

ADDERS IN PLDS

When using VHDL to synthesize an adder in a CPLD, some thought is necessary to obtain optimum results. VHDL, with no restrictions, will implement the adder with "Full Look-Ahead Carry." In other words, it will attempt to implement each output as a SOP function of the inputs. This can be very inefficient in terms of AREA and not especially efficient in terms of speed. An 8-bit adder synthesized this way uses the following capacity of a CYC375i CPLD.

FULL LOOK-AHEAD

Description	Used	Max
Dedicated Inputs	1	1
Clock/Inputs	4	4
I/O Macrocells	61	128
66 /		133 = 49 %

	Required	Max (Available)
CLOCK/LATCH ENABLE signals	0	4
Input REG/LATCH signals	0	5
Input PIN signals	5	5
Input PINs using I/O cells	13	13
Output PIN signals	48	115
Total PIN signals	66	133
Macrocells Used	48	128
Unique Product Terms	610	640
PROPAGATION DELAYS (PASSES)	S0, S1	1
	S2- S5	2
	S6, S7, Co	3

In accord with what was noted above, the propagation delays should have all been 1 pass. The problem is that no macrocell may have more than 16 product terms. Hence the fitter must split up functions with more than 16 Product terms. This explains the 2 and 3 pass outputs above.

One may use less area at the expense of a moderate increase in propagation delay by restricting the carries between states (use the "Synthesis_off" attribute). When C4 is restricted, two high speed 4-bit adders are connected by a ripple carry.

4-BIT LOOK AHEAD

Description	Used	Max
Dedicated Inputs	1	1
Clock/Inputs	4	4
I/O Macrocells	39	128
44 /		133 = 33 %

	Required	Max (Available)
CLOCK/LATCH ENABLE signals	0	4
Input REG/LATCH signals	0	5
Input PIN signals	5	5
Input PINs using I/O cells	13	13
Output PIN signals	26	115
Total PIN signals	44	133
Macrocells Used	26	128
Unique Product Terms	251	640
PROPAGATION DELAYS (PASSES)	S0, S1 1	
	S2, S3 2	
	S4, S5 3	
	S6, S7, Co 4	

One additional pass is needed in this case. One would expect that only 2 passes would be required. Again, the higher order bit functions have more than 16 Product Terms and hence the additional delay and area. One may restrict C2 and C6 as well as C4 to make 4 high speed 2-bit adders ripple carried together.

2-BIT LOOK AHEAD

Description	Used	Max
Dedicated Inputs	1	1
Clock/Inputs	4	4
I/O Macrocells	24	128
29 /		133 = 21 %

	Required	Max (Available)
CLOCK/LATCH ENABLE signals	0	4
Input REG/LATCH signals	0	5
Input PIN signals	5	5
Input PINs using I/O cells	12	12
Output PIN signals	12	116
Total PIN signals	29	133
Macrocells Used	12	128
Unique Product Terms	92	640
PROPAGATION DELAYS (PASSES)	S0, S1 1	
	S2, S3 2	
	S4, S5 3	
	S6, S7, Co 4	

One would expect a 4 pass delay with this type of adder and this is, indeed what is obtained. One may ask, finally, what is obtained is one uses a complete ripple carry adder?

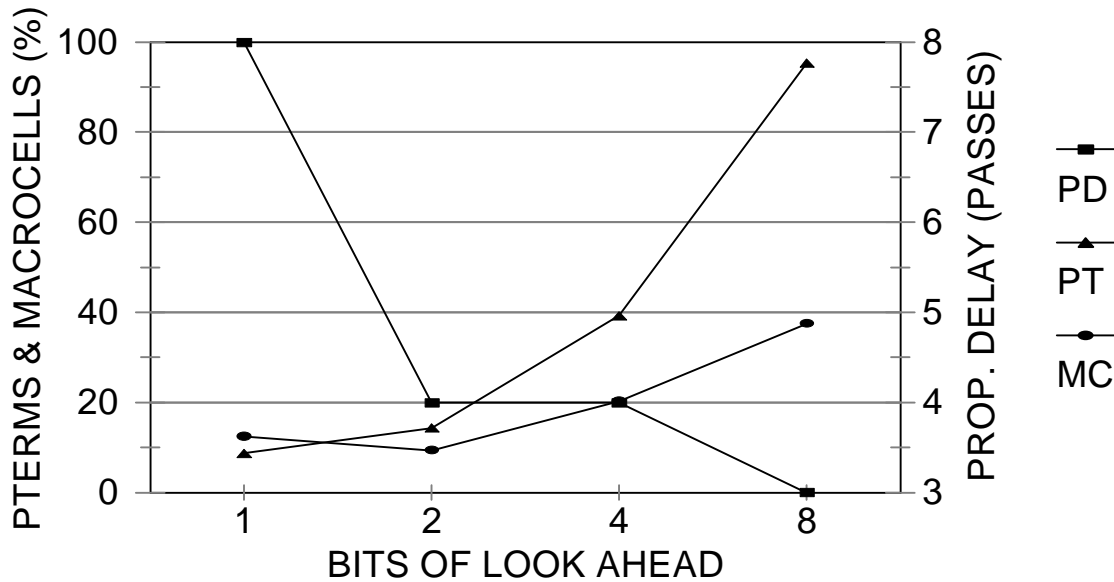
RIPPLE CARRY

Description	Used	Max
Dedicated Inputs	1	1
Clock/Inputs	4	4
I/O Macrocells	28	128
33 / 133		= 24 %

	Required	Max (Available)
CLOCK/LATCH ENABLE signals	0	4
Input REG/LATCH signals	0	5
Input PIN signals	5	5
Input PINs using I/O cells	12	12
Output PIN signals	16	116
Total PIN signals	33	133
Macrocells Used	16	128
Unique Product Terms	56	640
PROPAGATION DELAY (PASSES)	S0	1
	S1	2
	S2	3
	S3	4
	S4	5
	S5	6
	S6	7
	S7, Co	8

Again, this is what is to be expected. Finally, a plot may be made showing the device utilization (both in terms of Product Terms used and Macrocells used) and propagation delay.

ADDERS



Clearly, for this particular example, using 2-bit look ahead reduces the area significantly with only a 33% increase in propagation delay. A 4-bit look ahead is no better in speed and somewhat worse in area. A full look ahead uses 5 times the area of the other two. A full ripple carry adder, on the other hand, is quite slow.

If one wishes to spend the time to do it, a “real” look ahead carry adder can be implemented in VHDL as follows:

```
entity adder is port(
  a,b: in bit_vector(7 downto 0);
  ci : in bit;
  s  : out bit_vector(7 downto 0);
  co : out bit);
end adder;
architecture test_adder of adder is
  signal c,p,g: bit_vector(7 downto 0);
  attribute synthesis_off of c,p,g:signal is true;
begin
  process(a,b,ci,c) begin
    for i in 7 downto 0 loop
      g(i)<=a(i) and b(i);
      p(i)<=a(i) or b(i);
    end loop;
    c(1)<=g(0) or (p(0) and ci);
    c(2)<=g(1) or (p(1) and (g(0) or (p(0) and ci)));
    c(3)<=g(2) or (p(2) and (g(1) or (p(1) and (g(0) or (p(0) and ci)))));
    c(4)<=g(3) or (p(3) and (g(2) or (p(2) and (g(1) or (p(1) and
      (g(0) or (p(0) and ci)))))));
    c(5)<=g(4) or (p(4) and (g(3) or (p(3) and (g(2) or (p(2) and
      (g(1) or (p(1) and (g(0) or (p(0) and ci)))))))));
    c(6)<=g(5) or (p(5) and (g(4) or (p(4) and (g(3) or (p(3) and
      (g(2) or (p(2) and (g(1) or (p(1) and (g(0) or (p(0) and
      ci)))))))))));
    c(7)<=g(6) or (p(6) and (g(5) or (p(5) and (g(4) or (p(4) and
      (g(3) or (p(3) and (g(2) or (p(2) and (g(1) or (p(1) and
      (g(0) or (p(0) and ci)))))))))));
    co<=g(7) or (p(7) and (g(6) or (p(6) and (g(5) or (p(5) and
      (g(4) or (p(4) and (g(3) or (p(3) and (g(2) or (p(2) and
      (g(1) or (p(1) and (g(0) or (p(0) and ci)))))))))));
    for i in 7 downto 0 loop
      s(i)<=a(i) xor b(i) xor c(i);
    end loop;
  end process;
end;
```

The resources for this adder are:

Descripti on	Used	Max
Dedi cated Inputs	1	1
Clock/Inputs	4	4
I/O Macrocells	45	128
50 / 133 = 37 %		

	Required	Max (Available)
CLOCK/LATCH ENABLE signals	0	4
Input REG/LATCH signals	0	5
Input PIN signals	5	5
Input PINs using I/O cells	13	13
Output PIN signals	32	115
Total PIN signals	50	133
Macrocells Used	32	128
Unique Product Terms	92	640
PROPAGATION DELAYS (PASSES)	S0	1
	Co	2
	all others	3

Thus the delay is as fast as can be obtained in this device and uses less hardware than the 4-bit look-ahead but more than the 2-bit. It uses a whole lot less hardware than the “look ahead adder” created directly by VHDL.

Even the above is not the best design. The fact that it is not illustrates the need for the logic designer to be really aware of the hardware limitations of the device being used. In the above VHDL code, the c, g, and p arrays are listed in the “Synthesis_off” statement, forcing them to become nodes. But the g array is made up of single product terms (a and b) for which a special node is wasteful in a CPLD. If the g array is deleted from the synthesis_off term, the following usage report is obtained (the timing is the same).

Descripti on	Used	Max
Dedi cated Inputs	1	1
Clock/Inputs	4	4
I/O Macrocells	37	128
	42 /	133 = 31 %

	Required	Max (Available)
CLOCK/LATCH ENABLE signals	0	4
Input REG/LATCH signals	0	5
Input PIN signals	5	5
Input PINs using I/O cells	13	13
Output PIN signals	24	115
Total PIN signals	42	133
Macrocells Used	24	128
Unique Product Terms	84	640

This is a reduction from 32 to 24 macrocells!