UNIT 3: CARRY LOOKAHEAD ADDERS

Question and answers

1. What is the delay for the following types of 64-bit adders? Assume that each two-input gate delay is 150 ps and that a full adder delay is 450 ps.
   (a) a ripple-carry adder
   (b) a carry-lookahead adder with 4-bit blocks
   (c) a prefix adder

   Answer:

(a) From Equation $t_{\text{ripple}} = Nt_{FA}$

we find the 64-bit ripple-carry adder delay to be:

$$t_{\text{ripple}} = Nt_{\text{FA}} = 64(450\text{ ps}) = 28.8\text{ ns}$$

(b) From Equation

$$t_{CLA} = t_{pg} + t_{pg\_\text{block}} + \left(\frac{N}{k} - 1\right)t_{\text{AND\_OR}} + kt_{FA}$$

we find the 64-bit carry-lookahead adder delay to be:

$$t_{CLA} = t_{\text{pg}} + t_{\text{pg\_block}} + \left(\frac{N}{k} - 1\right)t_{\text{AND\_OR}} + kt_{\text{FA}}$$

$$t_{CLA} = \left[150 + (6 \times 150) + \left(\frac{64}{4} - 1\right)300 + (4 \times 450)\right] = 7.35\text{ ns}$$

(Note: the actual delay is only 7.2 ns because the first AND_OR gate only

has a 150 ps delay.)

(c) From Equation 5.11, we find the 64-bit prefix adder delay to be:

$$t_{PA} = t_{pg} + \log_2 N(t_{pg\_prefix}) + t_{XOR}$$

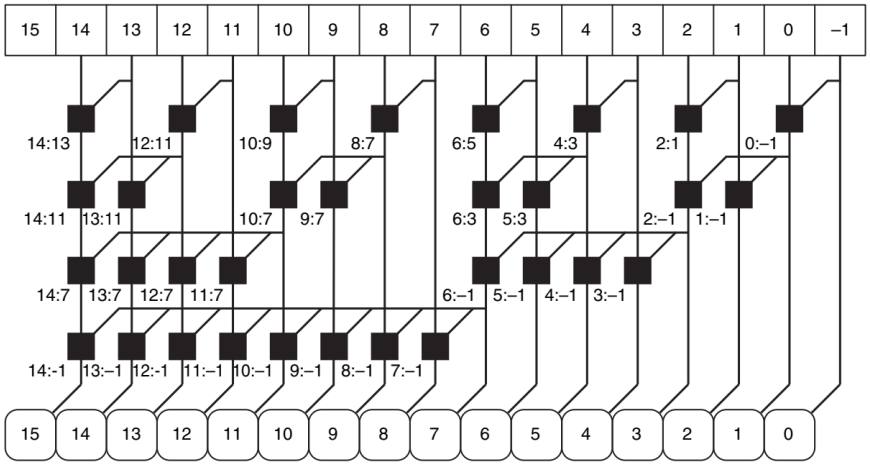
$$t_{PA} = t_{pg} + \log_2 N(t_{pg\_prefix}) + t_{XOR}$$

$$t_{PA} = [150 + 6(300) + 150] = 2.1 \text{ ns}$$

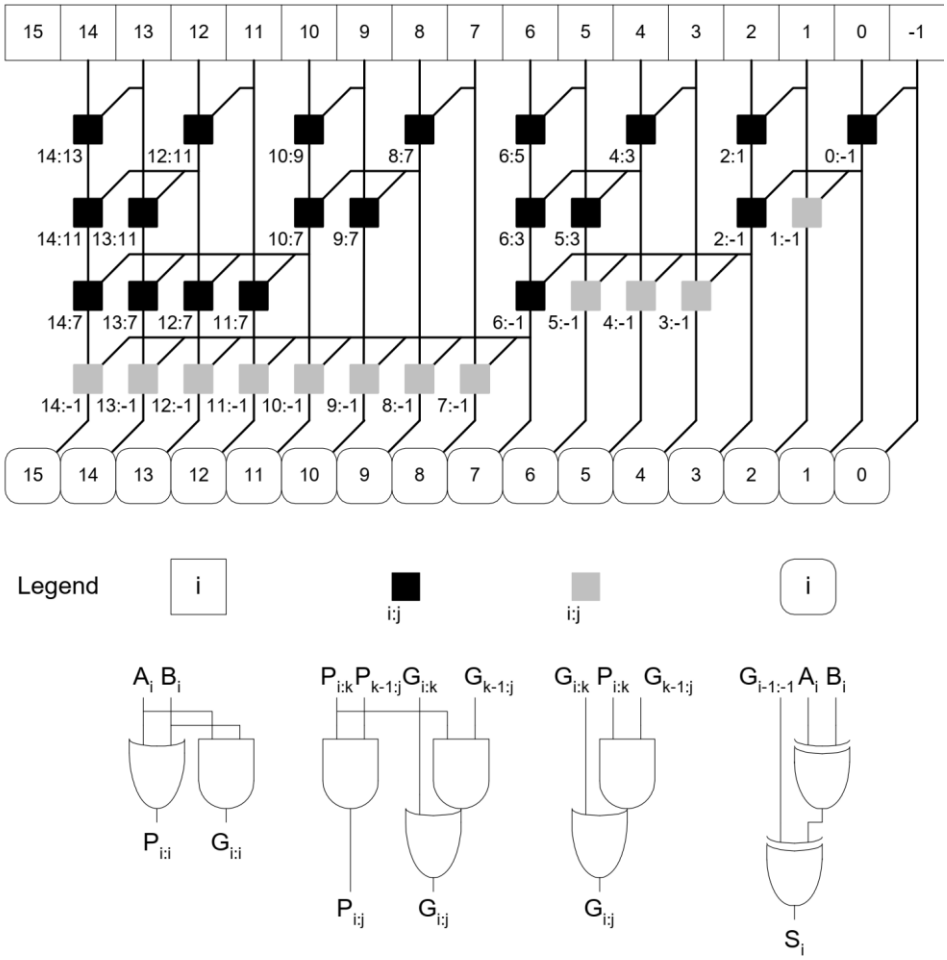
2.

The prefix network shown in the following Figure uses black cells to compute all of the prefixes. Some of the block propagate signals are not actually necessary.

Design a "gray cell" that receives G and P signals for bits i:k and k-1:j but produces only Gi:j, not Pi:j. Redraw the prefix network, replacing black cells with gray cells wherever possible



ANSWER:

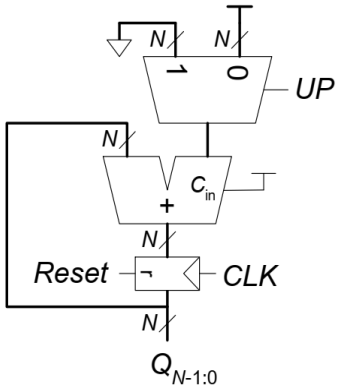

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 -1
14:13 12:11 10:9 8:7 6:5 4:3 2:1 0:-1
14:11 13:11 10:7 9:7 6:3 5:3 2:-1 1:-1
14:7 13:7 12:7 11:7 6:-1 5:-1 4:-1 3:-1
14:-1 13:-1 12:-1 11:-1 10:-1 9:-1 8:-1 7:-1
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Legend
i
i:j
i:j
i
A_i B_i
P_i:i G_i:i
P_i:k P_k-1:j G_i:k G_k-1:j
P_i:j G_i:j
G_i:k P_i:k G_k-1:j
G_i:j
G_i-1:-1 A_i B_i
S_i

UNIT 3: COUNTERS

Question and answers

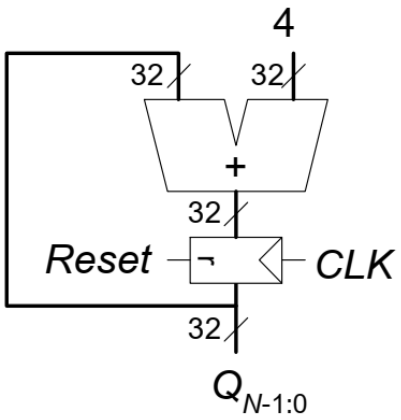
1. Build a 32-bit synchronous Up/Down counter. The inputs are Reset and Up. When Reset is 1, the outputs are all 0. Otherwise, when Up = 1, the circuit counts up, and when Up = 0, the circuit counts down.

