|  |  |
| --- | --- |
| ***full\_adder***  entity full\_adder is  port (a,b,cin:in bit; s,cout:out bit);  end full\_adder;  architecture fulladder of full\_adder is  signal x1,x2,x3 :bit;  begin  x1<= a xor b;  s<= x1 xor cin;  x2<= a and b;  x3<= x1 and cin ;  cout <= x2 or x3;  end fulladder ; | ***Half adder***  entity half\_adder is  port(a,b:in bit ; s,cout:out bit);  end half\_adder;  architecture halader of half\_adder is  begin  s<= a xor b;  cout<= a and b;  end halader; |
| ***Muxx 4x1 8bit***  entity mux\_4XX1\_8bit is  port(a,b,c,d:in bit\_vector(7 downto 0);s1,s0:in bit;o :out bit\_vector(7 downto 0));  end mux\_4Xx1\_8bit;  architecture muxx4x1 of mux\_4XX1\_8bit is  component eb  port(a,b,c,d,s1,s2:in bit ; o : out bit);  end component ;  for all :eb use entity work.mux4xx1(mux4x1);  begin  g0: eb port map (a(7),b(7),c(7),d(7),s1,s0,o(7));  g1: eb port map (a(6),b(6),c(6),d(6),s1,s0,o(6));  g2: eb port map (a(5),b(5),c(5),d(5),s1,s0,o(5));  g3: eb port map (a(4),b(4),c(4),d(4),s1,s0,o(4));  g4: eb port map (a(3),b(3),c(3),d(3),s1,s0,o(3));  g5: eb port map (a(2),b(2),c(2),d(2),s1,s0,o(2));  g6: eb port map (a(1),b(1),c(1),d(1),s1,s0,o(1));  g7: eb port map (a(0),b(0),c(0),d(0),s1,s0,o(0));  end muxx4x1; | ***Adder 8 bit***  entity adder\_8 is  port(a,b:in bit\_vector (7 downto 0);cin: in bit; cout : out bit ; s:out bit\_vector(7 downto 0));  end adder\_8 ;  architecture adder8bit of adder\_8 is  component eb is  port (a,b,cin:in bit ;s,cout : out bit );  end component ;  for all : eb use entity work.full\_adder(fulladder);  signal x: bit\_vector(8 downto 1);  begin  g0:eb port map (a(0),b(0),cin,s(0),x(1));  g1:eb port map (a(1),b(1),x(1),s(1),x(2));  g2:eb port map (a(2),b(2),x(2),s(2),x(3));  g3:eb port map (a(3),b(3),x(3),s(3),x(4));  g4:eb port map (a(4),b(4),x(4),s(4),x(5));  g5:eb port map (a(5),b(5),x(5),s(5),x(6));  g6:eb port map (a(6),b(6),x(6),s(6),x(7));  g7:eb port map (a(7),b(7),x(7),s(7),cout);  end adder8bit; |
| ضرب کننده دوبیتی  entity multiply is  port (A, B : in bit\_vector(1 downto 0);  P : out bit\_vector(3 downto 0)  );  end multiply;  architecture dataflow of multiply is  begin  P(0) <= A(0) AND B(0);  P(1) <= (A(1) AND B(0)) XOR (A(0) AND B(1));  P(2) <= ((A(1) AND B(0)) AND (A(0) AND B(1))) XOR (A(1) AND B(1));  P(3) <= ((A(1) AND B(0)) AND (A(0) AND B(1))) AND (A(1) AND B(1));  end architecture; | ***Mux4xx1***  entity mux4XX1 is  port(a,b,c,d,s1,s2:in bit ; o : out bit);  end mux4XX1;  architecture mux4x1 of mux4XX1 is  begin  o<=(not s1 and not s2 and a)or  (not s1 and s2 and b)or  ( s1 and not s2 and c)or  ( s1 and s2 and d);  end mux4x1; |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |