000",
             INIT_1=>X"0000",
             INIT_2=>X"0000",
             INIT_3=>X"0000"
  )
  PORT MAP (
    DO=>DOUT,
    DI=>DI,
    CLK=>CLK,
    WRE=>WRE,
    WAD=>WAD,
    RAD=>RAD
  );

```

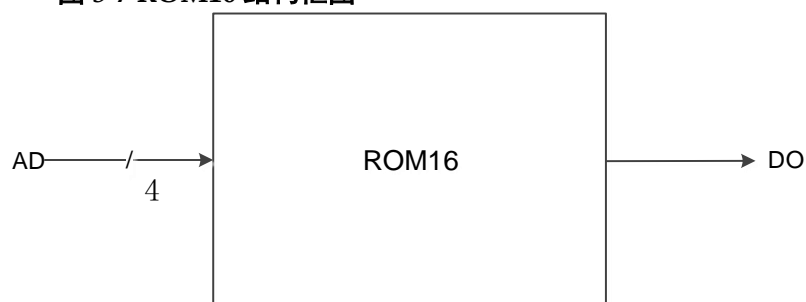
### 3.1.7 ROM16

#### 原语介绍

ROM16 是地址深度为 16，数据位宽为 1 的只读存储器，存储器的内容通过 INIT 进行初始化。

#### 结构框图

图 3-7 ROM16 结构框图



## Port 介绍

表 3-14 Port 介绍

Port Name	I/O	Description
AD[3:0]	Input	Address Input
DO	Output	Data Output

## Attribute 介绍

表 3-15 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT_0	16'h0000~16'hffff	16'h0000	Specifies Initial Contents of the ROM

## 原语例化

### Verilog 例化:

```
ROM16 instName (
    .AD(AD[3:0]),
    .DO(DOUT)
);
defparam instName.INIT_0=16'hfc00;
```

### Vhdl 例化:

```
COMPONENT ROM16
    GENERIC (INIT:bit_vector:=X"0000");
    PORT(
        DO:OUT std_logic;
        AD:IN std_logic_vector(3 downto 0)
    );
END COMPONENT;
 uut:ROM16
    GENERIC MAP(INIT=>X"0000")
    PORT MAP (
        DO=>DOUT,
        AD=>AD
    );
```

# 4Block SRAM

**Block SRAM**—块状静态随机存储器，具有静态存取功能。根据配置模式，可分为单端口模式（SP/SPX9）、双端口模式（DP/DPX9）、半双端口模式（SDP/SDPX9）和只读模式（ROM/ROMX9）。

支持器件：GW1N-1、GW1NZ-1、GW1N-2、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1N-2B、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55。

## 4.1 SP/SPX9

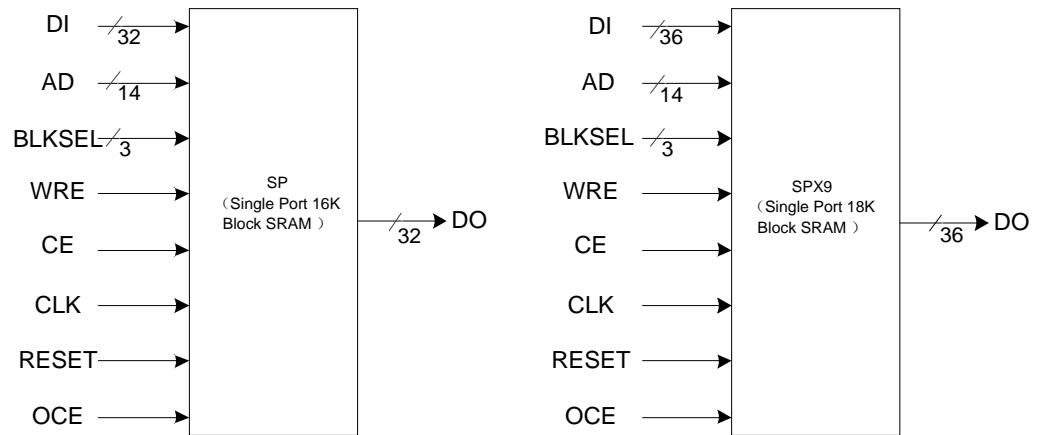
### 原语介绍

SP/SPX9(Single Port 16K Block SRAM/Single Port 18K Block SRAM),16K/18K 单端口 BSRAM 的存储空间为 16K bit/18K bit, 其工作模式为单端口模式，由一个时钟控制单端口的读/写操作，可支持 2 种读模式（bypass 模式和 pipeline 模式）和 3 种写模式（normal 模式、write-through 模式和 read-before-write 模式）。

SP 配置为 16bit/32bit、SPX9 配置为 18bit/36bit 时，可实现 BSRAM 的 byte enable 功能，即通过写入地址端口 AD 的低四位控制写入 memory 的数据，高电平使能。AD[0]控制 DI[7:0]/DI[8:0]是否写入 memory，AD[1]控制 DI[15:8]/DI[17:9]是否写入 memory，AD[2]控制 DI[23:16]/DI[26:18]是否写入 memory，AD[3]控制 DI[31:24]/DI[35:27]是否写入 memory。

## 结构框图

图 4-1 SP/SPX9 结构框图



## Port 介绍

表 4-1 Port 介绍

Port Name	I/O	Description
DO[31:0]/DO[35:0]	Output	Data Output
DI[31:0]/DI[35:0]	Input	Data Input
AD[13:0]	Input	Address Input
WRE	Input	Write Enable
CE	Input	Clock Enable
CLK	Input	Clock Input
RESET	Input	Reset Input
OCE	Input	Output Clock Enable
BLKSEL[2:0]	Input	Block RAM Selection Input

## Attribute 介绍

表 4-2 Attribute 介绍

Attribute Name	Type	Allowed Values	Default	Description
READ_MODE	Integer	1'b0,1'b1	1'b0	Output pipeline register can be bypassed 1'b0:bypass mode 1'b1:pipeline mode
WRITE_MODE	Integer	2'b00,2'b01,2'b10	2'b00	Write mode can be selected. 2'b00: normal mode 2'b01:write-through mode; 2'b10: read-before-write mode
BIT_WIDTH	Integer	SP:1,2,4,8,16,32 SPX9:9,18,36	SP:32 SPX9:36	Data width can be configured
BLK_SEL	Integer	3'b000~3'b111	3'b000	Block SRAM Selection
RESET_MODE	String	SYNC,ASYNC	SYNC	Reset mode config, synchronous or asynchronous
INIT_RAM_00~ INIT_RAM_3F	Integer	SP:256'h0...0~256'h1...1 SPX9:288'h0...0~288'h1...1	SP:256'h0...0 SPX9:288'h0...0	Initial value for initial BSRAM

## 配置关系

表 4-3 配置关系

Single Port Mode	BSRAM Capacity	Data Width	Address Depth
SP	16K	1	14
		2	13
		4	12
		8	11
		16	10
		32	9
SPX9	18K	9	11
		18	10
		36	9

## 原语例化

示例一

## Verilog 例化:

```

SP bram_sp_0 (
    .DO({dout[31:8], dout[7:0]}),
    .CLK(clk),
    .OCE(oce),
    .CE(ce),
    .RESET(reset),
    .WRE(wre),
    .BLKSEL({3'b000}),

```

```

        .AD({ad[10:0], 3'b000}),
        .DI({{24{1'b0}}, din[7:0]})
    );
    defparam bram_sp_0.READ_MODE = 1'b0;
    defparam bram_sp_0.WRITE_MODE = 2'b00;
    defparam bram_sp_0.BIT_WIDTH = 8;
    defparam bram_sp_0.BLK_SEL = 3'b000;
    defparam bram_sp_0.RESET_MODE = "SYNC";
    defparam bram_sp_0.INIT_RAM_00 =
    256'h00A0000000000000B00A000000000000B00A000000000000B00
    A0000000000000B;
    defparam bram_sp_0.INIT_RAM_01 =
    256'h00A0000000000000B00A000000000000B00A000000000000B00
    A0000000000000B;
    defparam bram_sp_0.INIT_RAM_3F =
    256'h00A0000000000000B00A000000000000B00A000000000000B00
    A0000000000000B;

```

**Vhdl 例化:**

```

    COMPONENT SP
        GENERIC(
            BIT_WIDTH:integer:=32;
            READ_MODE:bit:= '0';
            WRITE_MODE:bit_vector:= "01";
            BLK_SEL:bit_vector:= "000";
            RESET_MODE:string:= "SYNC";
            INIT_RAM_00:bit_vector:=X"00A0000000000000B
00A0000000000000B00A000000000000B00A000000000000B ";
            INIT_RAM_01:bit_vector:=X"00A0000000000000B
00A0000000000000B00A0000000000000B00A0000000000000B ";
            INIT_RAM_3F:bit_vector:=X"00A0000000000000B
00A0000000000000B00A0000000000000B00A0000000000000B "
        );
        PORT(
            DO:OUT std_logic_vector(31 downto 0):=conv_
std_logic_vector(0,32);
            CLK,CE,OCE,RESET,WRE:IN std_logic;
            AD:IN std_logic_vector(13 downto 0);
            BLKSEL:IN std_logic_vector(2 downto 0);
            DI:IN std_logic_vector(31 downto 0)
        );
    END COMPONENT;
    uut:SP
        GENERIC MAP(
            BIT_WIDTH=>32,
            READ_MODE=>'0',
            WRITE_MODE=>"01",
            BLK_SEL=>"000",
            RESET_MODE=>"SYNC",
            INIT_RAM_00=>X"00A0000000000000B00A00
00000000000000B00A000000000000B00A0000000000000B ",

```

```

                INIT_RAM_01=>X"00A00000000000B00A00
0000000000B00A000000000000B00A000000000000B ",
                INIT_RAM_02=>X"00A00000000000B00A00
0000000000B00A000000000000B00A000000000000B ",
                INIT_RAM_3F=>X"00A00000000000B00A00
0000000000B00A000000000000B00A000000000000B "
            )
        PORT MAP (
            DO=>dout,
            CLK=>clk,
            OCE=>oce,
            CE=>ce,
            RESET=>reset,
            WRE=>wre,
            BLKSEL=>blksel,
            AD=>ad,
            DI=>din
        );

```

示例二

#### Verilog 例化:

```

SPX9 bram_spx9_0 (
    .DO({dout[35:18],dout[17:0]}),
    .CLK(clk),
    .OCE(oce),
    .CE(ce),
    .RESET(reset),
    .WRE(wre),
    .BLKSEL({3'b000}),
    .AD({ad[9:0], 2'b00, byte_en[1:0]}),
    .DI({{18{1'b0}},din[17:0]})
);
defparam bram_spx9_0.READ_MODE = 1'b0;
defparam bram_spx9_0.WRITE_MODE = 2'b00;
defparam bram_spx9_0.BIT_WIDTH = 18;
defparam bram_spx9_0.BLK_SEL = 3'b000;
defparam bram_spx9_0.RESET_MODE = "SYNC";
defparam bram_spx9_0.INIT_RAM_00 =
288'h000000000C00000000000D0000050000C00000000000D000
000000C000000000000D0;
defparam bram_spx9_0.INIT_RAM_01 =
288'h000000000C00000000000D0000000000C000000003000D000
000000C000000000040D0;
defparam bram_spx9_0.INIT_RAM_3F =
288'h0000A0000C00000000000D0000000000C00000000000D001
000000C000000000000D0;

```

#### Vhdl 例化:

```

COMPONENT SPX9
    GENERIC(
        BIT_WIDTH:integer:=9;

```



```

        READ_MODE:bit:='0';
        WRITE_MODE:bit_vector:="00";
        BLK_SEL : bit_vector:="000";
        RESET_MODE : string:="SYNC";
        INIT_RAM_00:bit_vector:=X"000000000C000000
000000D0000050000C000000000000D0000000000C00000000000D0";
        INIT_RAM_01:bit_vector:=X"000000000C000000
000000D0000000000C000000003000D0000000000C000000000040D0";
        INIT_RAM_3F:bit_vector:=X"0000A0000C000000
000000D0000000000C000000000000D0010000000C00000000000D0"
    );
    PORT(
        DO:OUT std_logic_vector(35 downto 0):=conv_
std_logic_vector(0,36);
        CLK,CE,OCE,RESET,WRE:IN std_logic;
        AD:IN std_logic_vector(13 downto 0);
        DI:IN std_logic_vector(35 downto 0);
        BLKSEL:std_logic_vector(2 downto 0)
    );
END COMPONENT;
uut:SPX9
    GENERIC MAP(
        BIT_WIDTH=>9,
        READ_MODE=>'0',
        WRITE_MODE=>"00",
        BLK_SEL=>"000",
        RESET_MODE=>"SYNC",
        INIT_RAM_00=>X"00000000000000000000",
        INIT_RAM_01=>X"00000000000000000000",
        INIT_RAM_3F=>X"00000000000000000000"
    )
    PORT MAP(
        DO=>dout,
        CLK=>clk,
        OCE=>oce,
        CE=>ce,
        RESET=>reset,
        WRE=>wre,
        BLKSEL=>blkssel,
        AD=>ad,
        DI=>din
    );

```

## 4.2 SDP/SDPX9

### 原语介绍

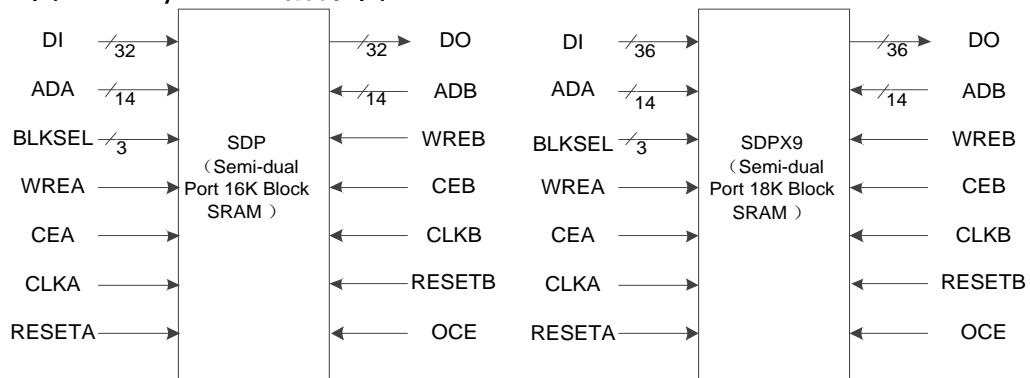
SDP/SDPX9(Semi Dual Port 16K Block SRAM /Semi Dual Port 18K

Block SRAM ),16K/18K 半双端口 BSRAM 的存储空间分别为 16K bit/18K bit, 其工作模式为半双端口模式, 端口 A 进行写操作, 端口 B 进行读操作, 可支持 2 种读模式 (bypass 模式和 pipeline 模式) 和 1 种写模式 (normal 模式)。

SDP 配置为 16bit/32bit、SDPX9 配置为 18bit/36bit 时, 可实现 BSRAM 的 byte enable 功能, 即通过写入地址端口 AD 的低四位控制写入 memory 的数据, 高电平使能。ADA[0]控制 DI[7:0]/DI[8:0]是否写入 memory, ADA[1]控制 DI[15:8]/DI[17:9]是否写入 memory, ADA[2]控制 DI[23:16]/DI[26:18]是否写入 memory, ADA[3]控制 DI[31:24]/DI[35:27]是否写入 memory。

### 结构框图

图 4-2 SDP/SDPX9 结构框图



### Port 介绍

表 4-4 Port 介绍

Port Name	I/O	Description
DO[31:0]/DO[35:0]	Output	Data Output
DI[31:0]/DI[35:0]	Input	Data Input
ADA[13:0]	Input	Port A Address
ADB[13:0]	Input	Port B Address
WREA	Input	Port A Write Enable
WREB	Input	PortB Write Enable
CEA	Input	Port A Clock Enable
CEB	Input	Port B Clock Enable
CLKA	Input	Port A Clock Input
CLKB	Input	Port B Clock Input
RESETA	Input	Port A Reset Input
RESETB	Input	PortB Reset Input
OCE	Input	Output Clock Enable
BLKSEL[2:0]	Input	Block SRAM Selection Input

## Attribute 介绍

表 4-5 Attribute 介绍

Attribute Name	Type	Allowed Values	Default	Description
READ_MODE	Integer	1'b0,1'b1	1'b0	Output pipeline register can be bypassed 1'b0:bypass mode 1'b1:pipeline mode
BIT_WIDTH_0	Integer	SDP:1,2,4,8,16,32 SDPX9:9,18,36	SDP:32 SDPX9:36	Port A's data width can be configured
BIT_WIDTH_1	Integer	SDP:1,2,4,8,16,32 SDPX9:9,18,36	SDP:32 SDPX9:36	Port B's data width can be configured
BLK_SEL	Integer	3'b000~3'b111	3'b000	Block SRAM Selection
RESET_MODE	String	SYNC,ASYN	SYNC	Reset mode config, synchronous or asynchronous
INIT_RAM_00~ INIT_RAM_3F	Integer	SDP:256'h0...0~256'h1...1 SDPX9:288'h0...0~288'h1...1	SDP:256'h0...0 SDPX9:288'h0...0	Initial value for initial BSRAM

## 配置关系

表 4-6 配置关系

Single Port Mode	BSRAM Capacity	Data Width	Address Depth
SDP	16K	1	14
		2	13
		4	12
		8	11
		16	10
		32	9
SDPX9	18K	9	11
		18	10
		36	9

## 原语例化

示例一

## Verilog 例化:

```
SDP bram_sdp_0 (
    .DO({dout[31:16],dout[15:0]}),
    .CLKA(clka),
    .CEA(cea),
    .RESETA(reseta),
    .WREA(wrea),
    .CLKB(clkb),
    .CEB(ceb),
    .RESETB(resetb),
    .WREB(wreb),
```

```

.OCE(oce),
.BLKSEL({3'b000}),
.ADA({ada[9:0], 2'b00, byte_en[1:0]}),
.DI({16{1'b0}},din[15:0]),
.ADB({adb[9:0],4'b0000})
);
defparam bram_sdp_0.READ_MODE = 1'b1;
defparam bram_sdp_0.BIT_WIDTH_0 = 16;
defparam bram_sdp_0.BIT_WIDTH_1 = 16;
defparam bram_sdp_0.BLK_SEL = 3'b000;
defparam bram_sdp_0.RESET_MODE = "SYNC";
defparam bram_sdp_0.INIT_RAM_00 =
256'h00A0000000000000B00A000000000000B00A000000000000B00
A0000000000000B;
defparam bram_sdp_0.INIT_RAM_3F =
256'h00A0000000000000B00A000000000000B00A000000000000B00
A0000000000000B;

```

**Vhdl 例化:**

```

COMPONENT SDP
  GENERIC(
    BIT_WIDTH_0:integer:=16;
    BIT_WIDTH_1:integer:=16;
    READ_MODE:bit:='0';
    BLK_SEL:bit_vector:="000";
    RESET_MODE:string:"SYNC";
    INIT_RAM_00:bit_vector:=X"00A0000000000000
B00A000000000000B00A000000000000B00A000000000000B";
    INIT_RAM_01:bit_vector:=X"00A0000000000000
B00A000000000000B00A000000000000B00A000000000000B";
    INIT_RAM_3F:bit_vector:=X"00A0000000000000
B00A000000000000B00A000000000000B00A000000000000B"
  );
  PORT(
    DO:OUT std_logic_vector(31 downto 0):=conv_
std_logic_vector(0,32);
    CLKA,CLKB,CEA,CEB,OCE,RESETA,RESETB,
WREA,WREB:IN std_logic;
    ADA,ADB:IN std_logic_vector(13 downto 0);
    BLKSEL:IN std_logic_vector(2 downto 0);
    DI:IN std_logic_vector(31 downto 0)
  );
END COMPONENT;
 uut:SDP
  GENERIC MAP(
    BIT_WIDTH_0=>16,
    BIT_WIDTH_1=>16,
    READ_MODE=>'0',
    BLK_SEL=>"000",
    RESET_MODE=>"SYNC",
    INIT_RAM_00=>X"00A0000000000000B00A00

```

```

0000000000B00A00000000000000B00A000000000000B",
        INIT_RAM_01=>X"00A0000000000000B00A00
0000000000B00A00000000000000B00A000000000000B",
        INIT_RAM_3F=>X"00A0000000000000B00A00
0000000000B00A00000000000000B00A000000000000B"
    )
    PORT MAP(
        DO=>dout,
        CLKA=>clka,
        CEA=>cea,
        RESETA=>reseta,
        WREA=>wrea,
        CLKB=>clkb,
        CEB=>ceb,
        RESETB=>resetb,
        WREB=>wreb,
        OCE=>oce,
        BLKSEL=>blksel,
        ADA=>ada,
        DI=>din,
        ADB=>adb
    );

```

示例二

**Verilog 例化:**

```

SDPX9 bram_sdp_x9_0 (
    .DO({dout[35:9],dout[8:0]}),
    .CLKA(clka),
    .CEA(cea),
    .RESETA(reseta),
    .WREA(wrea),
    .CLKB(clkb),
    .CEB(ceb),
    .RESETB(resetb),
    .WREB(wreb),
    .OCE(oce),
    .BLKSEL({3'b000}),
    .ADA({ada[10:0],3'b000}),
    .DI({{27{1'b0}},din[8:0]}),
    .ADB({adb[10:0],3'b000})
);
defparam bram_sdp_x9_0.READ_MODE = 1'b0;
defparam bram_sdp_x9_0.BIT_WIDTH_0 = 9;
defparam bram_sdp_x9_0.BIT_WIDTH_1 = 9;
defparam bram_sdp_x9_0.BLK_SEL = 3'b000;
defparam bram_sdp_x9_0.RESET_MODE = "SYNC";
defparam bram_sdp_x9_0.INIT_RAM_00 =
288'h000000000C00000000000000D0000050000C000000000000D000
0000000C000000000000D0;

```

```

defparam bram_sdp9_0.INIT_RAM_01 =
288'h00000000C00000000000D000000000C00000003000D000
000000C00000000040D0;
defparam bram_sdp9_0.INIT_RAM_3F =
288'h0000A000C00000000000D000000000C0000000000D001
000000C00000000000D0;

```

**Vhdl 例化:**

```

COMPONENT SDPX9
  GENERIC(
    BIT_WIDTH_0:integer:=18;
    BIT_WIDTH_1:integer:=18;
    READ_MODE:bit:='0';
    BLK_SEL:bit_vector:="000";
    RESET_MODE:string:="SYNC";
    INIT_RAM_00:bit_vector:=X"00000000C00000
000000D0000050000C0000000000D000000000C0000000000D0"
;
    INIT_RAM_01:bit_vector:=X"00000000C00000
000000D0000000000C00000003000D000000000C000000000040D0"
;
    INIT_RAM_3F:bit_vector:=X"0000A000C00000
000000D0000000000C0000000000D001000000C00000000000D0"
);
  PORT(
    DO:OUT std_logic_vector(35 downto 0):=conv
_std_logic_vector(0,36);
    CLKA,CLKB,CEA,CEB,OCE,RESETA,RESETB,
WREA,WREB:IN std_logic;
    ADA,ADB:IN std_logic_vector(13 downto 0);
    BLKSEL:IN std_logic_vector(2 downto 0);
    DI:IN std_logic_vector(35 downto 0)
);
END COMPONENT;
 uut:SDP
  GENERIC MAP(
    BIT_WIDTH_0=>18,
    BIT_WIDTH_1=>18,
    READ_MODE=>'0',
    BLK_SEL=>"000",
    RESET_MODE=>"SYNC",
    INIT_RAM_00=>X"00000000C00000000000D00
00050000C00000000000D000000000C00000000000D0",
    INIT_RAM_01=>X"00000000C00000000000D00
00000000C00000003000D000000000C000000000040D0",
    INIT_RAM_3F=>X"0000A000C00000000000D00
00000000C00000000000D001000000C00000000000D0"
)
  PORT MAP(
    DO=>dout,
    CLKA=>clka,

```

```

CEA=>cea,
RESETA=>reseta,
WREA=>wrea,
CLKB=>clkb,
CEB=>ceb,
RESETB=>resetb,
WREB=>wreb,
OCE=>oce,
BLKSEL=>blkssel,
ADA=>ada,
DI=>din,
ADB=>adb
);

```

## 4.3 DP/DPX9

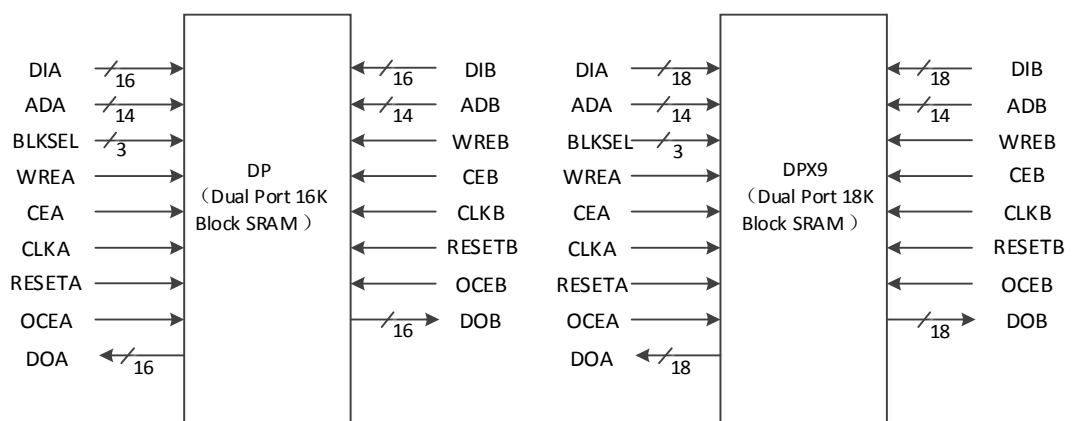
### 原语介绍

DP/DPX9(True Dual Port 16K Block SRAM/True Dual Port 18K Block SRAM), 16K/18K 双端口 BSRAM 的存储空间分别为 16K bit/18K bit, 其工作模式为双端口模式, 端口 A 和端口 B 均可分别独立实现读/写操作, 可支持 2 种读模式 (bypass 模式和 pipeline 模式) 和 3 种写模式 (normal 模式、write-through 模式和 read-before-write 模式)。

DP 配置为 16bit、DPX9 配置为 18bit 时, 可实现 BSRAM 的 byte enable 功能, 即通过写入地址端口 AD 的低四位控制写入 memory 的数据, 高电平使能。ADA[0]控制 DIA[7:0]/DIA[8:0]是否写入 memory, ADA[1]控制 DIA[15:8]/DIA[17:9]是否写入 memory, ADB[0]控制 DIB[7:0]/DIB[8:0]是否写入 memory, ADB[1]控制 DIB[15:8]/DIB[17:9]是否写入 memory。

### 结构框图

图 4-3 DP/DPX9 结构框图



## Port 介绍

表 4-7 Port 介绍

Port Name	I/O	Description
DOA[15:0]/DOA[17:0]	Output	Port A Data Output
DOB[15:0]/DOB[17:0]	Output	Port B Data Output
DIA[15:0]/DIA[17:0]	Input	Port A Data Input
DIB[15:0]/DIB[17:0]	Input	Port B Data Input
ADA[13:0]	Input	Port A Address
ADB[13:0]	Input	Port B Address
WREA	Input	Port A Write Enable
WREB	Input	Port B Write Enable
CEA	Input	Port A Clock Enable
CEB	Input	Port B Clock Enable
CLKA	Input	Port A Clock Input
CLKB	Input	Port B Clock Input
RESETA	Input	Port A Reset Input
RESETB	Input	Port B Reset Input
OCEA	Input	Port A Output Clock Enable
OCEB	Input	Port B Output Clock Enable
BLKSEL[2:0]	Input	Block RAM Selection



## Attribute 介绍

表 4-8 Attribute 介绍

Attribute Name	Type	Allowed Values	Default	Description
READ_MODE0	Integer	1'b0,1'b1	1'b0	Port A's output pipeline register can be bypassed 1'b0:bypass mode 1'b1:pipeline mode
READ_MODE1	Integer	1'b0,1'b1	1'b0	Port B's output pipeline register can be bypassed 1'b0:bypass mode 1'b1:pipeline mode
WRITE_MODE0	Integer	2'b00,2'b01,2'b10	2'b00	Port A's write mode can be selected 2'b00: normal mode 2'b01: write-through mode 2'b10: read-before-write mode
WRITE_MODE1	Integer	2'b00,2'b01,2'b10	2'b00	Port B's write mode can be selected 2'b00: normal mode 2'b01: write-through mode 2'b10: read-before-write mode
BIT_WIDTH_0	Integer	DP:1,2,4,8,16 DPX9:9,18	DP:16 DPX9:18	Port A's data width can be configured
BIT_WIDTH_1	Integer	DP:1,2,4,8,16 DPX9:9,18	DP:16 DP:18	Port B's data width can be configured
BLK_SEL	Integer	3'b000~3'b111	3'b000	Block RAM Selection
RESET_MODE	String	SYNC,ASYNC	SYNC	Reset mode config, synchronous or asynchronous
INIT_RAM_00~ INIT_RAM_3F	Integer	DP:256'h0...0~256' h1...1 DPX9:288'h0...0~2 88'h1...1	DP:256'h0... 0 DPX9:288'h0 ...0	Initial value for initial BSRAM

## 配置关系

表 4-9 配置关系

Single Port Mode	BSRAM Capacity	Data Width	Address Depth
DP	16K	1	14
		2	13
		4	12
		8	11
		16	10
DPX9	18K	9	11
		18	10

## 原语例化

示例一

**Verilog 例化:**

```

DP bram_dp_0 (
  .DOA({doa[15:8],doa[7:0]}),
  .DOB({doa[15:8],dob[7:0]}),
  .CLKA(clka),
  .OCEA(ocea),
  .CEA(cea),
  .RESETA(reseta),
  .WREA(wrea),
  .CLKB(clkb),
  .OCEB(oceb),
  .CEB(ceb),
  .RESETB(resetb),
  .WREB(wreb),
  .BLKSEL({3'b000}),
  .ADA({ada[10:0],3'b000}),
  .DIA({{8{1'b0}},dia[7:0]})
  .ADB({adb[10:0],3'b000}),
  .DIB({{8{1'b0}},dib[7:0]})
);
defparam bram_dp_0.READ_MODE0 = 1'b0;
defparam bram_dp_0.READ_MODE1 = 1'b0;
defparam bram_dp_0.WRITE_MODE0 = 2'b00;
defparam bram_dp_0.WRITE_MODE1 = 2'b00;
defparam bram_dp_0.BIT_WIDTH_0 = 8;
defparam bram_dp_0.BIT_WIDTH_1 = 8;
defparam bram_dp_0.BLK_SEL = 3'b000;
defparam bram_dp_0.RESET_MODE = "SYNC";
defparam bram_dp_0.INIT_RAM_00 =
256'h00A0000000000000B00A000000000000B00A000000000000B00A00
0000000000B;
defparam bram_dp_0.INIT_RAM_3E =
256'h00A0000000000000B00A000000000000B00A000000000000B00A00
0000000000B;
defparam bram_dp_0.INIT_RAM_3F =
256'h00A0000000000000B00A000000000000B00A000000000000B00A00
0000000000B;

```

**Vhdl 例化:**

```

COMPONENT DP
  GENERIC (
    BIT_WIDTH_0:integer:=16;
    BIT_WIDTH_1:integer:=16;
    READ_MODE0:bit:='0';
    READ_MODE1:bit:='0';
    WRITE_MODE0:bit_vector:="00";
    WRITE_MODE1:bit_vector:="00";
    BLK_SEL:bit_vector:="000";
    RESET_MODE:string:="SYNC";
    INIT_RAM_00:bit_vector:=X"0000000000000000
0000000000000000000000000000000000000000000000000";

```

```

        INIT_RAM_01:bit_vector:=X"0000000000000000
000000000000000000000000000000000000000000";
        INIT_RAM_3F:bit_vector:=X"0000000000000000
000000000000000000000000000000000000000000";
    );
    PORT (
        DOA,DOB:OUT std_logic_vector(15 downto 0):
=conv_std_logic_vector(0,16);
        CLKA,CLKB,CEA,CEB,OCEA,OCEB,RESETA,
RESETB,WREA,WREB:IN std_logic;
        ADA,ADB:IN std_logic_vector(13 downto 0);
        BLKSEL:IN std_logic_vector(2 downto 0);
        DIA,DIB:IN std_logic_vector(15 downto 0)
    );
END COMPONENT;
uut:DP
    GENERIC MAP(
        BIT_WIDTH_0=>16,
        BIT_WIDTH_1=>16,
        READ_MODE0=>'0',
        READ_MODE1=>'0',
        WRITE_MODE0=>"00",
        WRITE_MODE1=>"00",
        BLK_SEL=>"000",
        RESET_MODE=>"SYNC",
        INIT_RAM_00=>X"000000000000000000000000",
        INIT_RAM_01=>X"000000000000000000000000",
        INIT_RAM_3F=>X"000000000000000000000000"
    )
    PORT MAP(
        DOA=>doa,
        DOB=>dob,
        CLKA=>clka,
        CLKB=>clkb,
        CEA=>ceb,
        CEB=>ceb,
        OCEA=>ocea,
        OCEB=>oceb,
        RESETA=>reseta,
        RESETB=>resetb,
        WREA=>wrea,
        WREB=>wreb,
        ADA=>ada,
        ADB=>adb,
        BLKSEL=>blkssel,
        DIA=>dia,
        DIB=>dib
    )

```

```
);
```

示例二

**Verilog 例化:**

```
DPX9 bram_dpx9_0 (
    .DOA(doa[17:0]),
    .DOB(dob[17:0]),
    .CLKA(clka),
    .OCEA(ocea),
    .CEA(cea),
    .RESETA(reseta),
    .WREA(wrea),
    .CLKB(clkb),
    .OCEB(oceb),
    .CEB(ceb),
    .RESETB(resetb),
    .WREB(wreb),
    .BLKSEL({3'b000}),
    .ADA({ada[9:0], 2'b00,byte_ena[1:0]}),
    .DIA(dia[17:0]),
    .ADB({adb[9:0], 2'b00,byte_enb[1:0]}),
    .DIB(dib[17:0])
);
defparam bram_dpx9_0.READ_MODE0 = 1'b1;
defparam bram_dpx9_0.READ_MODE1 = 1'b1;
defparam bram_dpx9_0.WRITE_MODE0 = 2'b01;
defparam bram_dpx9_0.WRITE_MODE1 = 2'b01;
defparam bram_dpx9_0.BIT_WIDTH_0 = 18;
defparam bram_dpx9_0.BIT_WIDTH_1 = 18;
defparam bram_dpx9_0.BLK_SEL = 3'b000;
defparam bram_dpx9_0.RESET_MODE = "SYNC";
defparam bram_dpx9_0.INIT_RAM_00 =
288'h000000000C0000000000D000000000C00000000000D000
0000000C000000000000D0;
defparam bram_dpx9_0.INIT_RAM_01 =
288'h000000000C0000000000D000000000C00000000000D000
0000000C000000000000D0;
defparam bram_dpx9_0.INIT_RAM_3F =
288'h000000000C0000000000D000000000C00000000000D000
0000000C000000000000D0;
```

**Vhdl 例化:**

```
COMPONENT DPX9
    GENERIC (
        BIT_WIDTH_0:integer:=18;
        BIT_WIDTH_1:integer:=18;
        READ_MODE0:bit:= '0';
        READ_MODE1:bit:= '0';
        WRITE_MODE0:bit_vector:= "00";
        WRITE_MODE1:bit_vector:= "00";
```

```

        BLK_SEL:bit_vector:="000";
        RESET_MODE:string:="SYNC";
        INIT_RAM_00:bit_vector:=X"00000000000000
0000000000000000000000000000000000000000";
        INIT_RAM_01:bit_vector:=X"00000000000000
0000000000000000000000000000000000000000";
        INIT_RAM_3F:bit_vector:=X"00000000000000
0000000000000000000000000000000000000000"
    );
    PORT (
        DOA,DOB:OUT std_logic_vector(17 downto 0)
:=conv_std_logic_vector(0,18);
        CLKA,CLKB,CEA,CEB,OCEA,OCEB,RESETA,
RESETB,WREA,WREB:IN std_logic;
        ADA,ADB:IN std_logic_vector(13 downto 0);
        BLKSEL:IN std_logic_vector(2 downto 0);
        DIA:IN std_logic_vector(17 downto 0);
        DIB:IN std_logic_vector(17 downto 0)
    );
END COMPONENT;
 uut:DPX9
    GENERIC MAP(
        BIT_WIDTH_0=>18,
        BIT_WIDTH_1=>18,
        READ_MODE0=>'0',
        READ_MODE1=>'0',
        WRITE_MODE0=>"00",
        WRITE_MODE1=>"00",
        BLK_SEL=>"000",
        RESET_MODE=>"SYNC",
        INIT_RAM_00=>X"000000000000000000000000
0000000000000000000000000000000000000000",
        INIT_RAM_01=>X"000000000000000000000000
0000000000000000000000000000000000000000",
        INIT_RAM_3F=>X"000000000000000000000000
0000000000000000000000000000000000000000"
    )
    PORT MAP(
        DOA=>doa,
        DOB=>dob,
        CLKA=>clka,
        CLKB=>clkb,
        CEA=>ceb,
        CEB=>ceb,
        OCEA=>ocea,
        OCEB=>oceb,
        RESETA=>reseta,
        RESETB=>resetb,
        WREA=>wrea,
        WREB=>wreb,

```

```

ADA=>ada,
ADB=>adb,
BLKSEL=>blkssel,
DIA=>dia,
DIB=>dib

```

);

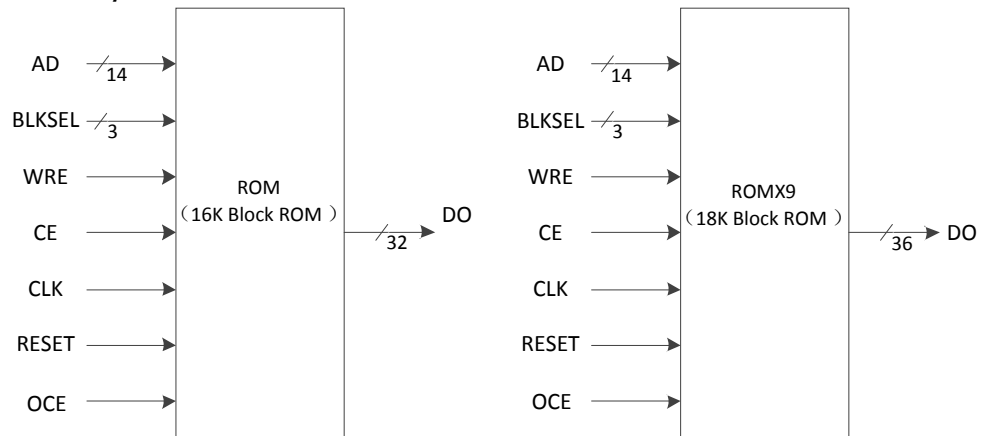
## 4.4 ROM/ROMX9

### 原语介绍

ROM/ROMX9(16K/18K Block ROM),16K/18K 块状只读储存器的存储空间分别为 16K bit/18K bit, 其工作模式为只读模式, 可支持 2 种读模式 (bypass 模式和 pipeline 模式)。

### 结构框图

图 4-4 ROM/ROMX9 结构框图



### Port 介绍

表 4-10 Port 介绍

Port Name	I/O	Description
DO[31:0]/DO[35:0]	Output	Data Output
AD[13:0]	Input	Address Input
WRE	Input	Write Enable
CE	Input	Clock Enable
CLK	Input	Clock Input
RESET	Input	Reset Input
OCE	Input	Output Clock Enable
BLKSEL[2:0]	Input	Block RAM Select

## Attribute 介绍

表 4-11 Attribute 介绍

Attribute Name	Type	Allowed Values	Default	Description
READ_MODE	Integer	1'b0,1'b1	1'b0	Output pipeline register can be bypassed 1'b0:bypass mode 1'b1:pipeline mode
BIT_WIDTH	Integer	ROM:1,2,4,8,16,32 ROMX9:9,18,36	ROM:32 ROMX9:36	Data width can be configured
BLK_SEL	Integer	3'b000~3'b111	3'b000	Block RAM Selection
RESET_MODE	String	SYNC,ASYNC	SYNC	Reset mode config, synchronous or asynchronous
INIT_RAM_00~ INIT_RAM_3F	Integer	ROM:256'h0...0~256'h1...1 ROMX9:288'h0...0~288'h1...1	ROM:256'h0...0 ROMX9:288'h0...0	Initial value for initial BSRAM

## 配置关系

表 4-12 配置关系

Single Port Mode	BSRAM Capacity	Data Width	Address Depth
ROM	16K	1	14
		2	13
		4	12
		8	11
		16	10
		32	9
ROMX9	18K	9	11
		18	10
		36	9

## 原语例化

示例一

### Verilog 例化:

```

ROM bram_rom_0 (
    .DO({dout[31:8],dout[7:0]}),
    .CLK(clk),
    .OCE(oce),
    .CE(ce),
    .RESET(reset),
    .WRE(wre),
    .BLKSEL({3'b000}),
    .AD({ad[10:0],3'b000})
);
defparam bram_rom_0.READ_MODE = 1'b0;

```

```

defparam bram_rom_0.BIT_WIDTH = 8;
defparam bram_rom_0.BLK_SEL = 3'b000;
defparam bram_rom_0.RESET_MODE = "SYNC";
defparam bram_rom_0.INIT_RAM_00 =
256'h9C23645D0F78986FFC3E36E141541B95C19F2F7164085E631
A819860D8FF0000;
defparam bram_rom_0.INIT_RAM_01 =
256'h000000000000000000000000000000000000000000000000
000FFFFFFBDCF;

```

**Vhdl 例化:**

```

COMPONENT ROM
  GENERIC(
    BIT_WIDTH:integer:=1;
    READ_MODE:bit:='0';
    BLK_SEL:bit_vector:="000";
    RESET_MODE:string:"SYNC";
    INIT_RAM_00:bit_vector:=X"9C23645D0F78986FF
C3E36E141541B95C19F2F7164085E631A819860D8FF0000";
    INIT_RAM_01:bit_vector:=X"000000000000000000
00000000000000000000000000000000FFFFFFBDCF"
  );
  PORT(
    DO:OUT std_logic_vector(31 downto 0):=conv_std
_logic_vector(0,32);
    CLK,CE,OCE,RESET,WRE:IN std_logic;
    BLKSEL:IN std_logic_vector(2 downto 0);
    AD:IN std_logic_vector(13 downto 0)
  );
END COMPONENT;
 uut:ROM
  GENERIC MAP(
    BIT_WIDTH=>1,
    READ_MODE=>'0',
    BLK_SEL=>"000",
    RESET_MODE=>"SYNC",
    INIT_RAM_00=>X"9C23645D0F78986FFC3E36
E141541B95C19F2F7164085E631A819860D8FF0000",
    INIT_RAM_01=>X"000000000000000000000000000000
00000000000000000000000000000000FFFFFFBDCF "
  )
  PORT MAP(
    DO=>do,
    AD=>ad,
    CLK=>clk,
    CE=>ce,
    OCE=>oce,
    RESET=>reset,
    WRE=>wre,
    BLKSEL=>blkssel
  );

```



## 示例二

**Verilog 例化:**

```

ROMX9 bram_romx9_0 (
    .DO({dout[35:9],dout[8:0]}),
    .CLK(clk),
    .OCE(oce),
    .CE(ce),
    .RESET(reset),
    .WRE(wre),
    .BLKSEL({3'b000}),
    .AD({ad[10:0],3'b000})
);
defparam bram_romx9_0.READ_MODE = 1'b0;
defparam bram_romx9_0.BIT_WIDTH = 9;
defparam bram_romx9_0.BLK_SEL = 3'b000;
defparam bram_romx9_0.RESET_MODE = "SYNC";
defparam bram_romx9_0.INIT_RAM_00 =
288'hCE08CC85D07DE1316FFE0F86DE1A09523795E0E7E5E71B2
020BC630D6053160EC7FC0000;
defparam bram_romx9_0.INIT_RAM_01 =
288'h0000000000000000000000000000000000000000000000000000000000000000
000000001FFFFFFFF7ACF;

```

**Vhdl 例化:**

```

COMPONENT ROMX9
    GENERIC(
        BIT_WIDTH:integer:=9;
        READ_MODE:bit:= '0';
        BLK_SEL:bit_vector:="000";
        RESET_MODE:string:"SYNC";
        INIT_RAM_00:bit_vector:=X"CE08CC85D07DE131
6FFE0F86DE1A09523795E0E7E5E71B2020BC630D6053160EC7FC000
0";
        INIT_RAM_01:bit_vector:=X"000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
000000001FFFFFFFF7ACF"
    );
    PORT(
        DO:OUT std_logic_vector(35 downto 0):=conv_std
_logic_vector(0,36);
        CLK,CE,OCE,RESET,WRE:IN std_logic;
        BLKSEL:IN std_logic_vector(2 downto 0);
        AD:IN std_logic_vector(13 downto 0)
    );
END COMPONENT;
Uut:ROMX9
    GENERIC MAP(
        BIT_WIDTH=>9,
        READ_MODE=>'0',
        BLK_SEL=>"000",

```



# 5 DSP

DSP(Digital Signal Processing) 是数字信号处理，包含预加器 (Pre-Adder)，乘法器 (MULT) 和 54 位算术逻辑单元 (ALU54D)。

支持器件：GW1N-2、GW1N-2B、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55。

## 5.1 Pre-adder

Pre-adder 是预加器，实现预加、预减和移位功能。Pre-adder 按照位宽分为两种，分别是 9 位位宽的 PADD9 和 18 位位宽的 PADD18。

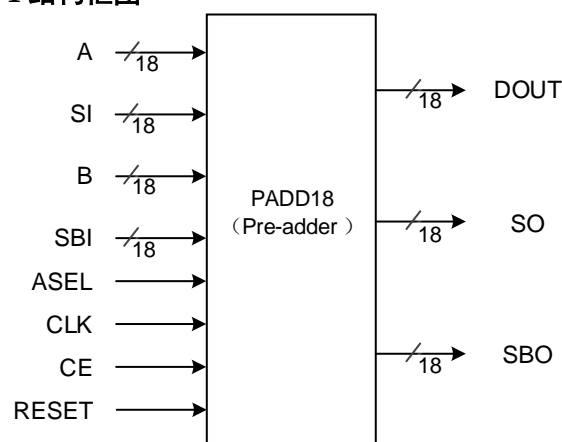
### 5.1.1 PADD18

#### 原语介绍

PADD18 (18-bit Pre-Adder) 是 18 位预加器，实现了 18 位的预加、预减或移位功能。

#### 结构框图

图 5-1 结构框图



## Port 介绍

表 5-1 Port 介绍

Port Name	I/O	Description
A[17:0]	Input	18-bit Data Input A
B[17:0]	Input	18-bit Data Input B
SI[17:0]	Input	Shift Data Input A
SBI[17:0]	Input	Pre-adder Shift Input, backward direction
ASEL	Input	Source Selection, SI or A
CLK	Input	Clock Input
CE	Input	Clock Enable
RESET	Input	Reset Input
SO[17:0]	Output	Shift Data Output A
SBO[17:0]	Output	Pre-adder Shift Output, backward direction
DOUT[17:0]	Output	Data Output

## Attribute 介绍

表 5-2 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
AREG	1'b0,1'b1	1'b0	Input A(A or SI)register can be bypassed 1'b0: bypass mode 1'b1: registered mode
BREG	1'b0,1'b1	1'b0	Input B(B or SBI ) register can be bypassed 1'b0: bypass mode 1'b1: registered mode
ADD_SUB	1'b0,1'b1	1'b0	ADD/SUB Selection 1'b0: add 1'b1: sub
PADD_RESET_MODE	SYNC,ASYNC	SYNC	Reset mode config,synchronous or asynchronous
BSEL_MODE	1'b1,1'b0	1'b1	Input B Selection. 1'b1: select SBI 1'b0: select B
SOREG	1'b0,1'b1	1'b0	Shift output register at port SO can be bypassed 1'b0: bypass mode 1'b1: registered mode

## 原语例化

### Verilog 例化:

```
PADD18 padd18_inst(
    .A(a[17:0]),
    .B(b[17:0]),
    .SO(so[17:0]),
    .SBO(sbo[17:0]),
    .DOUT(dout[17:0]),
    .SI(si[17:0]),
```

```

        .SBI(sbi[17:0]),
        .CE(ce),
        .CLK(clk),
        .RESET(reset),
        .ASEL(asel)
    );
    defparam padd18_inst.AREG = 1'b0;
    defparam padd18_inst.BREG = 1'b0;
    defparam padd18_inst.ADD_SUB = 1'b0;
    defparam padd18_inst.PADD_RESET_MODE = "SYNC";
    defparam padd18_inst.SOREG = 1'b0;
    defparam padd18_inst.BSEL_MODE = 1'b1;
Vhdl 例化:
    COMPONENT PADD18
        GENERIC (AREG:bit='0';
                BREG:bit='0';
                SOREG:bit='0';
                ADD_SUB:bit='0';
                PADD_RESET_MODE:string="SYNC" ;
                BSEL_MODE:bit='1'
        );
        PORT(
            A:IN std_logic_vector(17 downto 0);
            B:IN std_logic_vector(17 downto 0);
            ASEL:IN std_logic;
            CE:IN std_logic;
            CLK:IN std_logic;
            RESET:IN std_logic;
            SI:IN std_logic_vector(17 downto 0);
            SBI:IN std_logic_vector(17 downto 0);
            SO:OUT std_logic_vector(17 downto 0);
            SBO:OUT std_logic_vector(17 downto 0);
            DOUT:OUT std_logic_vector(17 downto 0)
        );
    END COMPONENT;
    uut:PADD18
        GENERIC MAP (AREG=>'0',
                    BREG=>'0',
                    SOREG=>'0',
                    ADD_SUB=>'0',
                    PADD_RESET_MODE=>"SYNC",
                    BSEL_MODE=>'1'
        )
        PORT MAP (
            A=>a,
            B=>b,
            ASEL=>asel,
            CE=>ce,
            CLK=>clk,
            RESET=>reset,

```

```

SI=>si,
SBI=>sbi,
SO=>so,
SBO=>sbo,
DOUT=>dout
);

```

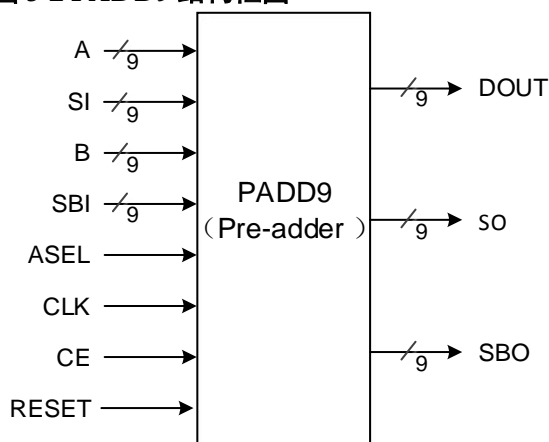
## 5.1.2 PADD9

### 原语介绍

PADD9 (9-bit Pre-Adder) 是 9 位预加器，实现了 9 位的预加、预减或移位功能。

### 结构框图

图 5-2 PADD9 结构框图



### Port 介绍

表 5-3 Port 介绍

Port Name	I/O	Description
A[8:0]	Input	9-bit Data Input A
B[8:0]	Input	9-bit Data Input B
SI[8:0]	Input	Shift Data Input A
SBI[8:0]	Input	Pre-adder Shift Input, backward direction
ASEL	Input	Source Selection, SI or A
CLK	Input	Clock input
CE	Input	Clock Enable
RESET	Input	Reset Input
SO[8:0]	Output	Shift Data Output A
SBO[8:0]	Output	Pre-adder Shift Output, backward direction
DOUT[8:0]	Output	Data Output

## Attribute 介绍

表 5-4 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
AREG	1'b0,1'b1	1'b0	Input A(A or SI) register can be bypassed 1'b0: bypass mode 1'b1: registered mode
BREG	1'b0,1'b1	1'b0	Input B(B or SBI ) register can be bypassed 1'b0: bypass mode 1'b1: registered mode
ADD_SUB	1'b0,1'b1	1'b0	ADD/SUB Selection 1'b0: add 1'b1: sub
PADD_RESET_M ODE	SYNC,ASYNC	SYNC	Reset mode config,synchronous or asynchronous
BSEL_MODE	1'b1,1'b0	1'b1	Input B Selection. 1'b1: select SBI 1'b0: select B
SOREG	1'b0,1'b1	1'b0	Shift output register at port SO can be bypassed 1'b0: bypass mode 1'b1: registered mode

## 原语例化

## Verilog 例化:

```

PADD9 padd9_inst(
    .A(a[8:0]),
    .B(b[8:0]),
    .SO(so[8:0]),
    .SBO(sbo[8:0]),
    .DOUT(dout[8:0]),
    .SI(si[8:0]),
    .SBI(sbi[8:0]),
    .CE(ce),
    .CLK(clk),
    .RESET(reset),
    .ASEL(asel)
);
defparam padd9_inst.AREG = 1'b0;
defparam padd9_inst.BREG = 1'b0;
defparam padd9_inst.ADD_SUB = 1'b0;
defparam padd9_inst.PADD_RESET_MODE = "SYNC";
defparam padd9_inst.SOREG = 1'b0;
defparam padd9_inst.BSEL_MODE = 1'b1;

```

## Vhdl 例化:

```

COMPONENT PADD9
    GENERIC (AREG:bit='0';
            BREG:bit='0';
            SOREG:bit='0';
            ADD_SUB:bit='0';

```

```

        PADD_RESET_MODE:string:="SYNC" ;
        BSEL_MODE:bit:='1'
    );
    PORT(
        A:IN std_logic_vector(8 downto 0);
        B:IN std_logic_vector(8 downto 0);
        ASEL:IN std_logic;
        CE:IN std_logic;
        CLK:IN std_logic;
        RESET:IN std_logic;
        SI:IN std_logic_vector(8 downto 0);
        SBI:IN std_logic_vector(8 downto 0);
        SO:OUT std_logic_vector(8 downto 0);
        SBO:OUT std_logic_vector(8 downto 0);
        DOUT:OUT std_logic_vector(8 downto 0)
    );
END COMPONENT;
 uut:PADD9
    GENERIC MAP (AREG=>'0',
                 BREG=>'0',
                 SOREG=>'0',
                 ADD_SUB=>'0',
                 PADD_RESET_MODE=>"SYNC",
                 BSEL_MODE=>'1'
    )
    PORT MAP (
        A=>a,
        B=>b,
        ASEL=>asel,
        CE=>ce,
        CLK=>clk,
        RESET=>reset,
        SI=>si,
        SBI=>sbi,
        SO=>so,
        SBO=>sbo,
        DOUT=>dout
    );

```

## 5.2 Multiplier

Multiplier 是 DSP 的乘法器单元，乘法器的乘数输入信号定义为 MDIA 和 MDIB，乘积输出信号定义为 MOUT，可实现乘法运算： $DOUT = A * B$ 。

Multiplier 根据数据位宽可配置成 9x9，18x18，36x36 等乘法器，分别对应原语 MULT9X9，MULT18X18，MULT36X36。



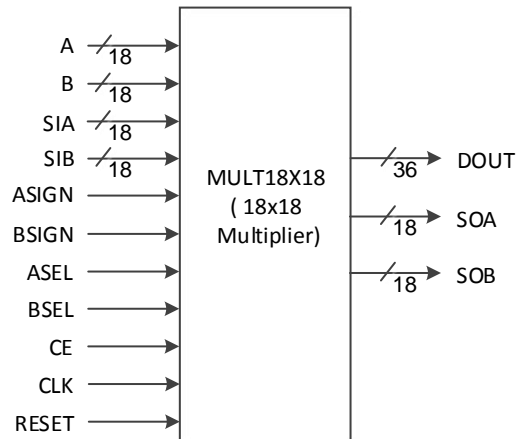
## 5.2.1 MULT18X18

### 原语介绍

MULT18X18 (18x18 Multiplier) 是 18x18 乘法器, 实现了 18 位乘法运算。

### 结构框图

图 5-3 MULT18X18 结构框图



### Port 介绍

表 5-5 Port 介绍

Port Name	I/O	Description
A[17:0]	Input	18-bit Data Input A
B[17:0]	Input	18-bit Data Input B
SIA[17:0]	Input	18-bit Shift Data Input A
SIB[17:0]	Input	18-bit Shift Data Input B
ASIGN	Input	Input A Sign Bit
BSIGN	Input	Input B Sign Bit
ASEL	Input	Source Selection, SIA or A
BSEL	Input	Source Selection, SIB or B
CLK	Input	Clock Input
CE	Input	Clock Enable
RESET	Input	Reset Input
DOUT[35:0]	Output	Multiplier Data Output
SOA[17:0]	Output	Multiplier Register Output A
SOB[17:0]	Output	Multiplier Register Output B

## Attribute 介绍

表 5-6 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
AREG	1'b0,1'b1	1'b0	Input A(SIA or A) register can be bypassed 1'b0:bypass mode 1'b1:registered mode
BREG	1'b0,1'b1	1'b0	Input B(SIB or B) register can be bypassed 1'b0:bypass mode 'b1:registered mode
OUT_REG	1'b0,1'b1	1'b0	Output register can be bypassed 1'b0:bypass mode 1'b1:registeredmode
PIPE_REG	1'b0,1'b1	1'b0	Pipeline register can be bypassed 1'b0:bypass mode 1'b1:registeredmode
ASIGN_REG	1'b0,1'b1	1'b0	ASIGN input register can be bypassed 1'b0:bypass mode 1'b1:registeredmode
BSIGN_REG	1'b0,1'b1	1'b0	BSIGN input register can be bypassed 1'b0:bypass mode 1'b1:registeredmode
SOA_REG	1'b0,1'b1	1'b0	SOA register can be bypassed 1'b0:bypassmode 1'b1:registered mode
MULT_RESET_MODE	SYNC,ASYNC	SYNC	Reset mode config, synchronous or asynchronous

## 原语例化

## Verilog 例化:

```

MULT18X18 uut(
    .DOUT(dout[35:0]),
    .SOA(sofar[17:0]),
    .SOB(sob[17:0]),
    .A(a[17:0]),
    .B(b[17:0]),
    .SIA(sia[17:0]),
    .SIB(sib[17:0]),
    .ASIGN(assign),
    .BSIGN(bsign),
    .ASEL(asel),
    .BSEL(bsel),
    .CE(ce),
    .CLK(clk),
    .RESET(reset)
);
defparam uut.AREG=1'b1;
defparam uut.BREG=1'b1;
defparam uut.OUT_REG=1'b1;
defparam uut.PIPE_REG=1'b0;
defparam uut.ASIGN_REG=1'b0;
defparam uut.BSIGN_REG=1'b0;

```

```

defparam uut.SOA_REG=1'b0;
defparam uut.MULT_RESET_MODE="ASYNC";
Vhdl 例化:
COMPONENT MULT18X18
  GENERIC (AREG:bit:='0';
           BREG:bit:='0';
           OUT_REG:bit:='0';
           PIPE_REG:bit:='0';
           ASIGN_REG:bit:='0';
           BSIGN_REG:bit:='0';
           SOA_REG:bit:='0';
           MULT_RESET_MODE:string:="SYNC"
  );
  PORT(
    A:IN std_logic_vector(17 downto 0);
    B:IN std_logic_vector(17 downto 0);
    SIA:IN std_logic_vector(17 downto 0);
    SIB:IN std_logic_vector(17 downto 0);
    ASIGN:IN std_logic;
    BSIGN:IN std_logic;
    ASEL:IN std_logic;
    BSEL:IN std_logic;
    CE:IN std_logic;
    CLK:IN std_logic;
    RESET:IN std_logic;
    SOA:OUT std_logic_vector(17 downto 0);
    SOB:OUT std_logic_vector(17 downto 0);
    DOUT:OUT std_logic_vector(35 downto 0)
  );
END COMPONENT;
uut:MULT18X18
  GENERIC MAP (AREG=>'1',
              BREG=>'1',
              OUT_REG=>'1',
              PIPE_REG=>'0',
              ASIGN_REG=>'0',
              BSIGN_REG=>'0',
              SOA_REG=>'0',
              MULT_RESET_MODE=>"ASYNC"
  )
  PORT MAP (
    A=>a,
    B=>b,
    SIA=>sia,
    SIB=>sib,
    ASIGN=>assign,
    BSIGN=>bsign,
    ASEL=>asel,
    BSEL=>bsel,
    CE=>ce,

```

```

CLK=>clk,
RESET=>reset,
SOA=>soa,
SOB=>sob,
DOUT=>dout

```

```
);
```

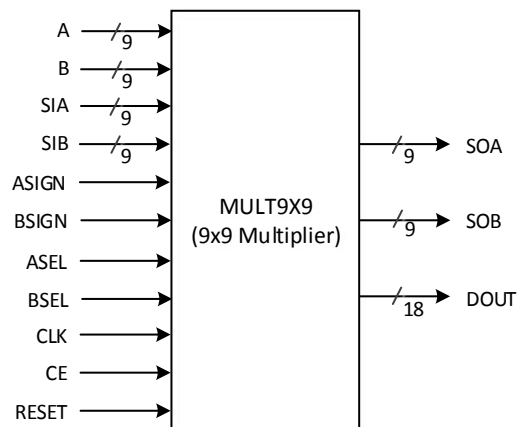
## 5.2.2 MULT9X9

### 原语介绍

MULT9X9 (9x9 Multiplier) 是 9x9 乘法器，实现了 9 位乘法运算。

### 结构框图

图 5-4 MULT9X9 结构框图



### Port 介绍

表 5-7 Port 介绍

Port Name	I/O	Description
A[8:0]	Input	9-bit Data Input A
B[8:0]	Input	9-bit Data Input B
SIA[8:0]	Input	9-bit Shift Data Input A
SIB[8:0]	Input	9-bit Shift Data Input B
ASIGN	Input	Input A Sign bit
BSIGN	Input	Input B Sign bit
ASEL	Input	Source Selection, SIA or A
BSEL	Input	Source Selection, SIB or B
CLK	Input	Clock Input
CE	Input	Clock Enable
RESET	Input	Reset Input
DOUT[17:0]	Output	Multiplier Data Output
SOA[8:0]	Output	Multiplier Register Output A
SOB[8:0]	Output	Multiplier Register Output B

## Attribute 介绍

表 5-8 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
AREG	1'b0,1'b1	1'b0	Input A(SIA or A) register can be bypassed 1'b0:bypass mode 1'b1:registered mode
BREG	1'b0,1'b1	1'b0	Input B(SIB or B) register can be bypassed 1'b0:bypass mode 1'b1:registered mode
OUT_REG	1'b0,1'b1	1'b0	Output register can be bypassed 1'b0:bypass mode 1'b1:registered mode
PIPE_REG	1'b0,1'b1	1'b0	Pipeline register can be bypassed 1'b0:bypass mode 1'b1:registered mode
ASIGN_REG	1'b0,1'b1	1'b0	ASIGN input register can be bypassed 1'b0:bypass mode 1'b1:registered mode
BSIGN_REG	1'b0,1'b1	1'b0	BSIGN input register can be bypassed 1'b0:bypass mode 1'b1:registered mode
SOA_REG	1'b0,1'b1	1'b0	SOA register can be bypassed 1'b0:bypassmode 1'b1:registered mode
MULT_RESET_MODE	SYNC, ASYNC	SYNC	Reset mode config, synchronous or asynchronous

## 原语例化

## Verilog 例化:

```

MULT9X9 uut(
    .DOUT(dout[17:0]),
    .SOA(sofar[8:0]),
    .SOB(sob[8:0]),
    .A(a[8:0]),
    .B(b[8:0]),
    .SIA(sia[8:0]),
    .SIB(sib[8:0]),
    .ASIGN(assign),
    .BSIGN(bsign),
    .ASEL(asel),
    .BSEL(bsel),
    .CE(ce),
    .CLK(clk),
    .RESET(reset)
);
defparam uut.AREG=1'b1;
defparam uut.BREG=1'b1;
defparam uut.OUT_REG=1'b1;
defparam uut.PIPE_REG=1'b0;
defparam uut.ASIGN_REG=1'b0;

```

```

defparam uut.BSIGN_REG=1'b0;
defparam uut.SOA_REG=1'b0;
defparam uut.MULT_RESET_MODE="ASYNC";

```

**Vhdl 例化:**

```

COMPONENT MULT9X9
  GENERIC (AREG:bit:= '0';
           BREG:bit:= '0';
           OUT_REG:bit:= '0';
           PIPE_REG:bit:= '0';
           ASIGN_REG:bit:= '0';
           BSIGN_REG:bit:= '0';
           SOA_REG:bit:= '0';
           MULT_RESET_MODE:string:= "SYNC"
  );
  PORT(
    A:IN std_logic_vector(8 downto 0);
    B:IN std_logic_vector(8 downto 0);
    SIA:IN std_logic_vector(8 downto 0);
    SIB:IN std_logic_vector(8 downto 0);
    ASIGN:IN std_logic;
    BSIGN:IN std_logic;
    ASEL:IN std_logic;
    BSEL:IN std_logic;
    CE:IN std_logic;
    CLK:IN std_logic;
    RESET:IN std_logic;
    SOA:OUT std_logic_vector(8 downto 0);
    SOB:OUT std_logic_vector(8 downto 0);
    DOUT:OUT std_logic_vector(17 downto 0)
  );
END COMPONENT;
uut:MULT9X9
  GENERIC MAP (AREG=>'1',
              BREG=>'1',
              OUT_REG=>'1',
              PIPE_REG=>'0',
              ASIGN_REG=>'0',
              BSIGN_REG=>'0',
              SOA_REG=>'0',
              MULT_RESET_MODE=>"ASYNC"
  )
  PORT MAP (
    A=>a,
    B=>b,
    SIA=>sia,
    SIB=>sib,
    ASIGN=>assign,
    BSIGN=>bsign,
    ASEL=>asel,
    BSEL=>bsel,

```

```
CE=>ce,  
CLK=>clk,  
RESET=>reset,  
SOA=>soa,  
SOB=>sob,  
DOUT=>dout  
);
```

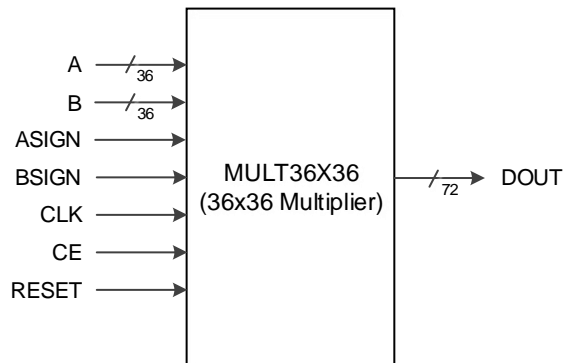
### 5.2.3 MULT36X36

#### 原语介绍

MULT36X36 (36x36 Multiplier) 是 36x36 乘法器，实现了 36 位的乘法运算。

#### 结构框图

图 5-5 MULT36X36 结构框图



## Port 介绍

表 5-9 Port 介绍

Port Name	I/O	Description
A[35:0]	Input	36-bit Data Input A
B[35:0]	Input	36-bit Data Input B
ASIGN	Input	Input A Sign bit
BSIGN	Input	Input B Sign bit
CLK	Input	Clock Input
CE	Input	Clock Enable
RESET	Input	Reset Input
DOUT[71:0]	Output	Multiplier Data Output

## Attribute 介绍

表 5-10 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
AREG	1'b0,1'b1	1'b0	Input A(A) register can be bypassed. 1'b0:bypass mode 1'b1:registered mode
BREG	1'b0,1'b1	1'b0	Input B(B) register can be bypassed. 1'b0:bypass mode 1'b1:registered mode
OUT0_REG	1'b0,1'b1	1'b0	The first output register can be bypassed 1'b0:bypass mode 1'b1:registered mode
OUT1_REG	1'b0,1'b1	1'b0	The second output register can be bypassed 1'b0:bypass mode 1'b1:registered mode
PIPE_REG	1'b0,1'b1	1'b0	Pipeline register can be bypassed 1'b0:bypass mode 1'b1:registered mode
ASIGN_REG	1'b0,1'b1	1'b0	ASIGN input register can be bypassed 1'b0:bypass mode 1'b1:registered mode
BSIGN_REG	1'b0,1'b1	1'b0	BSIGN input register can be bypassed 1'b0:bypass mode 1'b1:registered mode
MULT_RESET_MODE	SYNC,ASYNC	SYNC	Reset mode config,synchronous or asynchronous

## 原语例化

### Verilog 例化:

```
MULT36X36 uut(
    .DOUT(mout[71:0]),
    .A(mdia[35:0]),
    .B(mdib[35:0]),
    .ASIGN(assign),
    .BSIGN(bsign),
    .CE(ce),
```



```

        .CLK(clk),
        .RESET(reset)
    );
    defparam uut.AREG=1'b0;
    defparam uut.BREG=1'b0;
    defparam uut.OUT0_REG=1'b0;
    defparam uut.OUT1_REG=1'b1;
    defparam uut.PIPE_REG=1'b0;
    defparam uut.ASIGN_REG=1'b1;
    defparam uut.BSIGN_REG=1'b1;
    defparam uut.MULT_RESET_MODE="ASYNC";
Vhdl 例化:
    COMPONENT MULT36X36
        GENERIC (AREG:bit:='0';
                BREG:bit:='0';
                OUT0_REG:bit:='0';
                OUT1_REG:bit:='0';
                PIPE_REG:bit:='0';
                ASIGN_REG:bit:='0';
                BSIGN_REG:bit:='0';
                MULT_RESET_MODE:string:="SYNC"
        );
        PORT(
            A:IN std_logic_vector(35 downto 0);
            B:IN std_logic_vector(35 downto 0);
            ASIGN:IN std_logic;
            BSIGN:IN std_logic;
            CE:IN std_logic;
            CLK:IN std_logic;
            RESET:IN std_logic;
            DOUT:OUT std_logic_vector(71 downto 0)
        );
    END COMPONENT;
    uut:MULT36X36
        GENERIC MAP (AREG=>'0',
                    BREG=>'0',
                    OUT0_REG=>'0',
                    OUT1_REG=>'1',
                    PIPE_REG=>'0',
                    ASIGN_REG=>'1',
                    BSIGN_REG=>'1',
                    MULT_RESET_MODE=>"ASYNC"
        )
        PORT MAP (
            A=>mdia,
            B=>mdib,
            ASIGN=>assign,
            BSIGN=>bsign,
            CE=>ce,
            CLK=>clk,

```

```

        RESET=>reset,
        DOUT=>mout
    );

```

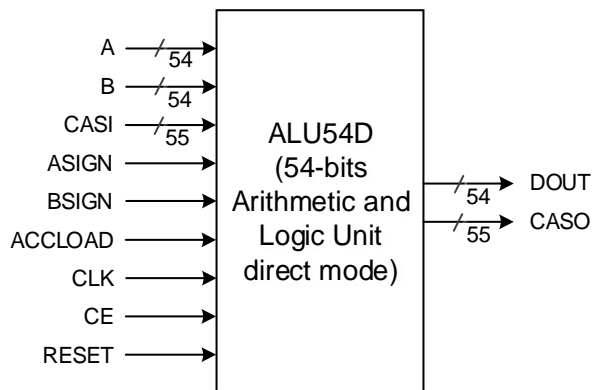
## 5.3 ALU54D

### 原语介绍

ALU54D（54-bit Arithmetic Logic Unit）是 54 位算术逻辑单元，实现 54 位的算术逻辑运算。

### 结构框图

图 5-6 ALU54D 结构框图



### Port 介绍

表 5-11 Port 介绍

Port Name	I/O	Description
A[53:0]	Input	54-bit Data Input A
B[53:0]	Input	54-bit Data Input B
CASI[54:0]	Input	55-bit Data Carry Input
ASIGN	Input	Input A Sign Bit
BSIGN	Input	Input B Sign Bit
ACCLOAD	Input	Accumulator Reload Mode Selection
CLK	Input	Clock Input
CE	Input	Clock Enable
RESET	Input	Reset Input
DOUT[53:0]	Output	ALU54D Data Output
CASO[54:0]	Output	55-bit Data Carry Output

## Attribute 介绍

表 5-12 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
AREG	1'b0,1'b1	1'b0	Input A(A) registers can be bypassed 1'b0:bypass mode 1'b1: registered mode
BREG	1'b0,1'b1	1'b0	Input B(B) registers can be bypassed 1'b0:bypass mode 1'b1: registered mode
ASIGN_REG	1'b0,1'b1	1'b0	ASIGN input register can be bypassed 1'b0:bypass mode 1'b1:registered mode
BSIGN_REG	1'b0,1'b1	1'b0	BSIGN input register can be bypassed 1'b0:bypass mode 1'b1:registered mode
ACCLOAD_REG	1'b0,1'b1	1'b0	Stage register of ACCLOAD can be bypassed 1'b0:bypass mode 1'b1:registered mode
OUT_REG	1'b0,1'b1	1'b0	The output registers can be bypassed. 1'b0:bypass mode 1'b1: registered mode
B_ADD_SUB	1'b0,1'b1	1'b0	B_OUT ADD/SUB Selection 1'b0: add 1'b1: sub
C_ADD_SUB	1'b0,1'b1	1'b0	C_OUT ADD/SUB Selection 1'b0: add 1'b1: sub
ALUMODE	0,1,2	0	ALU54 Operation Mode and Unit Input Selection 0:ACC/0 +/- B +/- A; 1:ACC/0 +/- B + CASI; 2:A +/- B + CASI;
ALU_RESET_MODE	SYNC,ASYNC	SYNC	Reset mode config, synchronous or asynchronous

## 原语例化

## Verilog 例化:

```

ALU54D alu54_inst (
    .A(a[53:0]),
    .B(b[53:0]),
    .CASI(casi[54:0]),
    .ASIGN(assign),
    .BSIGN(bsign),
    .ACCLOAD(acclload),
    .CE(ce),
    .CLK(clk),

```

```

        .RESET(reset),
        .DOUT(dout[53:0]),
        .CASO(caso[54:0])
    );
    defparam alu54_inst.AREG=1'b1;
    defparam alu54_inst.BREG=1'b1;
    defparam alu54_inst.ASIGN_REG=1'b0;
    defparam alu54_inst.BSIGN_REG=1'b0;
    defparam alu54_inst.ACCLOAD_REG=1'b1;
    defparam alu54_inst.OUT_REG=1'b0;
    defparam alu54_inst.B_ADD_SUB=1'b0;
    defparam alu54_inst.C_ADD_SUB=1'b0;
    defparam alu54_inst.ALUMODE=0;
    defparam alu54_inst.ALU_RESET_MODE="SYNC";
Vhdl 例化:
    COMPONENT ALU54D
        GENERIC (AREG:bit:=0';
                BREG:bit:=0';
                ASIGN_REG:bit:=0';
                BSIGN_REG:bit:=0';
                ACCLOAD_REG:bit:=0';
                OUT_REG:bit:=0';
                B_ADD_SUB:bit:=0';
                C_ADD_SUB:bit:=0';
                ALUD_MODE:integer:=0;
                ALU_RESET_MODE:string:="SYNC"
        );
        PORT(
            A:IN std_logic_vector(53 downto 0);
            B:IN std_logic_vector(53 downto 0);
            ASIGN:IN std_logic;
            BSIGN:IN std_logic;
            CE:IN std_logic;
            CLK:IN std_logic;
            RESET:IN std_logic;
            ACCLOAD:IN std_logic;
            CASI:IN std_logic_vector(54 downto 0);
            CASO:OUT std_logic_vector(54 downto 0);
            DOUT:OUT std_logic_vector(53 downto 0)
        );
    END COMPONENT;
    uut:ALU54D
        GENERIC MAP (AREG=>'1',
                    BREG=>'1',
                    ASIGN_REG=>'0',
                    BSIGN_REG=>'0',
                    ACCLOAD_REG=>'1',
                    OUT_REG=>'0',
                    B_ADD_SUB=>'0',
                    C_ADD_SUB=>'0',

```

```

        ALUD_MODE=>0,
        ALU_RESET_MODE=>"SYNC"
    )
    PORT MAP (
        A=>a,
        B=>b,
        ASIGN=>assign,
        BSIGN=>bsign,
        CE=>ce,
        CLK=>clk,
        RESET=>reset,
        ACCLOAD=>accload,
        CASI=>casi,
        CASO=>caso,
        DOUT=>dout
    );

```

## 5.4 MULTALU

MULTALU (Multiplier with ALU) 是带 ALU 功能的乘法器，分为 36X18 位和 18X18 位，分别对应原语 MULTALU36X18 和 MULTALU18X18。

MULTALU36X18 有三种运算模式：

$$DOUT = A * B \pm C$$

$$DOUT = \sum(A * B)$$

$$DOUT = A * B + CASI$$

MULTALU18X18 有三种运算模式：

$$DOUT = \sum(A * B) \pm C$$

$$DOUT = \sum(A * B) + CASI$$

$$DOUT = A * B \pm D + CASI$$

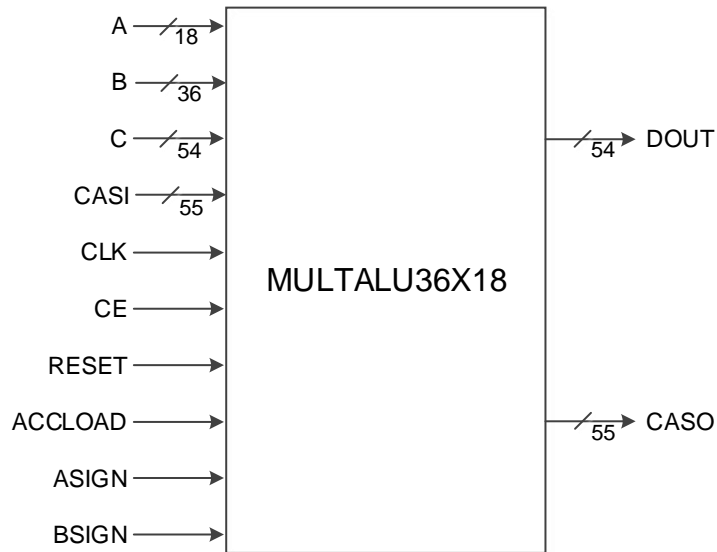
### 5.4.1 MULTALU36X18

#### 原语介绍

MULTALU36X18 (36x18 Multiplier with ALU) 是带 ALU 功能的 36X18 乘法器。

## 结构框图

图 5-7 MULTALU36X18 结构框图



## Port 介绍

表 5-13 Port 介绍

Port Name	I/O	Description
A[17:0]	Input	18-bit Data Input A
B[35:0]	Input	36-bit Data Input B
C[53:0]	Input	54-bit Reload Data Input
CASI[54:0]	Input	55-bit Data Carry Input
ASIGN	Input	Input A Sign Bit
BSIGN	Input	Input B Sign Bit
CLK	Input	Clock Input
CE	Input	Clock Enable
RESET	Input	Reset Input
ACCLoad	Input	Accumulator Reload Mode Selection
DOUT[53:0]	Output	Data Output
CASO[54:0]	Output	55-bit Data Carry Output

## Attribute 介绍

表 5-14 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
AREG	1'b0,1'b1	1'b0	Input A(A)register can be bypassed 1'b0:bypass mode 1'b1:registered mode
BREG	1'b0,1'b1	1'b0	Input B(B)register can be bypassed 1'b0:bypass mode 1'b1:registered mode
CREG	1'b0,1'b1	1'b0	Input C(C) register can be bypassed 1'b0:bypass mode 1'b1:registered mode
OUT_REG	1'b0,1'b1	1'b0	The output registers can be bypassed. 1'b0:bypass mode 1'b1: registered mode
PIPE_REG	1'b0,1'b1	1'b0	Pipeline register can be bypassed . 1'b0:bypass mode 1'b1:registered mode
ASIGN_REG	1'b0,1'b1	1'b0	ASIGN input register can be bypassed 1'b0:bypass mode 1'b1:registered mode
BSIGN_REG	1'b0,1'b1	1'b0	BSIGN input register can be bypassed. 1'b0:bypass mode 1'b1:registered mode
ACCLOAD_REG0	1'b0,1'b1	1'b0	The first stage register of ACCLOAD can be bypassed 1'b0:bypass mode 1'b1:registered mode
ACCLOAD_REG1	1'b0,1'b1	1'b0	The second stage register of ACCLOAD can be bypassed 1'b0:bypass mode 1'b1:registered mode
MULT_RESET_MODE	SYNC,ASYNC	SYNC	Reset mode config,synchronous or asynchronous
MULTALU36X18_MODE	0,1,2	0	MULTALU36X18 Operation Mode and Unit Input Selection 0:36x18 +/- C; 1:ACC/0 + 36x18; 2: 36x18 + CASI
C_ADD_SUB	1'b0,1'b1	1'b0	C_OUT ADD/SUB Selection 1'b0: add 1'b1: sub

## 原语例化

## Verilog 例化:

```
MULTALU36X18 multalu36x18_inst(
    .CASO(caso[54:0]),
    .DOUT(dout[53:0]),
    .ASIGN(assign),
    .BSIGN(bsign),
```

```

.CE(ce),
.CLK(clk),
.RESET(reset),
.CASI(casi[54:0]),
.ACCLOAD(accload),
.A(a[17:0]),
.B(b[35:0]),
.C(c[53:0])
);
defparam multalu36x18_inst.AREG = 1'b1;
defparam multalu36x18_inst.BREG = 1'b0;
defparam multalu36x18_inst.CREG = 1'b0;
defparam multalu36x18_inst.OUT_REG = 1'b1;
defparam multalu36x18_inst.PIPE_REG = 1'b0;
defparam multalu36x18_inst.ASIGN_REG = 1'b0;
defparam multalu36x18_inst.BSIGN_REG = 1'b0;
defparam multalu36x18_inst.ACCLOAD_REG0 = 1'b1;
defparam multalu36x18_inst.ACCLOAD_REG1 = 1'b0;
defparam multalu36x18_inst.SOA_REG = 1'b0;
defparam multalu36x18_inst.MULT_RESET_MODE = "SYNC";
defparam multalu36x18_inst.MULTALU36X18_MODE = 0;
defparam multalu36x18_inst.C_ADD_SUB = 1'b0;

```

**Vhdl 例化:**

```

COMPONENT MULTALU36X18
  GENERIC (AREG:bit:='0';
           BREG:bit:='0';
           CREG:bit:='0';
           OUT_REG:bit:='0';
           PIPE_REG:bit:='0';
           ASIGN_REG:bit:='0';
           BSIGN_REG:bit:='0';
           ACCLOAD_REG0:bit:='0';
           ACCLOAD_REG1:bit:='0';
           SOA_REG:bit:='0';
           MULTALU36X18_MODE:integer:=0;
           C_ADD_SUB:bit:='0';
           MULT_RESET_MODE:string:="SYNC"
  );
  PORT(
    A:IN std_logic_vector(17 downto 0);
    B:IN std_logic_vector(35 downto 0);
    C:IN std_logic_vector(53 downto 0);
    ASIGN:IN std_logic;
    BSIGN:IN std_logic;
    CE:IN std_logic;
    CLK:IN std_logic;
    RESET:IN std_logic;
    ACCLOAD:IN std_logic;
    CASI:IN std_logic_vector(54 downto 0);
    CASO:OUT std_logic_vector(54 downto 0);
  );

```



```
        DOUT:OUT std_logic_vector(53 downto 0)
    );
END COMPONENT;
uut:MULTALU36X18
    GENERIC MAP (AREG=>'1',
                BREG=>'0',
                CREG=>'0',
                OUT_REG=>'1',
                PIPE_REG=>'0',
                ASIGN_REG=>'0',
                BSIGN_REG=>'0',
                ACCLOAD_REG0=>'1',
                ACCLOAD_REG1=>'0',
                SOA_REG=>'0',
                MULTALU36X18_MODE=>0,
                C_ADD_SUB=>'0',
                MULT_RESET_MODE=>"SYNC"
    )
    PORT MAP (
        A=>a,
        B=>b,
        C=>c,
        ASIGN=>assign,
        BSIGN=>bsign,
        CE=>ce,
        CLK=>clk,
        RESET=>reset,
        ACCLOAD=>accload,
        CASI=>casi,
        CASO=>caso,
        DOUT=>dout
    );
```

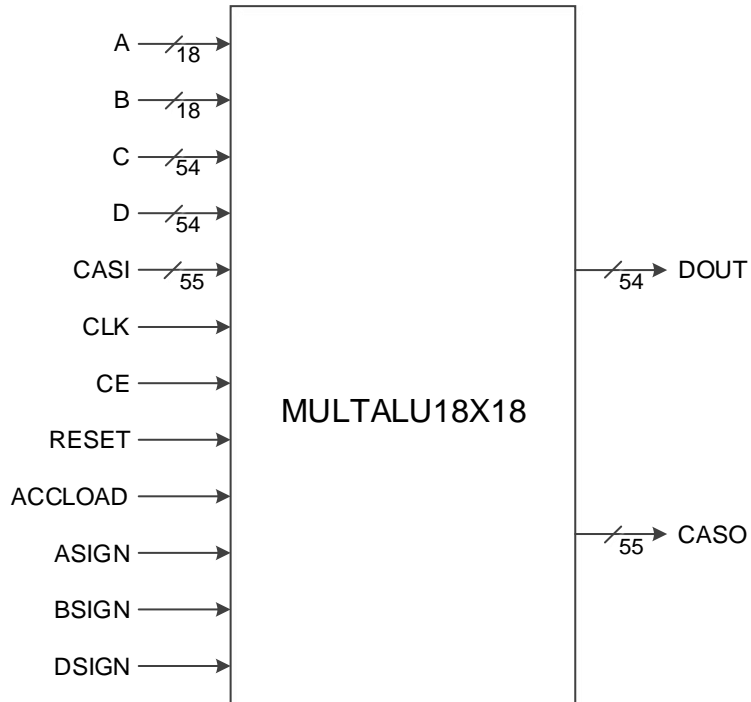
## 5.4.2 MULTALU18X18

### 原语介绍

MULTALU18X18（18x18 Multiplier with ALU）是带 ALU 功能的 18x18 乘法器。

### 结构框图

图 5-8 MULTALU18X18 结构框图



## Port 介绍

表 5-15 Port 介绍

Port Name	I/O	Description
A[17:0]	Input	18-bit Data Input A
B[17:0]	Input	18-bit Data Input B
C[53:0]	Input	54-bit Data Input C
D[53:0]	Input	54-bit Data Input D
CASI[54:0]	Input	55-bit Data Carry Input
ASIGN	Input	Input A Sign Bit
BSIGN	Input	Input B Sign Bit
DSIGN	Input	Input D Sign Bit
CLK	Input	Clock Input
CE	Input	Clock Enable
RESET	Input	Reset Input
ACCLOAD	Input	Accumulator Reload Mode selection
DOUT[53:0]	Output	Data Output
CASO[54:0]	Output	55-bit Data Carry Output

## Attribute 介绍

表 5-16 Attribute 介绍

Attribute Name	Allowed Values	Default Value	Description
AREG	1'b0,1'b1	1'b0	Input A(A)register can be bypassed 1'b0:bypass mode 1'b1:registered mode
BREG	1'b0,1'b1	1'b0	Input B(B)register can be bypassed 1'b0:bypass mode 1'b1:registered mode
CREG	1'b0,1'b1	1'b0	Input C(C) register can be bypassed 1'b0:bypass mode 1'b1:registered mode
DREG	1'b0,1'b1	1'b0	Input D(D) register can be bypassed 1'b0:bypass mode 1'b1:registered mode
DSIGN_REG	1'b0,1'b1	1'b0	DSIGN input register can be bypassed 1'b0:bypass mode 1'b1:registered mode
ASIGN_REG	1'b0,1'b1	1'b0	ASIGN input register can be bypassed 1'b0:bypass mode 1'b1:registered mode
BSIGN_REG	1'b0,1'b1	1'b0	BSIGN input register can be bypassed. 1'b0:bypass mode 1'b1:registered mode
ACCLOAD_REG0	1'b0,1'b1	1'b0	The first stage register of ACCLOAD can be bypassed 1'b0:bypass mode 1'b1:registered mode
ACCLOAD_REG1	1'b0,1'b1	1'b0	The second stage register of ACCLOAD can be bypassed 1'b0:bypass mode 1'b1:registered mode
MULT_RESET_MODE	SYNC,ASYNC	SYNC	Reset mode config,synchronous or asynchronous
PIPE_REG	1'b0,1'b1	1'b0	Pipeline register can be bypassed . 1'b0:bypass mode 1'b1:registered mode
OUT_REG	1'b0,1'b1	1'b0	The output registers can be bypassed. 1'b0:bypass mode 1'b1: registered mode
B_ADD_SUB	1'b0,1'b1	1'b0	B_OUT ADD/SUB Selection 1'b0: add 1'b1: sub
C_ADD_SUB	1'b0,1'b1	1'b0	C_OUT ADD/SUB Selection 1'b0: add 1'b1: sub
MULTALU18X18_MODE	0,1,2	0	MULTALU36X18 Operation Mode and Unit Input Selection 0:ACC/0 +/- 18x18 +/- C; 1:ACC/0 +/- 18x18 + CASI;

Attribute Name	Allowed Values	Default Value	Description
			2: 18x18 +/- D + CASI;

### 原语例化

#### Verilog 例化:

```
MULTALU18X18 multalu18x18_inst(
    .CASO(caso[54:0]),
    .DOUT(dout[53:0]),
    .ASIGN(assign),
    .BSIGN(bsign),
    .DSIGN(dsign),
    .CE(ce),
    .CLK(clk),
    .RESET(reset),
    .CASI(casi[54:0]),
    .ACCLOAD(accload),
    .A(a[17:0]),
    .B(b[17:0]),
    .C(c[53:0])
    .D(d[53:0])
);
defparam multalu18x18_inst.AREG = 1'b1;
defparam multalu18x18_inst.BREG = 1'b0;
defparam multalu18x18_inst.CREG = 1'b0;
defparam multalu18x18_inst.DREG = 1'b0;
defparam multalu18x18_inst.OUT_REG = 1'b1;
defparam multalu18x18_inst.PIPE_REG = 1'b0;
defparam multalu18x18_inst.ASIGN_REG = 1'b0;
defparam multalu18x18_inst.BSIGN_REG = 1'b0;
defparam multalu18x18_inst.DSIGN_REG = 1'b0;
defparam multalu18x18_inst.ACCLOAD_REG0 = 1'b1;
defparam multalu18x18_inst.ACCLOAD_REG1 = 1'b0;
defparam multalu18x18_inst.MULT_RESET_MODE = "SYNC";
defparam multalu18x18_inst.MULTALU18X18_MODE = 0;
defparam multalu18x18_inst.B_ADD_SUB = 1'b0;
defparam multalu18x18_inst.C_ADD_SUB = 1'b0;
```

#### Vhdl 例化:

```
COMPONENT MULTALU18X18
    GENERIC (AREG:bit='0';
            BREG:bit='0';
            CREG:bit='0';
            DREG:bit='0';
            OUT_REG:bit='0';
            PIPE_REG:bit='0';
            ASIGN_REG:bit='0';
            BSIGN_REG:bit='0';
            DSIGN_REG:bit='0';
            ACCLOAD_REG0:bit='0';
            ACCLOAD_REG1:bit='0');
```

```

        B_ADD_SUB:bit:='0';
        C_ADD_SUB:bit:='0';
        MULTALU18X18_MODE:integer:=0;
        MULT_RESET_MODE:string:="SYNC"
    );
    PORT(
        A:IN std_logic_vector(17 downto 0);
        B:IN std_logic_vector(17 downto 0);
        C:IN std_logic_vector(53 downto 0);
        D:IN std_logic_vector(53 downto 0);
        ASIGN:IN std_logic;
        BSIGN:IN std_logic;
        DSIGN:IN std_logic;
        CE:IN std_logic;
        CLK:IN std_logic;
        RESET:IN std_logic;
        ACCLOAD:IN std_logic;
        CASI:IN std_logic_vector(54 downto 0);
        CASO:OUT std_logic_vector(54 downto 0);
        DOUT:OUT std_logic_vector(53 downto 0)
    );
END COMPONENT;
uut:MULTALU18X18
    GENERIC MAP (AREG=>'1',
                BREG=>'0',
                CREG=>'0',
                DREG=>'0',
                OUT_REG=>'1',
                PIPE_REG=>'0',
                ASIGN_REG=>'0',
                BSIGN_REG=>'0',
                DSIGN_REG=>'0',
                ACCLOAD_REG0=>'1',
                ACCLOAD_REG1=>'0',
                B_ADD_SUB=>'0',
                C_ADD_SUB=>'0',
                MULTALU18X18_MODE=>0,
                MULT_RESET_MODE=>"SYNC"
    )
    PORT MAP (
        A=>a,
        B=>b,
        C=>c,
        D=>d,
        ASIGN=>assign,
        BSIGN=>bsign,
        DSIGN=>dsign,
        CE=>ce,
        CLK=>clk,
        RESET=>reset,

```

```

        ACCLOAD=>acclload,
        CASI=>casi,
        CASO=>caso,
        DOUT=>dout
    );

```

## 5.5 MULTADDALU

MULTADDALU (The Sum of Two Multipliers with ALU) 是带 ALU 功能的乘加器，实现乘法求和后累加或 reload 运算，对应的原语为 MULTADDALU18X18。

三种运算模式如下：

$$DOUT = A0 * B0 \pm A1 * B1 \pm C$$

$$DOUT = \sum (A0 * B0 \pm A1 * B1)$$

$$DOUT = A0 * B0 \pm A1 * B1 + CASI$$

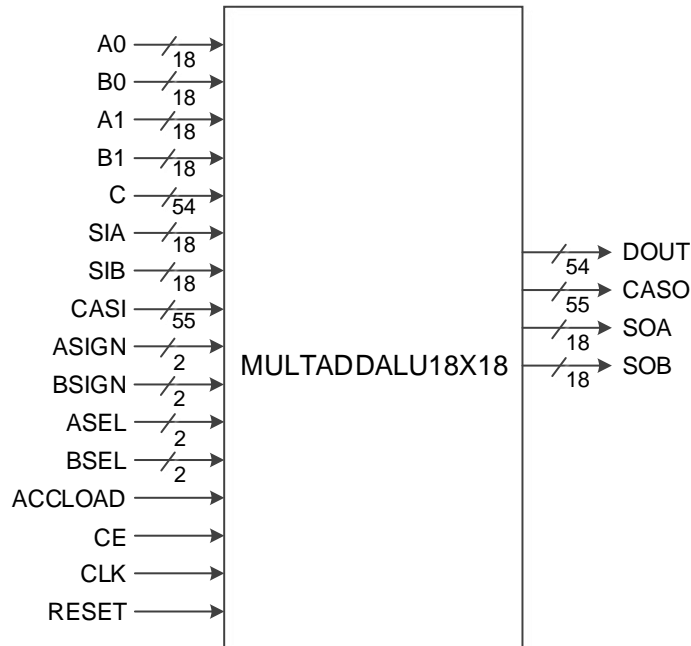
### 5.5.1 MULTADDALU18X18

#### 原语介绍

MULTADDALU18X18 (The Sum of Two 18x18 Multipliers with ALU) 是带 ALU 功能的 18x18 乘加器，实现 18 位的乘法求和后累加或 reload 运算。

#### 结构框图

图 5-9 MULTADDALU18X18 结构框图



## Port 介绍

表 5-17 Port 介绍

Port Name	I/O	Description
A0[17:0]	Input	18-bit Data Input A0
B0[17:0]	Input	18-bit Data Input B0
A1[17:0]	Input	18-bit Data Input A1
B1[17:0]	Input	18-bit Data Input B1
C[53:0]	Input	54-bit Reload Data Input
SIA[17:0]	Input	18-bit Shift Data Input A
SIB[17:0]	Input	18-bit Shift Data Input B
CASI[54:0]	Input	55-bit Data Carry Input
ASIGN[1:0]	Input	Input A0,A1 Sign bit
BSIGN[1:0]	Input	Input B0,B1 Sign bit
ASEL[1:0]	Input	Input A0,A1 Source Selection
BSEL[1:0]	Input	Input B0,B1 Source Selection
CLK	Input	Clock Input
CE	Input	Clock Enable
RESET	Input	Reset Input
ACCLOAD	Input	Accumulator Reload Mode Selection
DOUT[53:0]	Output	Data Output
CASO[54:0]	Output	55-bit Data Carry Output
SOA[17:0]	Output	Multiplier Register Output A
SOB[17:0]	Output	Multiplier Register Output B



## Attribute 介绍

表 5-18 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
A0REG	1'b0,1'b1	1'b0	Input A0(A0 or SIA) register can be bypassed. 1'b0:bypass mode 1'b1:registered mode
A1REG	1'b0,1'b1	1'b0	Input A1(A1 or Register Output A0) register can be bypassed. 1'b0:bypass mode 1'b1:registered mode
B0REG	1'b0,1'b1	1'b0	Input B0(B0 or SIB) register can be bypassed. 1'b0:bypass mode 1'b1:registered mode
B1REG	1'b0,1'b1	1'b0	Input B1(B1 or Register Output B0) register can be bypassed. 1'b0:bypass mode 1'b1:registered mode
CREG	1'b0,1'b1	1'b0	Input C(C) register can be bypassed 1'b0:bypass mode 1'b1:registered mode
PIPE0_REG	1'b0,1'b1	1'b0	Multiplier0 Pipeline register can be bypassed. 1'b0:bypass mode 1'b1:registered mode
PIPE1_REG	1'b0,1'b1	1'b0	Multiplier1 Pipeline register can be bypassed. 1'b0:bypass mode 1'b1:registered mode
OUT_REG	1'b0,1'b1	1'b0	Output register can be bypassed 1'b0:bypass mode 1'b1:registered mode
ASIGN0_REG	1'b0,1'b1	1'b0	ASIGN[0] input register can be bypassed. 1'b0:bypass mode 1'b1:registered mode
ASIGN1_REG	1'b0,1'b1	1'b0	ASIGN[1] input register can be bypassed. 1'b0:bypass mode 1'b1:registered mode
ACCLOAD_REG0	1'b0,1'b1	1'b0	The first stage register of ACCLOAD can be bypassed 1'b0:bypass mode 1'b1:registered mode
ACCLOAD_REG1	1'b0,1'b1	1'b0	The second stage register of ACCLOAD can be bypassed 1'b0:bypass mode 1'b1:registered mode
BSIGN0_REG	1'b0,1'b1	1'b0	BSIGN[0] input register can be bypassed. 1'b0:bypass mode 1'b1:registered mode
BSIGN1_REG	1'b0,1'b1	1'b0	BSIGN[1] input register can be bypassed. 1'b0:bypass mode 1'b1:registered mode
SOA_REG	1'b0,1'b1	1'b0	SOA register can be bypassed. 1'b0:bypassmode

Attribute Name	Allowed Values	Default	Description
			1'b1: registered mode
B_ADD_SUB	1'b0,1'b1	1'b0	B_OUT ADD/SUB Selection 1'b0: add 1'b1: sub
C_ADD_SUB	1'b0,1'b1	1'b0	C_OUT ADD/SUB Selection 1'b0: add 1'b1: sub
MULTADDALU18X18_MODE	0,1,2	0	MULTADDALU18X18 Operation Mode and Unit Input Selection 0: 18x18 +/- 18x18 +/- C; 1: ACC/0 + 18x18 +/- 18x18; 2: 18x18 +/- 18x18 + CASI
MULT_RESET_MODE	SYNC, ASYNC	SYNC	Reset mode config, synchronous or asynchronous

### 原语例化

#### Verilog 例化:

```

MULTADDALU18X18 uut(
    .DOUT(dout[53:0]),
    .CASO(caso[54:0]),
    .SOA(sofar[17:0]),
    .SOB(sob[17:0]),
    .A0(a0[17:0]),
    .B0(b0[17:0]),
    .A1(a1[17:0]),
    .B1(b1[17:0]),
    .C(c[53:0]),
    .SIA(sia[17:0]),
    .SIB(sib[17:0]),
    .CASI(casi[54:0]),
    .ACCLOAD(acclload),
    .ASEL(asel[1:0]),
    .BSEL(bsel[1:0]),
    .ASIGN(assign[1:0]),
    .BSIGN(bsign[1:0]),
    .CLK(clk),
    .CE(ce),
    .RESET(reset)
);
defparam uut.A0REG = 1'b0;
defparam uut.A1REG = 1'b0;
defparam uut.B0REG = 1'b0;
defparam uut.B1REG = 1'b0;
defparam uut.CREG = 1'b0;
defparam uut.PIPE0_REG = 1'b0;
defparam uut.PIPE1_REG = 1'b0;
defparam uut.OUT_REG = 1'b0;
defparam uut.ASIGN0_REG = 1'b0;
defparam uut.ASIGN1_REG = 1'b0;

```

```

defparam uut.ACCLOAD_REG0 = 1'b0;
defparam uut.ACCLOAD_REG1 = 1'b0;
defparam uut.BSIGN0_REG = 1'b0;
defparam uut.BSIGN1_REG = 1'b0;
defparam uut.SOA_REG = 1'b0;
defparam uut.B_ADD_SUB = 1'b0;
defparam uut.C_ADD_SUB = 1'b0;
defparam uut.MULTADDALU18X18_MODE = 0;
defparam uut.MULT_RESET_MODE = "SYNC";

```

**Vhdl 例化:**

```

COMPONENT MULTADDALU18X18
  GENERIC (A0REG:bit:='0';
           B0REG:bit:='0';
           A1REG:bit:='0';
           B1REG:bit:='0';
           CREG:bit:='0';
           OUT_REG:bit:='0';
           PIPE0_REG:bit:='0';
           PIPE1_REG:bit:='0';
           ASIGN0_REG:bit:='0';
           BSIGN0_REG:bit:='0';
           ASIGN1_REG:bit:='0';
           BSIGN1_REG:bit:='0';
           ACCLOAD_REG0:bit:='0';
           ACCLOAD_REG1:bit:='0';
           SOA_REG:bit:='0';
           B_ADD_SUB:bit:='0';
           C_ADD_SUB:bit:='0';
           MULTADDALU18X18_MODE:integer:=0;
           MULT_RESET_MODE:string:="SYNC"
  );
  PORT(
    A0:IN std_logic_vector(17 downto 0);
    A1:IN std_logic_vector(17 downto 0);
    B0:IN std_logic_vector(17 downto 0);
    B1:IN std_logic_vector(17 downto 0);
    SIA:IN std_logic_vector(17 downto 0);
    SIB:IN std_logic_vector(17 downto 0);
    C:IN std_logic_vector(53 downto 0);
    ASIGN:IN std_logic_vector(1 downto 0);
    BSIGN:IN std_logic_vector(1 downto 0);
    ASEL:IN std_logic_vector(1 downto 0);
    BSEL:IN std_logic_vector(1 downto 0);
    CE:IN std_logic;
    CLK:IN std_logic;
    RESET:IN std_logic;
    ACCLOAD:IN std_logic;
    CASI:IN std_logic_vector(54 downto 0);
    SOA:OUT std_logic_vector(17 downto 0);
    SOB:OUT std_logic_vector(17 downto 0);

```

```

        CASO:OUT std_logic_vector(54 downto 0);
        DOUT:OUT std_logic_vector(53 downto 0)
    );
END COMPONENT;
 uut:MULTADDALU18X18
    GENERIC MAP (A0REG=>'0',
                 B0REG=>'0',
                 A1REG=>'0',
                 B1REG=>'0',
                 CREG=>'0',
                 OUT_REG=>'0',
                 PIPE0_REG=>'0',
                 PIPE1_REG=>'0',
                 ASIGN0_REG=>'0',
                 BSIGN0_REG=>'0',
                 ASIGN1_REG=>'0',
                 BSIGN1_REG=>'0',
                 ACCLOAD_REG0=>'0',
                 ACCLOAD_REG1=>'0',
                 SOA_REG=>'0',
                 B_ADD_SUB=>'0',
                 C_ADD_SUB=>'0',
                 MULTADDALU18X18_MODE=>0,
                 MULT_RESET_MODE=>"SYNC"
    )
    PORT MAP (
        A0=>a0,
        A1=>a1,
        B0=>b0,
        B1=>b1,
        SIA=>sia,
        SIB=>sib,
        C=>c,
        ASIGN=>assign,
        BSIGN=>bsign,
        ASEL=>asel,
        BSEL=>bsel,
        CE=>ce,
        CLK=>clk,
        RESET=>reset,
        ACCLOAD=>accload,
        CASI=>casi,
        SOA=>soa,
        SOB=>sob,
        CASO=>caso,
        DOUT=>dout
    );

```

# 6Clock

## 6.1 PLL

### 原语介绍

高云 FPGA 提供了锁相环 (Phase\_Locked Loop)，可基于给定的输入时钟进行时钟相位调整、占空比调整、频率调整 (倍频和分频) 等来产生不同相位和频率的输出时钟。

PLL 的性能如下：

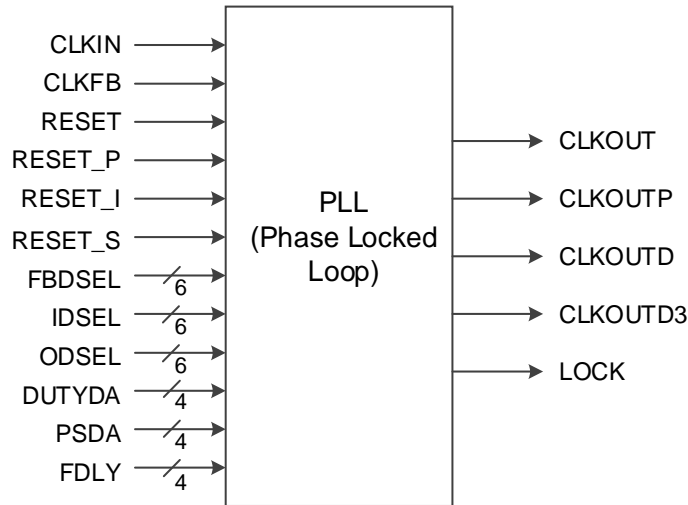
**表 6-1 PLL 性能**

	GW1N 家族	GW2A 家族
输入频率范围	3 MHz ~ 450MHz	3MHz ~ 500MHz
输入频率/输入分频倍数	3 MHz ~ 450MHz	3 MHz ~ 500MHz
VCO 震荡频率范围	400MHz ~ 900MHz	500MHz ~ 1300MHz
输出频率范围	3.125MHz ~ 450MHz	3.125MHz ~ 500MHz

支持器件：GW1N-1、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55。

## 结构框图

图 6-1 PLL 结构框图



## Port 介绍

表 6-2 Port 介绍

Port Name	I/O	Description
CLKIN	Input	Reference Clock Input
CLKFB	Input	Clock Feedback Input
RESET	Input	Reset Input
RESET_P	Input	Power Down Reset Input
RESET_I	Input	Reference Divider IDIV Reset Input
RESET_S	Input	Divider SDIV Reset Input
FBDSEL[5:0]	Input	Clock Frequency Multiplication Factor Dynamic Adjust
IDSEL[5:0]	Input	Clock Frequency Division Factor Dynamic Adjust
ODSEL[5:0]	Input	ODIV Frequency Division Factor Dynamic Adjust
DUTYDA[3:0]	Input	Duty Cycle Dynamic Adjust
PSDA[3:0]	Input	Phase Shift Dynamic Adjust
FDLY[3:0]	Input	Dynamic Fine Delay Adjust
CLKOUT	Output	Clock Output, No Phase Shift
LOCK	Output	PLL Lock Signal
CLKOUTP	Output	Clock Output With Phase Shift and Duty Cycle
CLKOUTD	Output	Clock Output, from CLKOUT and CLKOUTP, Divided by SDIV Divider
CLKOUTD3	Output	Clock Output, from CLKOUT and CLKOUTP, Divided by 3 Divider

## Attribute 介绍

表 6-3 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
FCLKIN	3~500	100	Reference Clock Frequency
IDIV_SEL	0~63	0	Clock Frequency Division Factor Static Adjust
DYN_IDIV_SEL	true,false	false	Clock Frequency Division Factor Dynamic/ Static Select false: Static true: Dynamic
FBDIV_SEL	0~63	0	Clock Frequency Multiplication Factor Static Adjust
DYN_FBDIV_SEL	true,false	false	Clock Frequency Multiplication Factor Dynamic/ Static Select false: Static true: Dynamic
ODIV_SEL	2,4,8,16,32,48,64,80,96, 112,128	8	ODIV Frequency Division Factor Static Adjust
DYN_ODIV_SEL	true,false	false	ODIV Frequency Division Factor Dynamic/ Static Select false: Static true: Dynamic
PSDA_SEL	0000~1111	0000	Phase Shift Static Adjust
DUTYDA_SEL	0010~1110	1000	Duty Cycle Static Adjust
DYN_DA_EN	true,false	false	Phase Shift and Duty Cycle Dynamic/ Static Select false: Static true: Dynamic
CLKOUT_FT_DIR	1'b0,1'b1	1'b1	CLKOUT Fine-tuning Direction Adjust Adjust 1'b0: sub 1'b1: add
CLKOUT_DLY_STEP	0,1,2,4	0	CLKOUT Fine-tuning Factor CLKOUT_DLY_STEP*delay(delay=50ps)
CLKOUTP_FT_DIR	1'b0,1'b1	1'b1	CLKOUTP Fine-tuning Direction Adjust Adjust 1'b0: sub 1'b1: add
CLKOUTP_DLY_STEP	0,1,2	0	CLKOUTP Fine-tuning Factor CLKOUTP_DLY_STEP*delay(delay=50ps)
DYN_SDIV_SEL	2~128	2	SDIV Frequency Division Factor
CLKFB_SEL	internal,external	internal	CLKFB Source Select internal: Internal Feedback CLKOUT external: Internal Feedback Signal
CLKOUTD_SRC	CLKOUT,CLKOUTP	CLKOUT	CLKOUTD source Select
CLKOUTD3_SRC	CLKOUT,CLKOUTP	CLKOUT	CLKOUTD3 source Select
CLKOUT_BYPASS	true,false	false	CLKIN Bypass PLL and Directly Drive the CLKOUT true: CLKIN Bypass to CLKOUT false: Normal Operation
CLKOUTP_BYPASS	true,false	false	CLKIN Bypass PLL and Directly Drive the CLKOUTP true: CLKIN Bypass to CLKOUTP false: Normal Operation

Attribute Name	Allowed Values	Default	Description
CLKOUTD_BYPASS	true,false	false	CLKIN Bypass PLL and Directly Drive the CLKOUTD true: CLKIN Bypass to CLKOUTD false: Normal Operation
DEVICE	GW1N-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55。	GW1N-2	Device Select

### 原语例化

#### Verilog 例化:

```

PLL pll_inst(
    .CLKOUT(clkout),
    .LOCK(lock),
    .CLKOUTP(clkoutp),
    .CLKOUTD(clkoutd),
    .CLKOUTD3(clkoutd3),
    .RESET(reset),
    .RESET_P(reset_p),
    .RESET_I(reset_i),
    .RESET_S(reset_s),
    .CLKIN(clkin),
    .CLKFB(clkfb),
    .FBDSEL(fbdsel),
    .IDSEL(idsel),
    .ODSEL(odsel),
    .PSDA(psda),
    .DUTYDA(dutyda),
    .FDLY(fdly)
);
defparam pll_inst.FCLKIN = "50";
defparam pll_inst.DYN_IDIV_SEL = "false";
defparam pll_inst.IDIV_SEL = 0;
defparam pll_inst.DYN_FBDIV_SEL = "false";
defparam pll_inst.FBDIV_SEL = 1;
defparam pll_inst.ODIV_SEL = 8;
defparam pll_inst.PSDA_SEL = "0100";
defparam pll_inst.DYN_DA_EN = "false";
defparam pll_inst.DUTYDA_SEL = "1000";
defparam pll_inst.CLKOUT_FT_DIR = 1'b1;
defparam pll_inst.CLKOUTP_FT_DIR = 1'b1;
defparam pll_inst.CLKOUT_DLY_STEP = 0;
defparam pll_inst.CLKOUTP_DLY_STEP = 0;

```



```

defparam pll_inst.CLKFB_SEL ="external";
defparam pll_inst.CLKOUT_BYPASS = "false";
defparam pll_inst.CLKOUTP_BYPASS = "false";
defparam pll_inst.CLKOUTD_BYPASS = "false";
defparam pll_inst.DYN_SDIV_SEL = 2;
defparam pll_inst.CLKOUTD_SRC = "CLKOUT";
defparam pll_inst.CLKOUTD3_SRC = "CLKOUT";
defparam pll_inst.DEVICE = "GW1N-4";

```

**Vhdl 例化:**

```

COMPONENT PLL
  GENERIC(
    FCLKIN:STRING:= "100.0";
    DEVICE:STRING:= "GW2A-18";
    DYN_IDIV_SEL:STRING:="false";
    IDIV_SEL:integer:=0;
    DYN_FBDIV_SEL:STRING:="false";
    FBDIV_SEL:integer:=0;
    DYN_ODIV_SEL:STRING:="false";
    ODIV_SEL:integer:=8;
    PSDA_SEL:STRING:="0000";
    DYN_DA_EN:STRING:="false";
    DUTYDA_SEL:STRING:="1000";
    CLKOUT_FT_DIR:bit:='1';
    CLKOUTP_FT_DIR:bit:='1';
    CLKOUT_DLY_STEP:integer:=0;
    CLKOUTP_DLY_STEP:integer:=0;
    CLKOUTD3_SRC:STRING:="CLKOUT";
    CLKFB_SEL : STRING:="internal";
    CLKOUT_BYPASS:STRING:="false";
    CLKOUTP_BYPASS:STRING:="false";
    CLKOUTD_BYPASS:STRING:="false";
    CLKOUTD_SRC:STRING:="CLKOUT";
    DYN_SDIV_SEL:integer:=2
  );
  PORT(
    CLKIN:IN std_logic;
    CLKFB:IN std_logic;
    IDSEL:IN std_logic_vector(5 downto 0);
    FBDSEL:IN std_logic_vector(5 downto 0);
    ODSEL:IN std_logic_vector(5 downto 0);
    RESET:IN std_logic;
    RESET_P:IN std_logic;

```

```

        RESET_I:IN std_logic;
        RESET_S:IN std_logic;
        PSDA,FDLY:IN std_logic_vector(3 downto 0);
        DUTYDA:IN std_logic_vector(3 downto 0);
        LOCK:OUT std_logic;
        CLKOUT:OUT std_logic;
        CLKOUTD:OUT std_logic;
        CLKOUTP:OUT std_logic;
        CLKOUTD3:OUT std_logic
    );
END COMPONENT;
 uut:PLL
    GENERIC MAP(
        FCLKIN =>"100.0",
        DEVICE =>"GW2A-18",
        DYN_IDIV_SEL=>"false",
        IDIV_SEL=>0,
        DYN_FBDIV_SEL=>"false",
        FBDIV_SEL=>0,
        DYN_ODIV_SEL=>"false",
        ODIV_SEL=>8,
        PSDA_SEL=>"0000",
        DYN_DA_EN=>"false",
        DUTYDA_SEL=>"1000",
        CLKOUT_FT_DIR=>'1',
        CLKOUTP_FT_DIR=>'1',
        CLKOUT_DLY_STEP=>0,
        CLKOUTP_DLY_STEP=>0,
        CLKOUTD3_SRC=>"CLKOUT",
        CLKFB_SEL=>"internal",
        CLKOUT_BYPASS=>"false",
        CLKOUTP_BYPASS=>"false",
        CLKOUTD_BYPASS=>"false",
        CLKOUTD_SRC=>"CLKOUT",
        DYN_SDIV_SEL=>2
    )
    PORT MAP(
        CLKIN=>clk_in,
        CLKFB=>clkfb,
        IDSEL=>idsel,

```

```

FBDSEL=>fbdsel,
ODSEL=>odsel,
RESET=>reset,
RESET_P=>reset_p,
RESET_I=>reset_i,
RESET_S=>reset_s,
PSDA=>psda,
FDLY=>fdly,
DUTYDA=>dutyda,
LOCK=>lock,
CLKOUT=>clkout,
CLKOUTD=>clkoutd,
CLKOUTP=>clkoutp ,
CLKOUTD3=>clkoutd3

```

```
);
```

## 6.2 DLL/DLLDLY

### 6.2.1 DLL

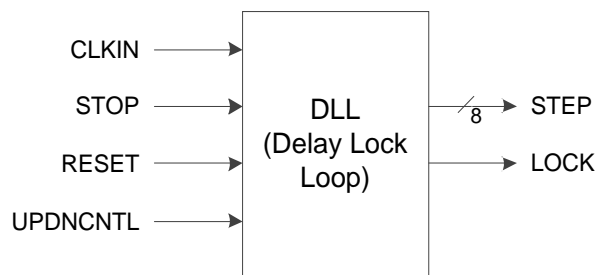
#### 原语介绍

DLL（Delay-Locked Loop）是延时锁相环，主要用于精确产生时间延时。支持器件：

GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55。

#### 结构框图

图 6-2 DLL 结构框图



## Port 介绍

表 6-4 Port 介绍

Port Name	I/O	Description
STEP[7:0]	Output	Step Code Output
LOCK	Output	Lock Output
CLKIN	Input	Clock Input
STOP	Input	Force DLL to stop working
RESET	Input	Reset Input
UPDNCNTL	Input	Control the Step Code Update

## Attribute 介绍

表 6-5 Attribute 介绍

Attribute Name	Type	Allowed Values	Defalut	Description
DLL_FORCE	Integer	0,1	0	Step and lock output mode can be selected 1:force lock and code 0:code/lock generated from DLL loop
CODESCAL	String	000,001,010,011,100,101, 110, 111	000	Phase offset config: 000:101° 001:112° 010:123° 011:135° 100:79° 101:68° 110:57° 111:45°
SCAL_EN	String	true,false	true	Output step mode can be selected: true:Phase offset correspond to parameter CODESCAL false:90°Phase offset
DIV_SEL	Integer	1'b0,1'b1	1'b0	Output lock mode can be selected: 1'b0:normal lock mode 1'b1:fast lock mode

## 原语例化

### Verilog 例化:

```

DLL dll_inst (
    .STEP(step),
    .LOCK(lock),
    .CLKIN(clkin),
    .STOP(stop),
    .RESET(reset),
    .UPDNCNTL(1'b0)
);
defparam dll_inst.DLL_FORCE = 1;

```

```

defparam dll_inst.CODESCAL = "000";
defparam dll_inst.SCAL_EN = "true";
defparam dll_inst.DIV_SEL = 1'b0;

```

**Vhdl 例化:**

```

COMPONENT DLL
  GENERIC(
    DLL_FORCE:integer:=0;
    DIV_SEL:bit:='1';
    CODESCAL:STRING:="000";
    SCAL_EN:STRING:="true"
  );
  PORT(
    CLKIN:IN std_logic;
    STOP:IN std_logic;
    RESET:IN std_logic;
    UPDNCNTL:IN std_logic;
    LOCK:OUT std_logic;
    STEP:OUT std_logic_vector(7 downto 0)
  );
END COMPONENT;
 uut:DLL
  GENERIC MAP(
    DLL_FORCE=>0,
    DIV_SEL=>'1',
    CODESCAL=>"000",
    SCAL_EN=>"true"
  )
  PORT MAP(
    CLKIN=>clk_in,
    STOP=>stop,
    RESET=>reset,
    UPDNCNTL=>updncntl,
    LOCK=>lock,
    STEP=>step
  );

```

## 6.2.2 DLLDLY

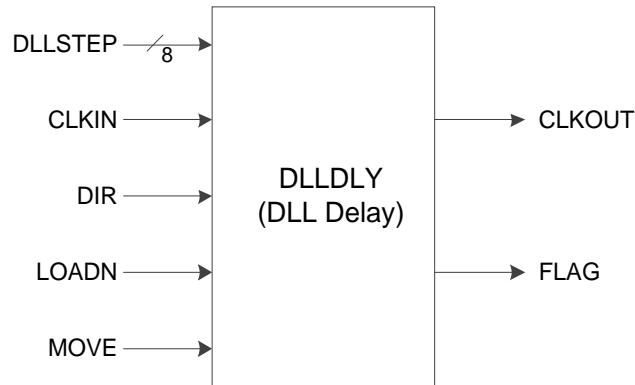
### 原语介绍

DLLDLY(DLL Delay)依据 DLLSTEP 信号对输入时钟进行调整，得到该时钟的延时调整输出。DLLSTEP 信号可来自于 DLL 的 STEP 输出。

支持器件：GW1N-1、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55。

## 结构框图

图 6-3 DLLDLY 结构框图



## Port 介绍

表 6-6 Port 介绍

Port Name	I/O	Description
CLKOUT	Output	Clock Output
FLAG	Output	Overflow Flag
DLLSTEP[7:0]	Input	STEP Input
CLKIN	Input	Clock Input
DIR	Input	Direction can be Selected to Decide Delay, Increase or Decrease
LOADN	Input	Control Delay Code's Download
MOVE	Input	Adjust Delay Value

## Attribute 介绍

表 6-7 Attribute 介绍

Attribute Name	Type	Allowed Values	Default	Description
DLL_INSEL	Integer	1'b0,1'b1	1'b0	DLLDLY can be bypassed 1'b0:bypass mode 1'b1: use dll_delay cell mode
DLY_SIGN	String	1'b0,1'b1	1'b0	Set symbol of delay adjustment: 1'b0:'+' 1'b1:'-'
DLY_ADJ	Integer	0~255	0	Delay adjustment: 1) dly_sign=0 dly_adj; 2) dly_sign=1 -256+dly_adj

## 原语例化

### Verilog 例化:

```
DLLDLY dlldly_0 (
```

```

        .CLKIN(clkin),
        .DLLSTEP(step[7:0]),
        .DIR(dir),
        .LOADN(loadn),
        .MOVE(move),
        .CLKOUT(clkout),
        .FLAG(flag)
    );
    defparam dlldly_0.DLL_INSEL=1'b1;
    defparam dlldly_0.DLY_SIGN=1'b1;
    defparam dlldly_0.DLY_ADJ=0;
Vhdl 例化:
    COMPONENT DLLDLY
        GENERIC(
            DLL_INSEL:bit='0';
            DLY_SIGN:bit='0';
            LY_ADJ:integer:=0
        );
        PORT(
            DLLSTEP:IN std_logic_vector(7 downto 0);
            CLKIN:IN std_logic;
            DIR,LOADN,MOVE:IN std_logic;
            CLKOUT:OUT std_logic;
            FLAG:OUT std_logic
        );
    END COMPONENT;
    uut:DLLDLY
        GENERIC MAP(
            DLL_INSEL=>'0',
            DLY_SIGN=>'0',
            LY_ADJ=>0
        )
        PORT MAP(
            DLLSTEP=>step,
            CLKIN=>clkin,
            DIR=>dir,
            LOADN=>loadn,
            MOVE=>move,
            CLKOUT=>clkout,
            FLAG=>flag
        );

```

## 6.3 CLKDIV

### 原语介绍

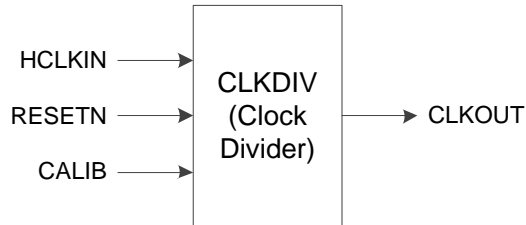
CLKDIV (Clock Divider)为时钟分频器,实现时钟动态调整。在 GW1N-6、GW1N-9 下支持 2/3.5/4/5/8 分频,其他器件下支持 2/3.5/4/5 分频。

支持器件: GW1N-1、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1N-4、GW1N-4B、GW1NR-4、

GW1NR-4B、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55。

### 结构框图

图 6-4 CLKDIV 结构框图



### Port 介绍

表 6-8 Port 介绍

Port Name	I/O	Description
HCLKIN	Input	Clock Input
RESETN	Input	Reset Input
CALIB	Input	Calib Signal, adjust output clock
CLKOUT	Output	Clock Output

### Attribute 介绍

表 6-9 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
DIV_MODE	2, 3.5, 4, 5, 8 (8 仅在 GW1N-6、GW1N-9 下支持)	2	Set the clock frequency division parameter
GSREN	false, true	false	Global reset

### 原语例化

#### Verilog 例化:

```

CLKDIV clkdiv_inst (
    .HCLKIN(hclk_in),
    .RESETN(resetn),
    .CALIB(calib),
    .CLKOUT(clkout)
);
defparam clkdiv_inst.DIV_MODE="3.5";
defparam clkdiv_inst.GSREN="false";
  
```

#### Vhdl 例化:

```

COMPONENT CLKDIV
  GENERIC(
    DIV_MODE:STRING:="2";
    GSREN:STRING:="false"
  );
  PORT(
  
```



```

        HCLKIN:IN std_logic;
        RESETN:IN std_logic;
        CALIB:IN std_logic;
        CLKOUT:OUT std_logic
    );
END COMPONENT;
    uut:CLKDIV
    GENERIC MAP(
        DIV_MODE=>"2",
        GSREN=>"false"
    )
    PORT MAP(
        HCLKIN=>hclk,
        RESETN=>resetn,
        CALIB=>calib,
        CLKOUT=>clkout
    );

```

## 6.4 DQCE

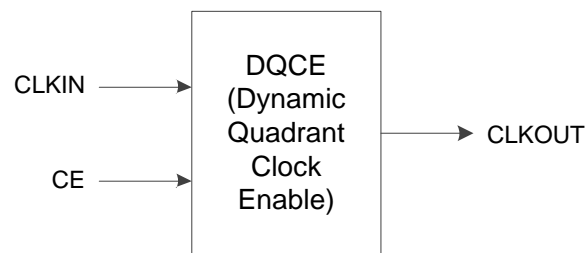
### 原语介绍

DQCE (Dynamic Quadrant Clock Enable) ， 动态使能高云 FPGA 象限时钟 GCLK0~GCLK5。

支持器件：GW1N-1、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55。

### 结构框图

图 6-5 DQCE 结构框图



### Port 介绍

表 6-10 Port 介绍

Port Name	I/O	Description
CLKIN	Input	Clock Input
CE	Input	Clock Enable
CLKOUT	Output	Clock Output

## 原语例化

### Verilog 例化:

```
DQCE dqce_inst (
    .CLKIN(clkin),
    .CE(ce),
    .CLKOUT(clkout)
);
```

### Vhdl 例化:

```
COMPONENT DQCE
    PORT(
        CLKOUT:OUT std_logic;
        CE:IN std_logic;
        CLKIN:IN std_logic
    );
END COMPONENT;
 uut:DQCE
PORT MAP(
    CLKIN=>clkin,
    CLKOUT=>clkout,
    CE=>ce
);
```

## 6.5 DCS

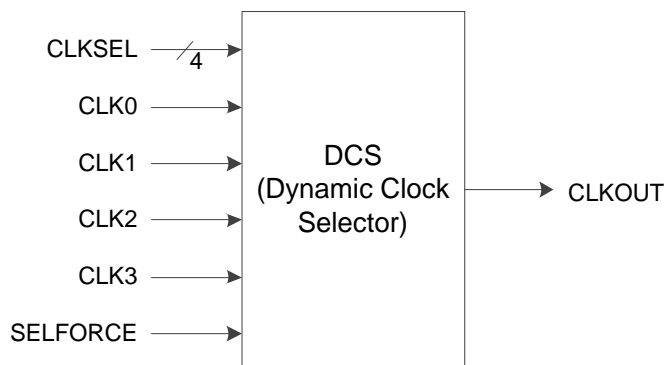
### 原语介绍

DCS (Dynamic Clock Select) 是动态时钟选择器，动态选择象限时钟 GCLK6 和 GCLK7。

支持器件：GW1N-1、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55。

### 结构框图

图 6-6 DCS 结构框图



## Port 介绍

表 6-11 Port 介绍

Port Name	I/O	Description
CLK0	Input	Clock0 Input
CLK1	Input	Clock1 Input
CLK2	Input	Clock2 Input
CLK3	Input	Clock3 Input
CLKSEL[3:0]	Input	Clock Select Signal
SELFORCE	Input	Select Force Signal
CLKOUT	Output	Clock Output

## Attribute 介绍

表 6-12 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
DCS_MODE	CLK0,CLK1,CLK2,CLK3, GND,VCC,RISING,FALLING, CLK0_GND,CLK1_GND, CLK2_GND,CLK3_GND, CLK0_VCC,CLK1_VCC, CLK2_VCC,CLK3_VCC	RISING	Set the clock selection mode

## 原语例化

### Verilog 例化:

```
DCS dcs_inst (
    .CLK0(clk0),
    .CLK1(clk1),
    .CLK2(clk2),
    .CLK3(clk3),
    .CLKSEL(clksel[3:0]),
    .SELFORCE(selforce),
    .CLKOUT(clkout)
);
defparam dcs_inst.DCS_MODE="RISING";
```

### Vhdl 例化:

```
COMPONENT DCS
    GENERIC(DCS_MODE:string:="RISING");
    PORT(
        CLK0:IN std_logic;
        CLK1:IN std_logic;
        CLK2:IN std_logic;
        CLK3:IN std_logic;
        CLKSEL:IN std_logic_vector(3 downto 0);
        SELFORCE:IN std_logic;
        CLKOUT:OUT std_logic
    );
END COMPONENT;
```

```

uut:DCS
    GENERIC MAP(DCS_MODE=>"RISING")
    PORT MAP(
        CLK0=>clk0,
        CLK1=>clk1,
        CLK2=>clk2,
        CLK3=>clk3,
        CLKSEL=>clkssel,
        SELFORCE=>selforce,
        CLKOUT=>clkout
    );
    
```

## 6.6 DQS

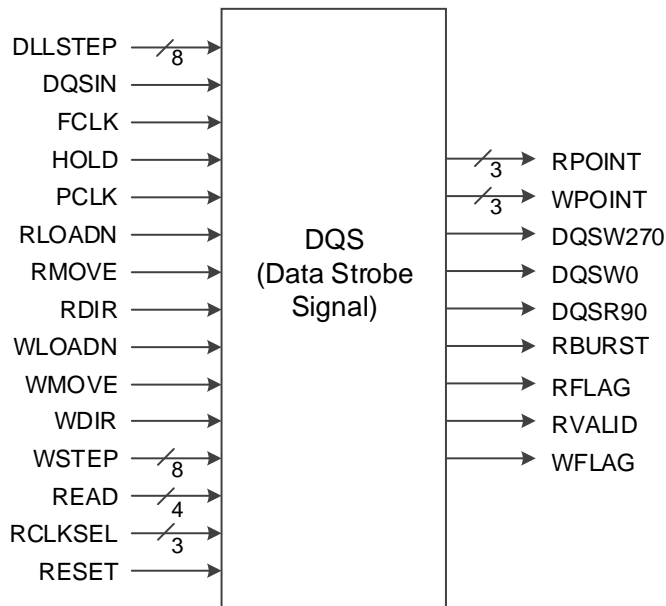
### 原语介绍

DQS（Bidirectional Data Strobe Circuit for DDR Memory）是 DDR 存储器接口双向数据选通脉冲电路，是内存控制器 IP 的关键器件，主要用于调整 DQSIN 与 DQSR90、DQSW0 与 DQSW270 信号间的相位关系并完成写平衡、读校准。

支持器件：GW2A-18、GW2AR-18、GW2A-55。

### 结构框图

图 6-7 DQS 结构框图



### Port 介绍

表 6-13 Port 介绍

Port Name	I/O	Description
DLLSTEP[7:0]	input	DQS delay control from DLL
DQSIN	input	DQS signal from PIO
FCLK	input	from 4 different ECLK tree output
HOLD	input	Stop DQSW0、DQSW270 and clear RPOINT、WPOINT

Port Name	I/O	Description
PCLK	input	from PCLK tree
RDIR	input	“0” to increase the code “1” to decrease the code for DDR read
RLOADN	input	asynchronous reset the final delay code to factory default value for DDR read
RMOVE	input	Move pulse’s rising edge will change the code according to direction value for DDR read
WDIR	input	“0” to increase the code “1” to decrease the code for DDR write
WLOADN	input	Asynchronous reset the final delay code to factory default value for DDR write
WMOVE	input	Move pulse’s rising edge will change the code according to direction value for DDR write
WSTEP[7:0]	input	DDR write leveling control also used for CDR mode
READ[3:0]	input	Read signal for DDR read mode
RCLKSEL[2:0]	input	Select read clock source and polarity control
RESET	input	DQS reset control
RPOINT[2:0]	output	FIFO control READ pointer (3-bit) to FIFO in PIC
WPOINT[2:0]	output	FIFO control WRITE pointer (3-bit) to FIFO in PIC
DQSW0	output	SCLK/ECLK phase shifted or delayed by 0 degree output
DQSW270	output	SCLK/ECLK phase shifted or delayed by 270 degree output
DQSR90	output	DQSI phase shifted or delayed by 90 degree output
RFLAG	output	Margin test output flag for READ to indicate the under-flow or over-flow
WFLAG	output	Margin test output flag for WRITE to indicate the under-flow or over-flow
RVALID	output	Data Valid Flag for READ mode
RBURST	output	READ burst detect output

## Attribute 介绍

表 6-14 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
FIFO_MODE_SEL	1'b0 , 1'b1	1'b0	FIFO mode config 1'b0: DDR memory mode 1'b1: GDDR mode
RD_PNTR	000,001,010,011, 100,101,110,111	3'b000	FIFO read pointer setting
DQS_MODE	X1,X2_DDR2,X2_ DDR3,X4,X2_DD R3_EXT	X1	MDDR select
HWL	false,true	false	Updata0/1 time relation control
GSREN	false,true	false	Global reset can be set

## 原语例化

### Verilog 例化:

```

DQS uut (
  .DQSIN(dqs),
  .PCLK(pclk),
  .FCLK(fclk),
  .RESET(reset),
  .READ(read),
  .RCLKSEL(rsel),
  .DLLSTEP(step),
  .WSTEP(wstep),
  .RLOADN(1'b0),
  .RMOVE(1'b0),
  .RDIR(1'b0),
  .WLOADN(1'b0),
  .WMOVE(1'b0),
  .WDIR(1'b0),
  .HOLD(hold),
  .DQSR90(dqsr90),
  .DQSW0(dqsw0),
  .DQSW270(dqsw270),
  .RPOINT(rpoint),
  .WPOINT(wpoint),
  .RVALID(rvalid),
  .RBURST(rburst),
  .RFLAG(rflag),
  .WFLAG(wflag)
);
defparam uut.DQS_MODE = "X1";
defparam uut.FIFO_MODE_SEL = 1'b0;
defparam uut.RD_PNTR = 3'b001;
Vhdl 例化:
COMPONENT DQS
  GENERIC(
    FIFO_MODE_SEL:bit:='0';
    RD_PNTR : bit_vector:="000";
    DQS_MODE:string:="X1";
    HWL:string:="false";
    GSREN : string:="false"
  );
  PORT(
    DQSIN,PCLK,FCLK,RESET:IN std_logic;
    READ:IN std_logic_vector(3 downto 0);
    RCLKSEL:IN std_logic_vector(2 downto 0);
    DLLSTEP,WSTEP:IN std_logic_vector(7 downto 0);
    RLOADN,RMOVE,RDIR,HOLD:IN std_logic;
    WLOADN,WMOVE,WDIR:IN std_logic;
    DQSR90,DQSW0,DQSW270:OUT std_logic;
    RPOINT, WPOINT:OUT std_logic_vector(2 downto 0);
    RVALID,RBURST,RFLAG,WFLAG:OUT std_logic
  );
END COMPONENT;

```

```

uut:DQS
  GENERIC MAP(
    FIFO_MODE_SEL=>'0',
    RD_PNTR=>"000",
    DQS_MODE=>"X1",
    HWL=>"false",
    GSREN=>"false"
  )
  PORT MAP(
    DQSIN=>dqsin,
    PCLK=>pclk,
    FCLK=>fclk,
    RESET=>reset,
    READ=>read,
    RCLKSEL=>rclkssel,
    DLLSTEP=>step,
    WSTEP=>wstep,
    RLOADN=>rloadn,
    RMOVE=>rmove,
    RDIR=>rdir,
    HOLD=>hold,
    WLOADN=>wloadn,
    WMOVE=>wmove,
    WDIR=>wdir,
    DQSR90=>dqsr90,
    DQSW0=>dqsw0,
    DQSW270=>dqsw270,
    RPOINT=>rpoint,
    WPOINT=>wpoint,
    RVALID=>rvalid,
    RBURST=>rburst,
    RFLAG=>rflag,
    WFLAG=>wflag
  );

```

## 6.7 OSC

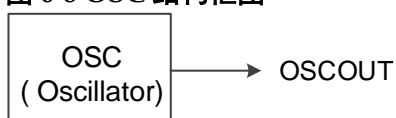
### 原语介绍

OSC(Oscillator)是片内晶振。

支持器件：GW1N-1、GW1N-2、GW1N-2B、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55。

### 结构框图

图 6-8 OSC 结构框图



## Port 介绍

表 6-15 Port 介绍

Port Name	I/O	Description
OSCOUT	output	OSC Clock Output

## Attribute 介绍

表 6-16 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
FREQ_DIV	2~128(even)	100	Frequency Division Factor
DEVICE	GW1N-1、GW1N-2、 GW1N-2B、GW1N-4、 GW1N-4B、GW1NR-4、 GW1NR-4B、GW1N-6、 GW1N-9、GW1NR-9、 GW2A-18、GW2AR-18、 GW2A-55。	GW1N-1(GW1N 系列) GW2A-18(GW2A 系列)	Device Select

## 原语例化

### Verilog 例化:

```
OSC uut(
    .OSCOUT(oscout)
);
defparam uut.FREQ_DIV=100;
defparam uut.DEVICE="GW2A-18";
```

### Vhdl 例化:

```
COMPONENT OSC
    GENERIC(
        FREQ_DIV:integer:=100;
        DEVICE:string:="GW2A-18"
    );
    PORT(OSCOUT:OUT STD_LOGIC);
END COMPONENT;
uut:OSC
    GENERIC MAP(
        FREQ_DIV=>100,
        DEVICE=>"GW2A-18"
    )
    PORT MAP(OSCOUT=>oscout);
```

## 6.8 OSCZ

### 原语介绍

OSCZ(Oscillator)是带有动态关闭 OSC 功能的片内晶振，增加了 model 的低功耗功能。

支持器件：GW1NZ-1



## 结构框图

图 6-9 OSCZ 结构框图



## Port 介绍

表 6-17 Port 介绍

Port Name	I/O	Description
OSCEN	input	OSC Enable
OSCOUT	output	OSC Clock Output

## Attribute 介绍

表 6-18 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
FREQ_DIV	2~128(even)	100	Frequency Division Factor

## 原语例化

### Verilog 例化:

```

OSCZ uut(
    .OSCOUT(oscout),
    .OSCEN(oscen)
);
defparam uut.FREQ_DIV=100;
  
```

### Vhdl 例化:

```

COMPONENT OSCZ
  GENERIC(
    FREQ_DIV:integer:=100;
  );
  PORT(
    OSCOUT:OUT STD_LOGIC;
    OSCEN:IN std_logic
  );
END COMPONENT;
uut:OSCZ
  GENERIC MAP(
    FREQ_DIV=>100,
  )
  PORT MAP(
    OSCOUT=>oscout,
    OSCEN(oscen)
  );
  
```

## 6.9 OSCF

### 原语介绍

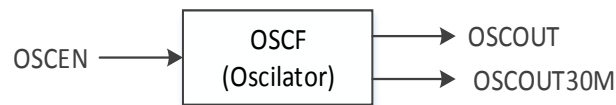
OSCF(Oscillator with CLKOUT30M and Dynamic OSC Enable)是带

30M 输出时钟和动态使能的片内晶振。

支持器件： GW1NS-2、GW1NS-2C 、GW1NSR-2、GW1NSR-2C

### 结构框图

图 6-10 OSCF 结构框图



### Port 介绍

表 6-19 Port 介绍

Port Name	I/O	Description
OSCEN	input	OSC Enable
OSCOUT	output	OSC Clock Output
OSCOUT30M	output	OSC Clock Output For Flash128K

### Attribute 介绍

表 6-20 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
FREQ_DIV	2~128(even)	96	Frequency Division Factor

### 原语例化

#### Verilog 例化:

```

OSCF uut(
    .OSCOUT(oscout),
    .OSCOUT30M(oscout30m),
    .OSCEN(oscen)
);
defparam uut.FREQ_DIV=96;
  
```

#### Vhdl 例化:

```

COMPONENT OSCF
  GENERIC(
    FREQ_DIV:integer:=96;
  );
  PORT(
    OSCOUT:OUT std_logic;
    OSCOUT30M:OUT std_logic;
    OSCEN:IN std_logic
  );
END COMPONENT;
uut:OSCF
  GENERIC MAP(FREQ_DIV=>96)
  PORT MAP(
    OSCOUT=>oscout,
    OSCOUT30M=>oscout30m,
    OSCEN(oscen)
  );
  
```

);

## 6.10 DHCEN

### 原语介绍

DHCEN (Dynamic HCLK Clock Enable with Inverted Gate) 可用于 hclk 的 stop，低电平时导通，可动态的打开/关闭高速时钟信号。

支持器件：GW1N-1、GW1NZ-1、GW1N-2、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1N-2B、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55。

### 结构框图

图 6-11 DHCEN 结构框图



### Port 介绍

表 6-21 Port 介绍

Port Name	I/O	Description
CLKIN	input	Clock Input
CE	input	Clock Enable
CLKOUT	output	Clock Output

### 原语例化

#### Verilog 例化:

```
DHCEN dhcen_inst (
    .CLKIN(clkin),
    .CE(ce),
    .CLKOUT(clkout)
);
```

#### Vhdl 例化:

```
COMPONENT DHCEN
    PORT(
        CLKOUT:OUT std_logic;
        CE:IN std_logic;
        CLKIN:IN std_logic
    );
END COMPONENT;
 uut:DHCEN
PORT MAP(
    CLKIN=>clkin,
    CLKOUT=>clkout,
    CE=>ce
```

);

## 6.11 BUFG

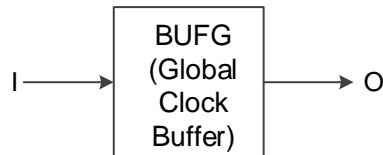
### 原语介绍

BUFG(Global Clock Buffer)是全局时钟缓冲器。

支持器件：GW1N-1、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55。

### 结构框图

图 6-12 BUFG 结构框图



### Port 介绍

表 6-22 Port 介绍

Port Name	I/O	Description
O	output	Clock Output
I	input	Clock Input

### 原语例化

#### Verilog 例化:

```

BUFG uut(
    .O(o),
    .I(i)
);
  
```

#### Vhdl 例化:

```

COMPONENT BUFG
  PORT(
    O:OUT std_logic;
    I:IN std_logic
  );
END COMPONENT;
uut:BUFG
  PORT MAP(
    O=>o,
    I=>i
  );
  
```

## 6.12 BUFS

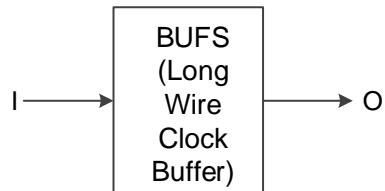
### 原语介绍

BUFS(Long Wire Clock Buffer)是长线时钟缓冲器。

支持器件：GW1N-1、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55。

### 结构框图

图 6-13 BUFS 结构框图



### Port 介绍

表 6-23 Port 介绍

Port Name	I/O	Description
O	output	Clock Output
I	input	Clock Input

### 原语例化

#### Verilog 例化:

```

BUFS uut(
    .O(o),
    .I(i)
);
  
```

#### Vhdl 例化:

```

COMPONENT BUFS
  PORT(
    O:OUT std_logic;
    I:IN std_logic
  );
END COMPONENT;
uut:BUFS
  PORT MAP(
    O=>o,
    I=>i
  );
  
```

# 7 User Flash

## 7.1 FLASH96K

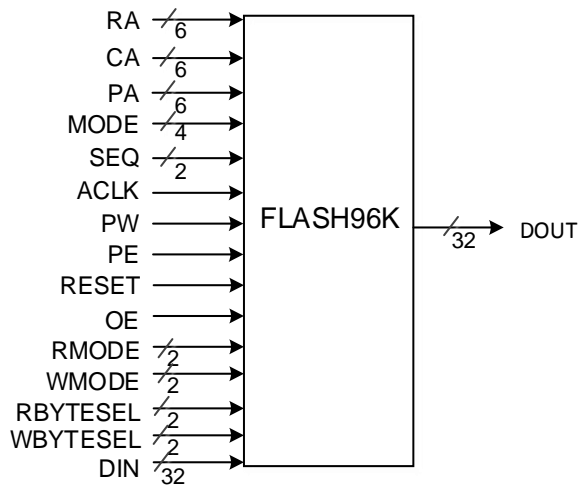
### 原语介绍

FLASH96K（96Kbit User Flash）的存储空间为 96K bit。寄存器的宽度和深度是固定的，不可对其进行配置。其宽度为 4 Byte（32 bit），地址深度为 3k，具有非易失性和断电保存功能，但不具有 BSRAM 的初始值功能。

支持器件：GW1N-1。

### 结构框图

图 7-1 FLASH96K 结构框图



## Port 介绍

表 7-1 Port 介绍

Port Name	I/O	Description
DOUT[31:0]	Output	Data Output
DIN[31:0]	Input	Data Input
RA[5:0]	Input	Row Address
CA[5:0]	Input	Column Address
PA[5:0]	Input	Page latch Address
MODE[3:0]	Input	Operation mode select
SEQ[1:0]	Input	NV operation sequence control
ACLK	Input	Synchronous clock for read and write operation
PW	Input	Write page latch clock
RESET	Input	Macro reset
PE	Input	Pump enable
OE	Input	Output enable
RMODE[1:0]	Input	Read out bit width select
WMODE[1:0]	Input	Write in bit width select
RBYTESEL[1:0]	Input	Read data Byte address
WBYTESEL[1:0]	Input	Write data Byte address

## 原语例化

### Verilog 例化:

```
FLASH96K flash96k_inst(
    .RA(ra[5:0]),
    .CA(ca[5:0]),
    .PA(pa[5:0]),
    .MODE(mode[3:0]),
    .SEQ(seq[1:0]),
    .ACLK(aclk),
    .PW(pw),
    .RESET(reset),
    .PE(pe),
    .OE(oe),
    .RMODE(rmode[1:0]),
    .WMODE(wmode[1:0]),
    .RBYTESEL(rbytesel[1:0]),
    .WBYTESEL(wbytesel[1:0]),
    .DIN(din[31:0]),
    .DOUT(dout[31:0])
);
```

### Vhdl 例化:

```
COMPONENT FLASH96K
    PORT(
        RA:IN std_logic_vector(5 downto 0);
```

```

        CA:IN std_logic_vector(5 downto 0);
        PA:IN std_logic_vector(5 downto 0);
        MODE:IN std_logic_vector(3 downto 0);
        SEQ:IN std_logic_vector(1 downto 0);
        ACLK:IN std_logic;
        PW:IN std_logic;
        RESET:IN std_logic;
        PE:IN std_logic;
        OE:IN std_logic;
        RMODE:IN std_logic_vector(1 downto 0);
        WMODE:IN std_logic_vector(1 downto 0);
        RBYTESEL:IN std_logic_vector(1 downto 0);
        WBYTESEL:IN std_logic_vector(1 downto 0);
        DIN:IN std_logic_vector(31 downto 0);
        DOUT:OUT std_logic_vector(31 downto 0)
    );
END COMPONENT;
 uut: FLASH96K
     PORT MAP (
         RA=>ra,
         CA=>ca,
         PA=>pa,
         MODE=>mode,
         SEQ=>seq,
         RESET=>reset,
         ACLK=>ack,
         PW=>pw,
         PE=>pe,
         OE=>oe,
         RMODE=>rmode,
         WMODE=>wmode,
         RBYTESEL=>rbytesel,
         WBYTESEL=> wbytesel,
         DIN=>din,
         DOUT=>dout
     );

```

## 7.2 FLASH96KZ

### 原语介绍

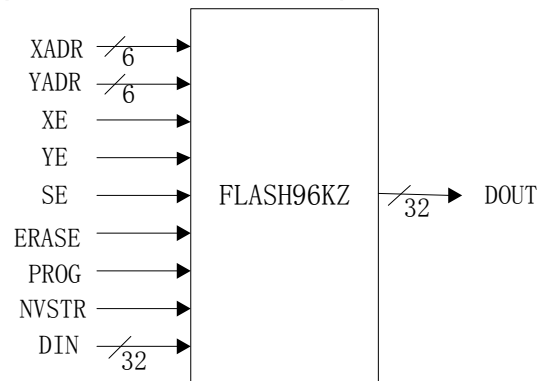
FLASH96KZ（96Kbit User Flash）的存储空间为 96K bit。寄存器的宽度和深度是固定的，不可对其进行配置。具有非易失性和断电保存功能，但不具有 BSRAM 的初始值功能。

支持器件：GW1NZ-1。



## 结构框图

图 7-2 FLASH96KZ 结构框图



## Port 介绍

表 7-2 Port 介绍

Port Name	I/O	Description
DOUT[31:0]	Output	Data Output
DIN[31:0]	Input	Data Input
XADR[5:0]	Input	X address input
YADR[5:0]	Input	Y address input
XE	Input	X address enable
YE	Input	Y address enable
SE	Input	Sense amplifier enable
ERASE	Input	Defines erase cycle
PROG	Input	Defines program cycle
NVSTR	Input	Defines non-volatile store cycle

## 原语例化

### Verilog 例化:

```
FLASH96KZ flash96kz_inst(
    .XADR(xadr[5:0]),
    .YADR(yadr[5:0]),
    .XE(xe),
    .YE(ye),
    .SE(se),
    .ERASE(erase),
    .PROG(prog),
    .NVSTR(nvstr),
    .DIN(din[31:0]),
    .DOUT(dout[31:0])
);
```

### Vhdl 例化:

```
COMPONENT FLASH96KZ
    PORT(
```

```

XADR:IN std_logic_vector(5 downto 0);
YADR:IN std_logic_vector(5 downto 0);
XE:IN std_logic;
YE:IN std_logic;
SE:IN std_logic;
ERASE:IN std_logic;
PROG:IN std_logic;
NVSTR:IN std_logic;
DIN:IN std_logic_vector(31 downto 0);
DOUT:OUT std_logic_vector(31 downto 0)
);
END COMPONENT;
 uut: FLASH96KZ
   PORT MAP (
     XADR=>xadr,
     YADR=>yadr,
     XE=>xe,
     YE=>ye,
     SE=>se,
     ERASE=>erase,
     PROG=>prog,
     NVSTR=>nvstr,
     DIN=>din,
     DOUT=>dout
   );

```

## 7.3 FLASH128K

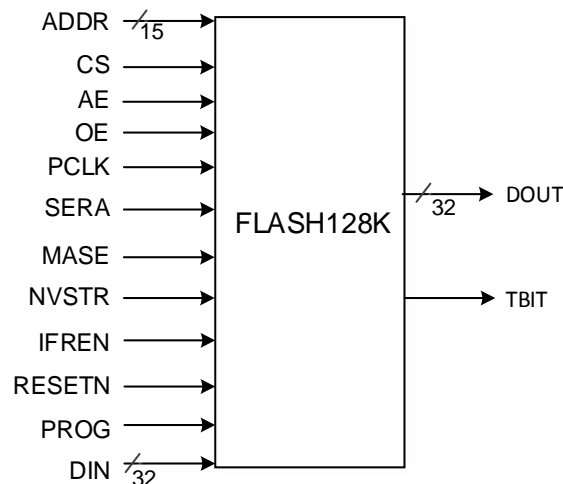
### 原语介绍

FLASH128K（128KByte Embedded Flash）的存储空间为 128K Byte。寄存器的宽度和深度是固定的，不可对其进行配置。具有非易失性和断电保存功能，但不具有 BSRAM 的初始值功能。

支持器件：GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C。

### 结构框图

图 7-3 FLASH128K 结构框图



## Port 介绍

表 7-3 Port 介绍

Port Name	I/O	Description
DOUT[31:0]	Output	Data Output
TBIT	Output	Indicator of write or erase
DIN[31:0]	Input	Data Input
ADDR[14:0]	Input	Address Input
CS	Input	Chip enable
AE	Input	Address enable
OE	Input	Output enable
PCLK	Input	Clock input
PROG	Input	Defines program cycle
SERA	Input	Sector erase signal
MASE	Input	Chip erase signal
NVSTR	Input	Defines non-volatile store cycle
IFREN	Input	Flash IP information page Selection
RESETN	Input	Power On Reset Input

## 原语例化

### Verilog 例化:

```
FLASH128K flash128k_inst(
    .ADDR(addr[14:0]),
    .CS(cs),
    .AE(ae),
    .OE(oe),
    .PCLK(pclk),
    .PROG(prog),
    .SERA(sera),
    .MASE(mase),
    .NVSTR(nvstr),
    .IFREN(ifren),
    .RESETN(resetn),
    .DIN(din[31:0]),
    .DOUT(dout[31:0]),
    .TBIT(tbit)
);
```

### Vhdl 例化:

```
COMPONENT FLASH128K
PORT(
    DIN:IN std_logic_vector(31 downto 0);
    ADDR:IN std_logic_vector(14 downto 0);
    CS:IN std_logic;
    AE:IN std_logic;
    OE:IN std_logic;
```

```

        PCLK:IN std_logic;
        PROG:IN std_logic;
        SERA:IN std_logic;
        MASE:IN std_logic;
        NVSTR:IN std_logic;
        IFREN:IN std_logic;
        RESETN:IN std_logic;
        DOUT:OUT std_logic_vector(31 downto 0);
        TBIT:OUT std_logic;
    );
END COMPONENT;
 uut: FLASH128K
     PORT MAP (
        DIN=>din,
        ADDR=>addr,
        CS=>cs,
        AE=>ae,
        OE=>oe,
        PCLK=>pclk,
        PROG=>prog,
        SERA=>sera,
        MASE=>mase,
        NVSTR=>nvstr,
        IFREN=>ifren,
        RESETN=>resetn,
        DOUT=>dout,
        TBIT=>tbit
    );

```

## 7.4 FLASH256K

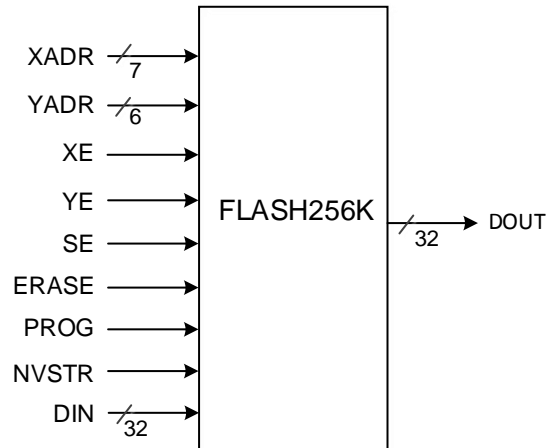
### 原语介绍

FLASH256K（256Kbit User Flash）的存储空间为 256K bit。寄存器的宽度和深度是固定的，不可对其进行配置。具有非易失性和断电保存功能，但不具有 BSRAM 的初始值功能。

支持器件：GW1N-2、GW1N-2B、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B。

## 结构框图

图 7-4 FLASH256K 结构框图



## Port 介绍

表 7-4 Port 介绍

Port Name	I/O	Description
DOUT[31:0]	Output	Data Output
DIN[31:0]	Input	Data Input
XADR[6:0]	Input	X address input
YADR[5:0]	Input	Y address input
XE	Input	X address enable
YE	Input	Y address enable
SE	Input	Sense amplifier enable
PROG	Input	Defines program cycle
ERASE	Input	Defines erase cycle
NVSTR	Input	Defines non-volatile store cycle

## 原语例化

### Verilog 例化:

```
FLASH256K flash256k_inst(
    .XADR(xadr[6:0]),
    .YADR(yadr[5:0]),
    .XE(xe),
    .YE(ye),
    .SE(se),
    .ERASE(erase),
    .PROG(prog),
    .NVSTR(nvstr),
    .DIN(din[31:0]),
    .DOUT(dout[31:0])
);
```

### Vhdl 例化:

```
COMPONENT FLASH256K
```

```

PORT(
    DIN:IN std_logic_vector(31 downto 0);
    XADR:IN std_logic_vector(6 downto 0);
    YADR:IN std_logic_vector(5 downto 0);
    XE:IN std_logic;
    YE:IN std_logic;
    SE:IN std_logic;
    ERASE:IN std_logic;
    PROG:IN std_logic;
    NVSTR:IN std_logic;
    DOUT:OUT std_logic_vector(31 downto 0)
);
END COMPONENT;
 uut: FLASH256K
    PORT MAP (
        DIN=>din,
        XADR=>xadr,
        YADR=>yadr,
        XE=>xe,
        YE=>ye,
        SE=>se,
        ERASE=>erase,
        PROG=>prog,
        NVSTR=>nvstr,
        DOUT=>dout
    );

```

## 7.5 FLASH608K

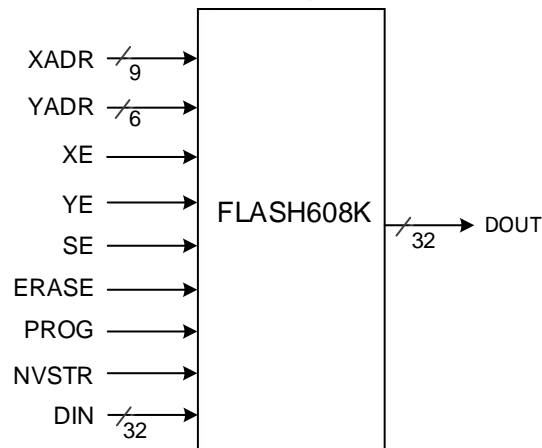
### 原语介绍

FLASH608K（608Kbit Users Flash）的存储空间为 608K bit。寄存器的宽度和深度是固定的，不可对其进行配置。具有非易失性和断电保存功能，但不具有 BSRAM 的初始值功能。

支持器件：GW1N-6、GW1N-9、GW1NR-9、。

### 结构框图

图 7-5 FLASH608K 结构框图



## Port 介绍

表 7-5 Port 介绍

Port Name	I/O	Description
DOUT[31:0]	Output	Data Output
DIN[31:0]	Input	Data Input
XADR[8:0]	Input	X address input
YADR[5:0]	Input	Y address input
XE	Input	X address enable
YE	Input	Y address enable
SE	Input	Sense amplifier enable
PROG	Input	Defines program cycle
ERASE	Input	Defines erase cycle
NVSTR	Input	Defines non-volatile store cycle

## 原语例化

### Verilog 例化:

```
FLASH608K flash608k_inst(
    .XADR(xadr[8:0]),
    .YADR(yadr[5:0]),
    .XE(xe),
    .YE(ye),
    .SE(se),
    .ERASE(erase),
    .PROG(prog),
    .NVSTR(nvstr),
    .DIN(din[31:0]),
    .DOUT(dout[31:0])
);
```

### Vhdl 例化:

```
COMPONENT FLASH608K
    PORT(
        DIN:IN std_logic_vector(31 downto 0);
        XADR:IN std_logic_vector(8 downto 0);
        YADR:IN std_logic_vector(5 downto 0);
        XE:IN std_logic;
        YE:IN std_logic;
        SE:IN std_logic;
        ERASE:IN std_logic;
        PROG:IN std_logic;
        NVSTR:IN std_logic;
        DOUT:OUT std_logic_vector(31 downto 0)
    );
END COMPONENT;
 uut: FLASH608K
    PORT MAP (
```

```
DIN=>din,  
XADR=>xadr,  
YADR=>yadr,  
XE=>xe,  
YE=>ye,  
SE=>se,  
ERASE=>erase,  
PROG=>prog,  
NVSTR=>nvstr,  
DOUT=>dout  
);
```





## 8.1 MCU

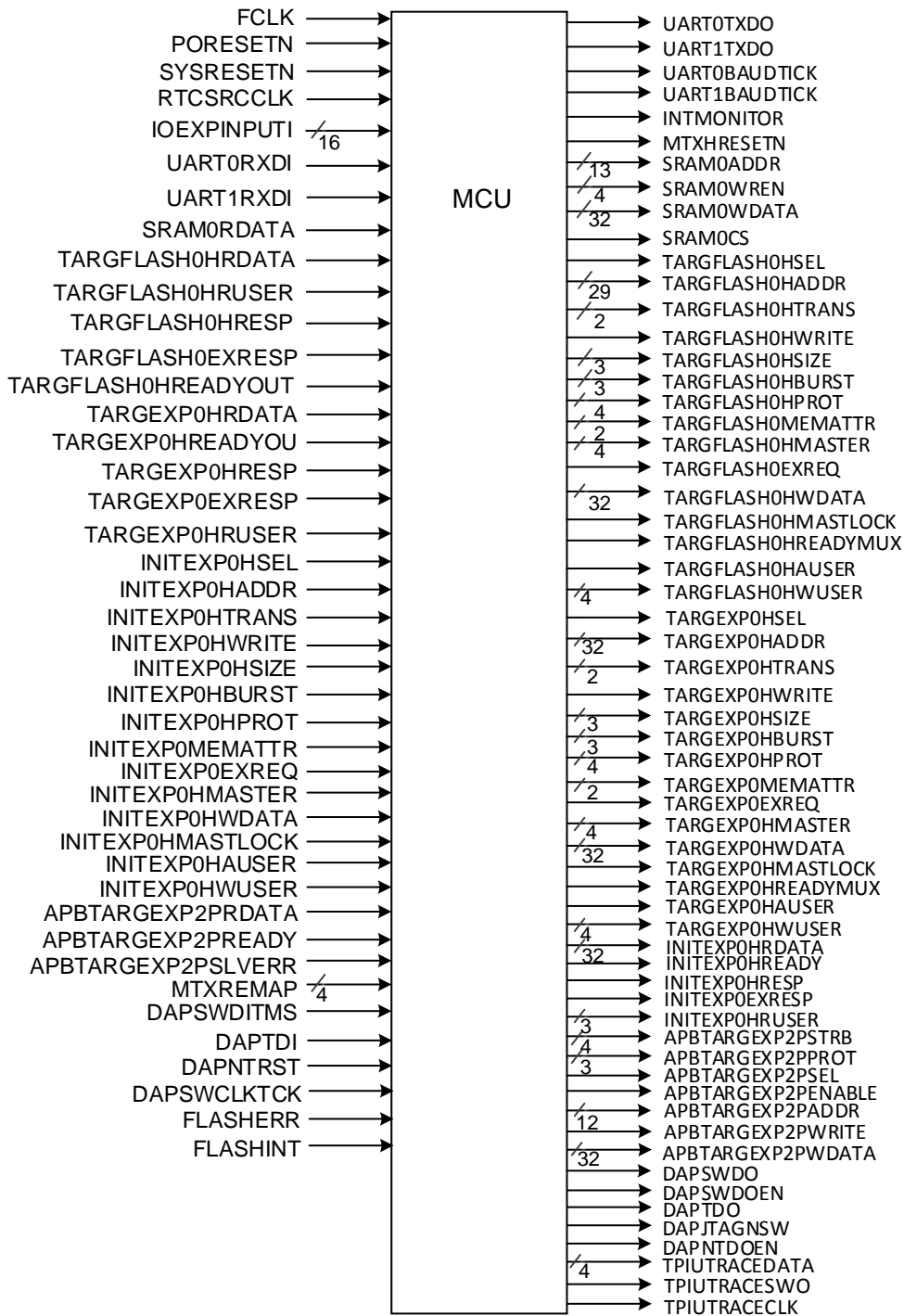
### 原语介绍

MCU(ARM Cortex-M3 Microcontroller Unit)是一款基于 ARM Cortex-M3 的微处理器。采用了 32 位 AHB/APB 的总线模式。其内部实现了 2 个 UART、2 个 Timer 和 Watchdog 的功能。并且对外提供 16 位 GPIO、2 个 UART、JTAG、2 个 User Interrupt 接口。以及 AHB Flash 读取接口、AHB Sram 读写接口。同时对外还提供了 2 个 AHB 总线扩展接口和 1 个 APB 总线扩展接口。

支持器件： GW1NS-2C 、 GW1NSR-2C。

结构框图

图 8-1 MCU 结构框图



## Port 介绍

表 8-1 Port 介绍

Port Name	I/O	Description
FCLK	input	Free running clock
PORESETN	input	Power on reset
SYSRESETN	input	System reset
RTCSRCLK	input	Used to generate RTC clock
IOEXPINPUTI[15:0]	input	IOEXPINPUTI
UART0RXDI	input	UART0RXDI
UART1RXDI	input	UART1RXDI
SRAM0DATA[31:0]	input	SRAM Read data bus
TARGFLASH0HRDATA[31:0]	input	TARGFLASH0, HRDATA
TARGFLASH0HRUSER[2:0]	input	TARGFLASH0, HRUSER
TARGFLASH0HRESP	input	TARGFLASH0, HRESP
TARGFLASH0EXRESP	input	TARGFLASH0, EXRESP
TARGFLASH0HREADYOUT	input	TARGFLASH0, EXRESP
TARGEXP0HRDATA[31:0]	input	TARGEXP0, HRDATA
TARGEXP0HREADYOUT	input	TARGEXP0, HREADY
TARGEXP0HRESP	input	TARGEXP0, HRESP
TARGEXP0EXRESP	input	TARGEXP0, EXRESP
TARGEXP0HRUSER[2:0]	input	TARGEXP0, HRUSER
INITEXP0HSEL	input	INITEXP0, HSELx
INITEXP0HADDR[31:0]	input	INITEXP0, HADDR
INITEXP0HTRANS[1:0]	input	INITEXP0, HTRANS
INITEXP0HWRITE	input	INITEXP0, HWRITE
INITEXP0HSIZE[2:0]	input	INITEXP0, HSIZE
INITEXP0HBURST[2:0]	input	INITEXP0, HBURST
INITEXP0HPROT[3:0]	input	INITEXP0, HPROT
INITEXP0MEMATTR[1:0]	input	INITEXP0, MEMATTR
INITEXP0EXREQ	input	INITEXP0, EXREQ
INITEXP0HMASTER[3:0]	input	INITEXP0, HMASTER
INITEXP0HWDATA[31:0]	input	INITEXP0, HWDATA
INITEXP0HMASTLOCK	input	INITEXP0, HMASTLOCK
INITEXP0HAUSER	input	INITEXP0, HAUSER
INITEXP0HWUSER[3:0]	input	INITEXP0, HWUSER
APBTARGEXP2PRDATA[31:0]	input	APBTARGEXP2, PRDATA
APBTARGEXP2PREADY	input	APBTARGEXP2, PREADY
APBTARGEXP2PSLVERR	input	APBTARGEXP2, PSLVERR
MTXREMAP[3:0]	input	The MTXREMAP signals control the remapping of the boot memory range.
DAPSWDITMS	input	Debug TMS
DAPTDI	input	Debug TDI
DAPNTRST	input	Test reset
DAPSWCLKTCK	input	Test clock / SWCLK
FLASHERR	input	Output clock, used by the TPA to sample the other pins

Port Name	I/O	Description
FLASHINT	input	Output clock, used by the TPA to sample the other pins
IOEXPOUTPUTO[15:0]	output	IOEXPOUTPUTO
IOEXPOUTPUTENO[15:0]	output	IOEXPOUTPUTENO
UART0TXDO	output	UART0TXDO
UART1TXDO	output	UART1TXDO
UART0BAUDTICK	output	UART0BAUDTICK
UART1BAUDTICK	output	UART1BAUDTICK
INTMONITOR	output	INTMONITOR
MTXHRESETN	output	SRAM/Flash Chip reset
SRAM0ADDR[12:0]	output	SRAM address
SRAM0WREN[3:0]	output	SRAM Byte write enable
SRAM0WDATA[31:0]	output	SRAM Write data
SRAM0CS	output	SRAM Chip select
TARGFLASH0HSEL	output	TARGFLASH0, HSELx
TARGFLASH0HADDR[28:0]	output	TARGFLASH0, HADDR
TARGFLASH0HTRANS[1:0]	output	TARGFLASH0, HTRANS
TARGFLASH0HWRITE	output	TARGFLASH0, HWRITE
TARGFLASH0HSIZE[2:0]	output	TARGFLASH0, HSIZE
TARGFLASH0HBURST[2:0]	output	TARGFLASH0, HBURST
TARGFLASH0HPROT[3:0]	output	TARGFLASH0, HPROT
TARGFLASH0MEMATTR[1:0]	output	TARGFLASH0, MEMATTR
TARGFLASH0EXREQ	output	TARGFLASH0, EXREQ
TARGFLASH0HMASTER[3:0]	output	TARGFLASH0, HMASTER
TARGFLASH0HWDATA[31:0]	output	TARGFLASH0, HWDATA
TARGFLASH0HMASTLOCK	output	TARGFLASH0, HMASTLOCK
TARGFLASH0HREADYMUX	output	TARGFLASH0, HREADYOUT
TARGFLASH0HAUSER	output	TARGFLASH0, HAUSER
TARGFLASH0HWUSER[3:0]	output	TARGFLASH0, HWUSER
TARGEXP0HSEL	output	TARGEXP0, HSELx
TARGEXP0HADDR[31:0]	output	TARGEXP0, HADDR
TARGEXP0HTRANS[1:0]	output	TARGEXP0, HTRANS
TARGEXP0HWRITE	output	TARGEXP0, HWRITE
TARGEXP0HSIZE[2:0]	output	TARGEXP0, HSIZE
TARGEXP0HBURST[2:0]	output	TARGEXP0, HBURST
TARGEXP0HPROT[3:0]	output	TARGEXP0, HPROT
TARGEXP0MEMATTR[1:0]	output	TARGEXP0, MEMATTR
TARGEXP0EXREQ	output	TARGEXP0, EXREQ
TARGEXP0HMASTER[3:0]	output	TARGEXP0, HMASTER
TARGEXP0HWDATA[31:0]	output	TARGEXP0, HWDATA
TARGEXP0HMASTLOCK	output	TARGEXP0, HMASTLOCK
TARGEXP0HREADYMUX	output	TARGEXP0, HREADYOUT
TARGEXP0HAUSER	output	TARGEXP0, HAUSER
TARGEXP0HWUSER[3:0]	output	TARGEXP0, HWUSER
INITEXP0HRDATA[31:0]	output	INITEXP0, HRDATA

Port Name	I/O	Description
INITEXP0HREADY	output	INITEXP0, HREADY
INITEXP0HRESP	output	INITEXP0, HRESP
INITEXP0EXRESP	output	INITEXP0, EXRESP
INITEXP0HRUSER[2:0]	output	INITEXP0, HRUSER
APBTARGEXP2PSTRB[3:0]	output	APBTARGEXP2, PSTRB
APBTARGEXP2PPROT[2:0]	output	APBTARGEXP2, PPROT
APBTARGEXP2PSEL	output	APBTARGEXP2, PSELx
APBTARGEXP2PENABLE	output	APBTARGEXP2, PENABLE
APBTARGEXP2PADDR[11:0]	output	APBTARGEXP2, PADDR
APBTARGEXP2PWRITE	output	APBTARGEXP2, PWRITE
APBTARGEXP2PWDATA[31:0]	output	APBTARGEXP2, PWDATA
DAPSWDO	output	Serial Wire Data Out
DAPSWDOEN	output	Serial Wire Output Enable
DAPTDO	output	Debug TDO
DAPJTAGNSW	output	JTAG or Serial-Wire selection JTAG mode(1) or SW mode(0)
DAPNTDOEN	output	TDO output pad control signal
TPIUTRACEDATA[3:0]	output	Output data
TPIUTRACESWO	output	Serial Wire Viewer data
TPIUTRACECLK	output	Output clock, used by the TPA to sample the other pins

### 原语例化

#### Verilog 例化:

```
MCU u_sse050_top_syn (
    .FCLK(fclk),
    .PORESETN(poresetn),
    .SYSRESETN(sysresetn),
    .RTCSRCLK(rtcsrclk),
    .IOEXPINPUTI(ioexpinputi[15:0]),
    .IOEXPOUTPUTO(ioexpoutputo[15:0]),
    .IOEXPOUTPUTENO(ioexpoutputeno[15:0]),
    .UART0RXDI(uart0rxdi),
    .UART0TXDO(uart0txdo),
    .UART1RXDI(uart1rxdi),
    .UART1TXDO(uart1txdo),
    .SRAM0RDATA(sram0rdata[31:0]),
    .SRAM0ADDR(sram0addr[12:0]),
    .SRAM0WREN(sram0wren[3:0]),
    .SRAM0WDATA(sram0wdata[31:0]),
    .SRAM0CS(sram0cs),
    .MTXHRESETN(mtxhreset),
    .TARGFLASH0HSEL(targflash0hsel),
    .TARGFLASH0HADDR(targflash0haddr[28:0]),
    .TARGFLASH0HTRANS(targflash0htrans[1:0]),
    .TARGFLASH0HWRITE(targflash0hwrite),
```

.TARGFLASH0HSIZE(targflash0hsize[2:0]),  
.TARGFLASH0HBURST(targflash0hburst[2:0]),  
.TARGFLASH0HPROT(targflash0hprot[3:0]),  
.TARGFLASH0MEMATTR(targflash0memattr[1:0]),  
.TARGFLASH0EXREQ(targflash0exreq),  
.TARGFLASH0HMASTER(targflash0hmaster[3:0]),  
.TARGFLASH0HWDATA(targflash0hwdata[31:0]),  
.TARGFLASH0HMASTLOCK(targflash0hmastlock),  
.TARGFLASH0HREADYMUX(targflash0hreadymux),  
.TARGFLASH0HAUSER(targflash0hauser),  
.TARGFLASH0HWUSER(targflash0hwuser[3:0]),  
.TARGFLASH0HRDATA(targflash0hrdata[31:0]),  
.TARGFLASH0HRUSER(targflash0hruser[2:0]),  
.TARGFLASH0HRESP(targflash0hresp),  
.TARGFLASH0EXRESP(targflash0exresp),  
.TARGFLASH0HREADYOUT(targflash0hreadyout),  
.TARGEXP0HSEL(targexp0hssel),  
.TARGEXP0HADDR(targexp0haddr[31:0]),  
.TARGEXP0HTRANS(targexp0htrans[1:0]),  
.TARGEXP0HWRITE(targexp0hwrite),  
.TARGEXP0HSIZE(targexp0hsize[2:0]),  
.TARGEXP0HBURST(targexp0hburst[2:0]),  
.TARGEXP0HPROT(targexp0hprot[3:0]),  
.TARGEXP0MEMATTR(targexp0memattr[1:0]),  
.TARGEXP0EXREQ(targexp0exreq),  
.TARGEXP0HMASTER(targexp0hmaster[3:0]),  
.TARGEXP0HWDATA(targexp0hwdata[31:0]),  
.TARGEXP0HMASTLOCK(targexp0hmastlock),  
.TARGEXP0HREADYMUX(targexp0hreadymux),  
.TARGEXP0HAUSER(targexp0hauser),  
.TARGEXP0HWUSER(targexp0hwuser[3:0]),  
.TARGEXP0HRDATA(targexp0hrdata[31:0]),  
.TARGEXP0HREADYOUT(targexp0hreadyout),  
.TARGEXP0HRESP(targexp0hresp),  
.TARGEXP0EXRESP(targexp0exresp),  
.TARGEXP0HRUSER(targexp0hruser[2:0]),  
.INITEXP0HSEL(initexp0hssel),  
.INITEXP0HADDR(initexp0haddr[31:0]),  
.INITEXP0HTRANS(initexp0htrans[1:0]),  
.INITEXP0HWRITE(initexp0hwrite),  
.INITEXP0HSIZE(initexp0hsize[2:0]),  
.INITEXP0HBURST(initexp0hburst[2:0]),  
.INITEXP0HPROT(initexp0hprot[3:0]),  
.INITEXP0MEMATTR(initexp0memattr[1:0]),  
.INITEXP0EXREQ(initexp0exreq),  
.INITEXP0HMASTER(initexp0hmaster[3:0]),  
.INITEXP0HWDATA(initexp0hwdata[31:0]),  
.INITEXP0HMASTLOCK(initexp0hmastlock),  
.INITEXP0HAUSER(initexp0hauser),  
.INITEXP0HWUSER(initexp0hwuser[3:0]),

```

.INITEXP0HRDATA(initexp0hrdata[31:0]),
.INITEXP0HREADY(initexp0hready),
.INITEXP0HRESP(initexp0hresp),
.INITEXP0EXRESP(initexp0exresp),
.INITEXP0HRUSER(initexp0hruser[2:0]),
.APBTARGEXP2PSEL(apbtargexp2psel),
.APBTARGEXP2PENABLE(apbtargexp2penable),
.APBTARGEXP2PADDR(apbtargexp2paddr[11:0]),
.APBTARGEXP2PWRITE(apbtargexp2pwrite),
.APBTARGEXP2PWDATA(apbtargexp2pwdata[31:0]),
.APBTARGEXP2PRDATA(apbtargexp2prdata[31:0]),
.APBTARGEXP2PREADY(apbtargexp2pready),
.APBTARGEXP2PSLVERR(apbtargexp2pslverr),
.APBTARGEXP2PSTRB(apbtargexp2pstrb[3:0]),
.APBTARGEXP2PPROT(apbtargexp2pprot[2:0]),
.MTXREMAP(mtxremap[3:0]),
.DAPSWDITMS(dapswditms),
.DAPSWDO(dapswdo),
.DAPSWDOEN(dapswdoen),
.DAPTDI(daptdi),
.DAPTDO(daptdo),
.DAPNTRST(dapntrst),
.DAPSWCLKTCK(dapswclk_tck),
.DAPNTDOEN(dapntdoen),
.DAPJTAGNSW(dapjtagns),
.TPIUTRACEDATA(tpiutracedata[3:0]),
.TPIUTRACESWO(tpiutraceswo),
.TPIUTRACECLK(tpiutracedclk),
.FLASHERR(flasherr),
.FLASHINT(flashint)
);

```

**Vhdl 例化:**

```

COMPONENT MCU
  PORT(
    FCLK:IN std_logic;
    PORESETN:IN std_logic;
    SYSRESETN:IN std_logic;
    RTCSRCLK:IN std_logic;
    UART0RXDI:IN std_logic;
    UART1RXDI:IN std_logic;
    CLK:IN std_logic;
    RESET:IN std_logic;
    IOEXPINPUTI:IN std_logic_vector(15 downto 0);
    SRAM0RDATA:IN std_logic_vector(31 downto 0);
    TARGFLASH0HRDATA:IN std_logic_vector(31 downto 0);
    TARGFLASH0HRUSER:IN std_logic_vector(2 downto 0);
    TARGFLASH0HRESP:IN std_logic;
    TARGFLASH0EXRESP:IN std_logic;
    TARGFLASH0HREADYOUT:IN std_logic;
    TARGEXP0HRDATA: IN std_logic_vector(31 downto 0);

```

TARGEXP0HREADYOUT:IN std\_logic;  
TARGEXP0HRESP:IN std\_logic;  
TARGEXP0EXRESP:IN std\_logic;  
TARGEXP0HRUSER: IN std\_logic\_vector(2 downto 0);  
INITEXP0HSEL:IN std\_logic;  
INITEXP0HADDR: IN std\_logic\_vector(31 downto 0);  
INITEXP0HTRANS: IN std\_logic\_vector(1 downto 0);  
INITEXP0HWRITE: IN std\_logic;  
INITEXP0HSIZE: IN std\_logic\_vector(2 downto 0);  
INITEXP0HBURST: IN std\_logic\_vector(2 downto 0);  
INITEXP0HPROT: IN std\_logic\_vector(3 downto 0);  
INITEXP0MEMATTR: IN std\_logic\_vector(1 downto 0);  
INITEXP0EXREQ: IN std\_logic;  
INITEXP0HMASTER: IN std\_logic\_vector(3 downto 0);  
INITEXP0HWDATA: IN std\_logic\_vector(31 downto 0);  
INITEXP0HMASTLOCK: IN std\_logic;  
INITEXP0HAUSER: IN std\_logic;  
INITEXP0HWUSER: IN std\_logic\_vector(3 downto 0);  
APBTARGEXP2PRDATA: IN std\_logic\_vector(3 downto 0);  
APBTARGEXP2PREADY: IN std\_logic;  
APBTARGEXP2PSLVERR: IN std\_logic;  
MTXREMAP: IN std\_logic\_vector(3 downto 0);  
DAPSWDITMS: IN std\_logic;  
DAPTDI: IN std\_logic;  
DAPNTRST: IN std\_logic;  
DAPSWCLKTCK: IN std\_logic;  
FLASHERR: IN std\_logic;  
FLASHINT: IN std\_logic;  
IOEXPOUTPUTO:OUT std\_logic\_vector(15 downto 0);  
IOEXPOUTPUTENO:OUT std\_logic\_vector(15 downto 0);  
IOEXPINPUTI:OUT std\_logic\_vector(15 downto 0);  
UART0TXDO: OUT std\_logic;  
UART1TXDO: OUT std\_logic;  
UART0BAUDTICK: OUT std\_logic;  
UART1BAUDTICK: OUT std\_logic;  
INTMONITOR: OUT std\_logic;  
MTXHRESETN: OUT std\_logic;  
SRAM0ADDR:OUT std\_logic\_vector(12 downto 0);  
SRAM0WREN:OUT std\_logic\_vector(3 downto 0);  
SRAM0WDATA:OUT std\_logic\_vector(31 downto 0);  
SRAM0CS: OUT std\_logic;  
TARGFLASH0HSEL: OUT std\_logic;  
TARGFLASH0HWRITE: OUT std\_logic;  
TARGFLASH0EXREQ: OUT std\_logic;  
TARGFLASH0HMASTLOCK: OUT std\_logic;  
TARGFLASH0HREADYMUX: OUT std\_logic;  
TARGFLASH0HAUSER: OUT std\_logic;  
SRAM0RDATA:OUT std\_logic\_vector(31 downto 0);  
TARGFLASH0HADDR:OUT std\_logic\_vector(28 downto 0);  
TARGFLASH0HTRANS:OUT std\_logic\_vector(1 downto 0);



```

TARGFLASH0HSIZE:OUT std_logic_vector(2 downto 0);
TARGFLASH0HBURST:OUT std_logic_vector(2 downto 0);
TARGFLASH0HPROT:OUT std_logic_vector(3 downto 0);
TARGFLASH0MEMATTR:OUT std_logic_vector(1 downto 0);
TARGFLASH0HMASTER:OUT std_logic_vector(3 downto 0);
TARGFLASH0HWDATA:OUT std_logic_vector(31 downto 0);
TARGFLASH0HWUSER:OUT std_logic_vector(3 downto 0);
TARGFLASH0HRDATA:OUT std_logic_vector(31 downto 0);
TARGEXP0HADDR:OUT std_logic_vector(31 downto 0);
TARGEXP0HSEL: OUT std_logic;
TARGEXP0HWRITE: OUT std_logic;
TARGEXP0EXREQ: OUT std_logic;
TARGEXP0HMASTLOCK: OUT std_logic;
TARGEXP0HREADYMUX: OUT std_logic;
TARGEXP0HAUSER: OUT std_logic;
INITEXP0HREADY: OUT std_logic;
INITEXP0HRESP: OUT std_logic;
INITEXP0EXRESP: OUT std_logic;
TARGEXP0HTRANS:OUT std_logic_vector(1 downto 0);
TARGEXP0HSIZE:OUT std_logic_vector(2 downto 0);
TARGEXP0HBURST:OUT std_logic_vector(2 downto 0);
TARGEXP0HPROT:OUT std_logic_vector(3 downto 0);
TARGEXP0MEMATTR:OUT std_logic_vector(1 downto 0);
TARGEXP0HMASTER:OUT std_logic_vector(3 downto 0);
TARGEXP0HWDATA:OUT std_logic_vector(31 downto 0);
TARGEXP0HWUSER:OUT std_logic_vector(3 downto 0);
INITEXP0HRDATA:OUT std_logic_vector(31 downto 0);
INITEXP0HRUSER:OUT std_logic_vector(2 downto 0);
APBTARGEXP2PSTRB:OUT std_logic_vector(3 downto 0);
APBTARGEXP2PPROT:OUT std_logic_vector(2 downto 0);
APBTARGEXP2PADDR:OUT std_logic_vector(11 downto 0);
APBTARGEXP2PWDATA:OUT std_logic_vector(31 downto 0);
TPIUTRACEDATA:OUT std_logic_vector(3 downto 0);
APBTARGEXP2PSEL: OUT std_logic;
APBTARGEXP2PENABLE: OUT std_logic;
APBTARGEXP2PWRITE: OUT std_logic;
DAPSWDO: OUT std_logic;
DAPSWDOEN: OUT std_logic;
DAPTD0: OUT std_logic;
DAPJTAGNSW: OUT std_logic;
DAPNTDOEN: OUT std_logic;
TPIUTRACESWO: OUT std_logic;
TPIUTRACECLK: OUT std_logic;
);
END COMPONENT;

 uut: MCU
    PORT MAP (
      FCLK=> fclk;
      PORESETN=> poresetn;

```

SYSRESETN=> sysresetn;  
RTCSRCLK=> rtcsrclk;  
UART0RXDI=> uart0rxdi;  
UART1RXDI=> uart1rxdi;  
CLK=>clk,  
RESET=>reset,  
IOEXPINPUTI=>ioexpinputi,  
SRAM0RDATA=>sram0rdata,  
TARGFLASH0HRDATA=>targflash0hrdata,  
TARGFLASH0HRUSER=>targflash0hruser,  
TARGFLASH0HRESP=>targflash0hresp,  
TARGFLASH0EXRESP=>targflash0exresp,  
TARGFLASH0HREADYOUT=>targflash0hreadyout,  
TARGEXP0HRDATA=>targexp0hrdata,  
TARGEXP0HREADYOUT=>targexp0hreadyout,  
TARGEXP0HRESP=>targexp0hresp,  
TARGEXP0EXRESP=>targexp0exresp,  
TARGEXP0HRUSER=>targexp0hruser,  
INITEXP0HSEL=>initexp0hsel,  
INITEXP0HADDR=>initexp0haddr,  
INITEXP0HTRANS=>initexp0htrans,  
INITEXP0HWRITE=>initexp0hwrite,  
INITEXP0HSIZE=>initexp0hsize,  
INITEXP0HBURST=>initexp0hburst,  
INITEXP0HPROT=>initexp0hprot,  
INITEXP0MEMATTR=>initexp0memattr,  
INITEXP0EXREQ=>initexp0exreq,  
INITEXP0HMASTER=>initexp0hmaster,  
INITEXP0HWDATA=>initexp0hwdata,  
INITEXP0HMASTLOCK=>initexp0hmastlock,  
INITEXP0HAUSER=>initexp0hauser,  
INITEXP0HWUSER=>initexp0hwuser,  
APBTARGEXP2PRDATA=>apbtargexp2prdata,  
APBTARGEXP2PREADY=>apbtargexp2pready,  
APBTARGEXP2PSLVERR=>apbtargexp2pslverr,  
MTXREMAP=>mtxremap,  
DAPSWDITMS=>dapswditms,  
DAPTDI=>daptidi,  
DAPNTRST=>dapntrst,  
DAPSWCLKTCK=>dapswclktck,  
FLASHERR=>flasherr,  
FLASHINT=>flashint,  
IOEXPOUTPUTO=>ioexpoutputo,  
IOEXPOUTPUTENO=>ioexpoutputeno,  
IOEXPINPUTI=>ioexpinputi,  
UART0TXDO=>uart0txdo,  
UART1TXDO=>uart1txdo,  
UART0BAUDTICK=>uart0baudtick,  
UART1BAUDTICK=>uart1baudtick,  
INTMONITOR=>intmonitor,

MTXHRESETN=>mtxhresetn,  
SRAM0ADDR=>sram0addr,  
SRAM0WREN=>sram0wren,  
SRAM0WDATA=>sram0wdata,  
SRAM0CS=>sram0cs,  
TARGFLASH0HSEL=>targflash0hssel,  
TARGFLASH0HWRITE=>targflash0hwrite,  
TARGFLASH0EXREQ=>targflash0exreq,  
TARGFLASH0HMASTLOCK=>targflash0hmastlock,  
TARGFLASH0HREADYMUX=>targflash0hreadymux,  
TARGFLASH0HAUSER=>targflash0hauser,  
SRAM0RDATA=>sram0rdata,  
TARGFLASH0HADDR=>targflash0haddr,  
TARGFLASH0HTRANS=>targflash0htrans,  
TARGFLASH0HSIZE=>targflash0hsize,  
TARGFLASH0HBURST=>targflash0hburst,  
TARGFLASH0HPROT=>targflash0hprot,  
TARGFLASH0MEMATTR=>targflash0memattr,  
TARGFLASH0HMASTER=>targflash0hmaster,  
TARGFLASH0HWDATA=>targflash0hwdata,  
TARGFLASH0HWUSER=>targflash0hwuser,  
TARGFLASH0HRDATA=>targflash0hrdata,  
TARGEXP0HADDR=>targexp0haddr,  
TARGEXP0HSEL=>targexp0hssel,  
TARGEXP0HWRITE=>targexp0hwrite,  
TARGEXP0EXREQ=>targexp0exreq,  
TARGEXP0HMASTLOCK=>targexp0hmastlock,  
TARGEXP0HREADYMUX=>targexp0hreadymux,  
TARGEXP0HAUSER=>targexp0hauser,  
INITEXP0HREADY=>initexp0hready,  
INITEXP0HRESP=>initexp0hresp,  
INITEXP0EXRESP=>initexp0exresp,  
TARGEXP0HTRANS=>targexp0htrans,  
TARGEXP0HSIZE=>targexp0hsize,  
TARGEXP0HBURST=>targexp0hburst,  
TARGEXP0HPROT=>targexp0hprot,  
TARGEXP0MEMATTR=>targexp0memattr,  
TARGEXP0HMASTER=>targexp0hmaster,  
TARGEXP0HWDATA=>targexp0hwdata,  
TARGEXP0HWUSER=>targexp0hwuser,  
INITEXP0HRDATA=>initexp0hrdata,  
INITEXP0HRUSER=>initexp0hruser,  
APBTARGEXP2PSTRB=>apbtargexp2pstrb,  
APBTARGEXP2PPROT=>apbtargexp2pprot,  
APBTARGEXP2PADDR=>apbtargexp2paddr,  
APBTARGEXP2PWDATA=>apbtargexp2pwdata,  
TPIUTRACEDATA=>tpiutracedata,  
APBTARGEXP2PSEL=>apbtargexp2psel,  
APBTARGEXP2PENABLE=>apbtargexp2penable,  
APBTARGEXP2PWRITE=>apbtargexp2pwrite,

```

DAPSWDO=>dapswdo,
DAPSWDOEN=>dapswdoen,
DAPTD0=>daptdo,
DAPJTAGNSW=>dapjtagsw,
DAPNTDOEN=>dapntdoen,
TPIUTRACESWO=>tpiutraceswo,
TPIUTRACECLK=>tpiutraceclk );

```

## 8.2 USB20\_PHY

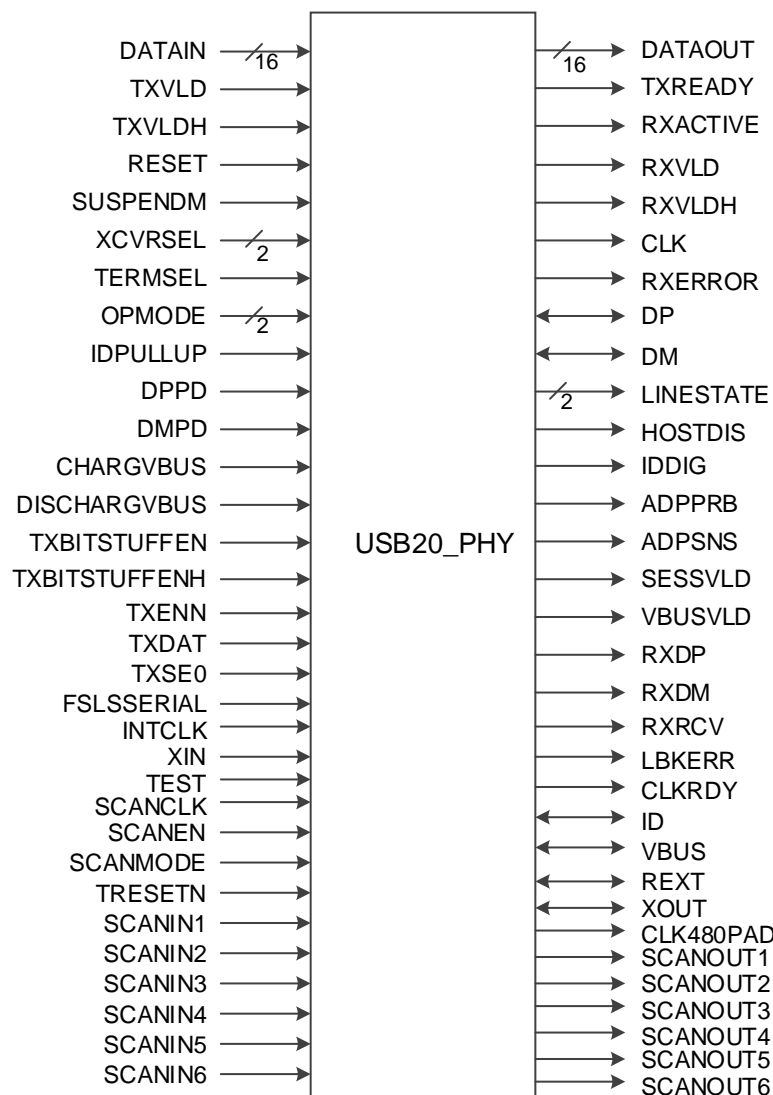
### 原语介绍

USB20\_PHY 是完整的混合信号 IP 解决方案，实现从 Soc（System-on-Chip）到其他特殊制造工艺的 OTG 连接。USB20\_PHY 支持 USB 2.0 480-Mbps 的协议和数据速率，并且后向兼容 USB 1.1 1.5-Mbps 和 12-Mbps 的协议和数据速率。

支持器件：GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C。

### 结构框图

图 8-2 USB20\_PHY 结构框图



## Port 介绍

表 8-2 Port 介绍

Port Name	I/O	Description
DATAIN[15:0]	input	16-bit parallel USB data input bus
TXVLD	input	Transmit Valid. Indicates that the DataIn bus is valid.
TXVLDH	input	Transmit Valid High. When DataBus16_8 = 1, this signal indicates that the DataIn[15:8] bus contains valid transmit data.
RESET	input	Reset. Reset all state machines in the UTM.
SUSPENDM	input	Suspend. 0:suspend, 1: normal
XCVRSEL[1:0]	input	Transceiver Select. This signal selects between the LS, FS and HS transceivers
TERMSEL	input	Termination Select. This signal selects between the FS and HS terminations
OPMODE[1:0]	input	Operational Mode. These signals select between various operational modes
IDPULLUP	input	Signal that enables the sampling of the analog Id line.
DPPD	input	This signal enables the 15k Ohm pull-down resistor on the DP line.
DMPD	input	0b : Pull-down resistor not connected to DM; 1b : Pull-down resistor connected to DM
CHARGVBUS	input	This signal enables charging Vbus
DISCHARGVBUS	input	The signal enables discharging Vbus.
TXBITSTUFFEN	input	Indicates if the data on the DataOut[7:0] lines needs to be bitstuffed or not.
TXBITSTUFFENH	input	Indicates if the data on the DataOut[15:8] lines needs to be bitstuffed or not.
TXENN	input	Active low enable signal. Only used when FsLsSerialMode is set to 1b
TXDAT	input	Differential data at D+/D- output. Only used when FsLsSerialMode is set to 1b
TXSE0	input	Force Single-Ended Zero. Only used when FsLsSerialMode is set to 1b
FSLSSERIAL	input	0b : FS and LS packets are sent using the parallel interface. 1b : FS and LS packets are sent using the serial interface.
INTCLK	input	Clock signals provided internally of the SoC
TEST	input	For IP TESTING purpose. Please leave it unconnected since there are already soft pull-down in the IP
SCANCLK	input	Clock signals for scan mode
SCANEN	input	Select to shift mode
SCANMODE	input	High effective signal to enter scan mode
TRESETN	input	Low effective RESET signal for scan mode
SCANIN1	input	Scan chain input
SCANIN2	input	Scan chain input
SCANIN3	input	Scan chain input
SCANIN4	input	Scan chain input
SCANIN5	input	Scan chain input
SCANIN6	input	Scan chain input
DP	inout	USB data pin Data+
DM	inout	USB data pin Data-
ID	inout	ID signal from the cable
VBUS	inout	Vbus signals connected with the cable
REXT	inout	12.7K High precision resistor

Port Name	I/O	Description
XIN	inout	Crystal in signals, supported range is 12MHZ~24MHZ
XOUT	inout	Crystal out signals
DATAOUT[15:0]	output	DataOut. 16-bit parallel USB data output bus.
TXREADY	output	Transmit Data Ready.
RXACTIVE	output	Receive Active. Indicates that the receive state machine has detected SYNC and is active.
RXVLD	output	Receive Data Valid. Indicates that the DataOut bus has valid data.
RXVLDH	output	Receive Data Valid High.
CLK	output	Clock. This output is used for clocking receive and transmit parallel data.
RXERROR	output	Receive Error.
LINESTATE[1:0]	output	Line State. These signals reflect the current state of the single ended receivers.
HOSTDIS	output	This signal is used for all types of peripherals connected to it.
IDDIG	output	Indicates whether the connected plug is a mini-A or mini-B.
ADPPRB	output	Indicates if the voltage on Vbus ( $0.6V < V_{th} < 0.75V$ ).
ADPSNS	output	Indicates if the voltage on Vbus ( $0.2V < V_{th} < 0.55V$ ).
SESSVLD	output	Indicates if the session for an A/B-peripheral is valid ( $0.8V < V_{th} < 2V$ ).
VBUSVLD	output	Indicates if the voltage on Vbus is at a valid level for operation ( $4.4V < V_{th} < 4.75V$ )
RXDP	output	Single-ended receive data, positive terminal.This signal is only valid if FsLsSerialMode is set to 1b
RXDM	output	Single-ended receive data, negative terminal.This signal is only valid if FsLsSerialMode is set to 1b
RXRCV	output	Receive data.This signal is only valid if FsLsSerialMode is set to 1b
LBKERR	output	used for observation
CLKRDY	output	Observation/debug signal to show that the internal PLL has locked and is ready.
CLK480PAD	output	480MHZ clock output for observation
SCANOUT1	output	Scan chain output
SCANOUT2	output	Scan chain output
SCANOUT3	output	Scan chain output
SCANOUT4	output	Scan chain output
SCANOUT5	output	Scan chain output
SCANOUT6	output	Scan chain output

## Attribute 介绍

表 8-3 Attribute 介绍

Attribute Name	Default	Description
DATABUS16_8	1'b0	Selects between 8 and 16 bit data transfers.
ADP_PRBEN	1'b0	Enables/disables the ADP Probe comparator
TEST_MODE	5'b0	used for testing and debugging purpose
HSDRV1	1'b0	High speed drive adjustment. Please connect to 0 for normal operation.
HSDRV0	1'b0	High speed drive adjustment. Please connect to 0 for normal operation.
CLK_SEL	1'b0	Clock source selection signal. 0 to select external clock provided by the crystal connected on XIN, XOUT. 1 to select internal clock provided on INTCLK port
M	4'b0	M divider input data bit
N	6'b101000	N divider input data bit
C	2'b01	Control charge pump current input data bit, it supports from 30uA (00) to 60uA (11).
FOC_LOCK	1'b0	0: LOCK is generated by PLL lock detector. 1: LOCK is always high(always lock)

## 原语例化

### Verilog 例化:

```

USB20_PHY usb20_phy_inst (
    .DATAOUT(dataout[15:0]),
    .TXREADY(txready),
    .RXACTIVE(rxactive),
    .RXVLD(rxvld),
    .RXVLDH(rxvldh),
    .CLK(clk),
    .RXERROR(rxerror),
    .DP(dp),
    .DM(dm),
    .LINESTATE(linestate[1:0]),
    .DATAIN(datain[15:0]),
    .TXVLD(txvld),
    .TXVLDH(txvldh),
    .RESET(reset),
    .SUSPENDM(suspendm),
    .XCVRSEL(xcvrsel[1:0]),
    .TERMSEL(termsel),
    .OPMODE(opmode[1:0]),
    .HOSTDIS(hostdis),
    .IDDIG(iddig),
    .ADPPRB(adpprb),
    .ADPSNS(adpsns),
    .SESSVLD(sessvld),
    .VBUSVLD(vbusvld),
    .RXDP(rxdp),
    .RXDM(rxdm),

```

```

.RXRCV(rxrcv),
.IDPULLUP(idpullup),
.DPPD(dppd),
.DMPD(dmpd),
.CHARGVBUS(chargvbus),
.DISCHARGVBUS(dischargvbus),
.TXBITSTUFFEN(txbitstufen),
.TXBITSTUFFENH(txbitstuffenh),
.TXENN(txenn),
.TXDAT(txdat),
.TXSE0(txse0),
.FSLSSERIAL(fslsserial),
.LBKERR(lbkerr),
.CLKRDY(clkrdy),
.INTCLK(intclk),
.ID(id),
.VBUS(vbus),
.REXT(rent),
.XIN(xin),
.XOUT(xout),
.CLK480PAD(clk480pad),
.TEST(test),
.SCANOUT1(scanout1),
.SCANOUT2(scanout2),
.SCANOUT3(scanout3),
.SCANOUT4(scanout4),
.SCANOUT5(scanout5),
.SCANOUT6(scanout6),
.SCANCLK(scanclk),
.SCANEN(scanen),
.SCANMODE(scanmode),
.TRESETN(tresetn),
.SCANIN1(scanin1),
.SCANIN2(scanin2),
.SCANIN3(scanin3),
.SCANIN4(scanin4),
.SCANIN5(scanin5),
.SCANIN6(scanin6)
);
defparam usb20_phy_inst.DATABUS16_8 = 1'b0;
defparam usb20_phy_inst.ADP_PRBEN = 1'b0;
defparam usb20_phy_inst.TEST_MODE = 5'b0;;
defparam usb20_phy_inst.HSDRV1 = 1'b0;
defparam usb20_phy_inst.HSDRV0 = 1'b0;
defparam usb20_phy_inst.CLK_SEL = 1'b0;
defparam usb20_phy_inst.M = 4'b0;
defparam usb20_phy_inst.N = 6'b101000;
defparam usb20_phy_inst.C = 2'b01;
defparam usb20_phy_inst.FOC_LOCK = 1'b0;

```



**Vhdl 例化:**

```

COMPONENT USB20_PHY
  GENERIC (
    TEST_MODE:bit_vector:="00000";
    DATABUS16_8:bit:='0';
    ADP_PRBEN:bit:='0';
    HSDRV1:bit:='0';
    HSDRV0:bit:='0';
    CLK_SEL:bit:='0';
    M:bit_vector:="0000";
    N:bit_vector:=" 101000";
    C:bit_vector:="01";
    FOC_LOCK:bit:='0';
  );
PORT(
  DATAIN:IN std_logic_vector(15 downto 0);
  TXVLD:IN std_logic;
  TXVLDH:IN std_logic;
  RESET:IN std_logic;
  SUSPENDM:IN std_logic;
  XCVRSEL:IN std_logic_vector(1 downto 0);
  TERMSEL:IN std_logic;
  OPMODE:IN std_logic_vector(1 downto 0);
  DATAOUT:OUT std_logic_vector(15 downto 0);
  TXREADY:OUT std_logic;
  RXACTIVE:OUT std_logic;
  RXVLD:OUT std_logic;
  RXVLDH:OUT std_logic;
  CLK:OUT std_logic;
  RXERROR:OUT std_logic;
  DP:INOUT std_logic;
  DM:INOUT std_logic;
  LINESTATE:OUT std_logic_vector(1 downto 0);
  IDPULLUP:IN std_logic;
  DPPD:IN std_logic;
  DMPD:IN std_logic;
  CHARGVBUS:IN std_logic;
  DISCHARGVBUS:IN std_logic;
  TXBITSTUFFEN:IN std_logic;
  TXBITSTUFFENH:IN std_logic;
  TXENN:IN std_logic;
  TXDAT:IN std_logic;
  TXSE0:IN std_logic;
  FSLSSERIAL:IN std_logic;
  HOSTDIS:OUT std_logic;
  IDDIG:OUT std_logic;
  ADPPRB:OUT std_logic;
  ADPSNS:OUT std_logic;
  SESSVLD:OUT std_logic;
  VBUSVLD:OUT std_logic;

```

```

RXDP:OUT std_logic;
RXDM:OUT std_logic;
RXRCV:OUT std_logic;
LBKERR:OUT std_logic;
CLKRDY:OUT std_logic;
INTCLK:IN std_logic;
ID:INOUT std_logic;
VBUS:INOUT std_logic;
REXT:INOUT std_logic;
XIN:IN std_logic;
XOUT:INOUT std_logic;
TEST:IN std_logic;
CLK480PAD:OUT std_logic;
SCANCLK:IN std_logic;
SCANEN:IN std_logic;
SCANMODE:IN std_logic;
TRESETN:IN std_logic;
SCANIN1:IN std_logic;
SCANOUT1:OUT std_logic;
SCANIN2:IN std_logic;
SCANOUT2:OUT std_logic;
SCANIN3:IN std_logic;
SCANOUT3:OUT std_logic;
SCANIN4:IN std_logic;
SCANOUT4:OUT std_logic;
SCANIN5:IN std_logic;
SCANOUT5:OUT std_logic;
SCANIN6:IN std_logic;
SCANOUT6:OUT std_logic;
);
END COMPONENT;
 uut: USB20_PHY
    PORT MAP (
        DATAIN=>datain,
        TXVLD=>txvld,
        TXVLDH=>txvldh,
        RESET=>reset,
        SUSPENDM=>suspendm,
        XCVRSEL=>xcvrssel,
        TERMSEL=>termssel,
        OPMODE=>opmode,
        DATAOUT=>dataout,
        TXREADY=>txready,
        RXACTIVE=>rxactive,
        RXVLD=>rxvld,
        RXVLDH=>rxvldh,
        CLK=>clk,
        RXERROR=>rxerror,
        DP=>dp,
        DM=>dm,

```

```
LINESTATE=>linestate,  
IDPULLUP=>idpullup,  
DPPD=>dppd,  
DMPD=>dmpd,  
CHARGVBUS=>chargvbus,  
DISCHARGVBUS=>dischargvbus,  
TXBITSTUFFEN=>txbitstufen,  
TXBITSTUFFENH=>txbitstuffenh,  
TXENN=>txenn,  
TXDAT=>txdat,  
TXSE0=>txse0,  
FSLSSERIAL=>fslsserial,  
HOSTDIS=>hostdis,  
IDDIG=>iddig,  
ADPPRB=>adpprb,  
ADPSNS=>adpsns,  
SESSVLD=>sessvld,  
VBUSVLD=>vbusvld,  
RXDP=>rxdp,  
RXDM=>rxdm,  
RXRCV=>rxrcv,  
LBKERR=>lbkerr,  
CLKRDY=>clkrdy,  
INTCLK=>intclk,  
ID=>id,  
VBUS=>vbus,  
REXT=>rext,  
XIN=>xin,  
XOUT=>xout,  
TEST=>test,  
CLK480PAD=>clk480pad,  
SCANCLK=>scanclk,  
SCANEN=>scanen,  
SCANMODE=>scanmode,  
TRESETN=>tresetn,  
SCANIN1=>scanin1,  
SCANOUT1=>scanout1,  
SCANIN2=>scanin2,  
SCANOUT2=>scanout2,  
SCANIN3=>scanin3,  
SCANOUT3=>scanout3,  
SCANIN4=>scanin4,  
SCANOUT4=>scanout4,  
SCANIN5=>scanin5,  
SCANOUT5=>scanout5,  
SCANIN6=>scanin6,  
SCANOUT6=>scanout6  
);
```

## 8.3 ADC

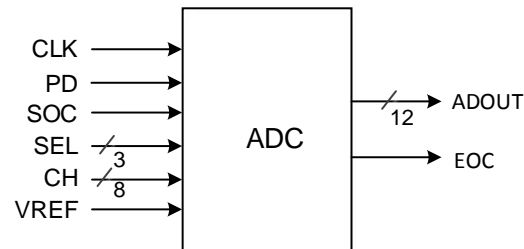
### 原语介绍

ADC (Analog-to-digital Converter) 是一个 8 通道单端 12 位的模数转换器，具有低功耗、低漏电和高动态特性。

支持器件：GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C。

### 结构框图

图 8-3 ADC 结构框图



### Port 介绍

表 8-4 Port 介绍

Port Name	I/O	Description
ADOUT[11:0]	Output	ad conversion results.
EOC	Output	end of conversion.
CLK	Input	main clock.
PD	Input	power down signal.
SOC	Input	start of conversion.
SEL[2:0]	Input	channel select signal.
CH[7:0]	Input	channel signal-ended analog voltage input.
VREF	Input	voltage reference

### 原语例化

#### Verilog 例化:

```
ADC adc_inst(
    .CLK(clk),
    .PD(pd),
    .SOC(soc),
    .SEL(sel[2:0]),
    .CH(ch[7:0]),
    .VREF(vref),
    .EOC(eoc),
    .ADOUT(adout[11:0])
);
```

#### Vhdl 例化:

```
COMPONENT ADC
    PORT(
        CLK=>IN std_logic;
        PD=>IN std_logic;
        SOC=>IN std_logic;
```

```
        SEL=>IN std_logic_vector(2 downto 0);
        CH=>IN std_logic_vector(7 downto 0);
        VREF=>IN std_logic;
        EOC=>OUT std_logic;
        ADOUT=>OUT std_logic_vector(11 downto 0)
    );
END COMPONENT;
uut=> ADC
    PORT MAP (
        CLK=>clk,
        PD=>pd,
        SOC=>soc,
        SEL=>sel,
        CH=>ch,
        VREF=>vref,
        EOC=>eoc,
        ADOUT=>adout
    );
```