   ANSWER:



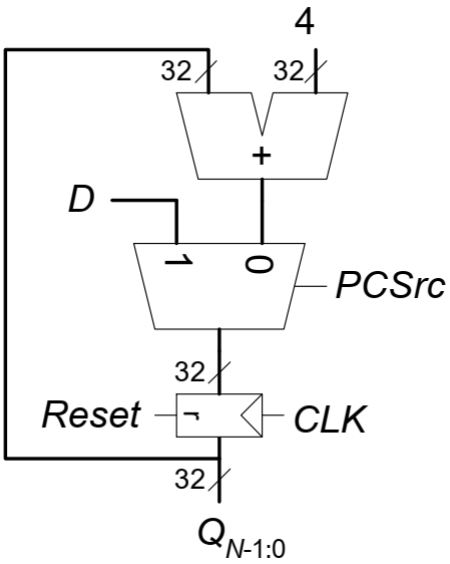
2. Design a 32-bit counter that adds 4 at each clock edge. The counter has reset and clock inputs. Upon reset, the counter output is all 0.

   Answer: 32-bit counter that increments by 4 on each clock edge



3. Modify the counter from Exercise 2 such that the counter will either increment by 4 or load a new 32-bit value, D, on each clock edge, depending on a control signal Load. When Load = 1, the counter loads the new value D.

   Answer:



UNIT 3: FINITE STATE MACHINES

Question and answers

1. *You are designing an FSM to keep track of the mood of four students working in the digital design lab. Each student's mood is either HAPPY (the circuit works), SAD (the circuit blew up), BUSY (working on the circuit), CLUELESS (confused about the circuit), or ASLEEP (face down on the circuit board). How many states does the FSM have? What is the minimum number of bits necessary to represent these states?*

   Answer:

   The FSM has 54 = 625 states. This requires at least 10 bits to represent all the states.

2. *How would you factor the FSM from question 1 into multiple simpler machines? How many states does each simpler machine have? What is the minimum total number of bits necessary in this factored design?*

   Answer:

   The FSM could be factored into four independent state machines, one for each student. Each of these machines has five states and requires 3 bits, so at
   least 12 bits of state are required for the factored design.

3. *Describe in words what the state machine in the following figure does. Using*
   binary state encodings, complete a state transition table and output table for the
   FSM. Write Boolean equations for the next state and output and sketch a schematic of the FSM.

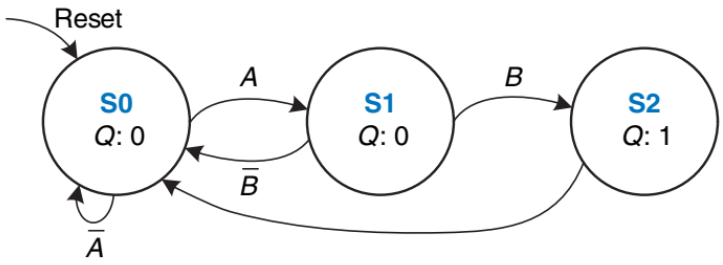


Figure: State transition diagram

Answer:

This finite state machine asserts the output Q for one clock cycle if A is TRUE followed by B being TRUE.

state	encoding $s_{1:0}$
S0	00
S1	01
S2	10

Table: State Encoding

current state		inputs		next state	
s_1	s_0	a	b	s'_1	s'_0
0	0	0	X	0	0
0	0	1	X	0	1

TABLE State transition table with binary encodings

current state		inputs		next state	
s_1	s_0	a	b	s'_1	s'_0
0	1	X	0	0	0
0	1	X	1	1	0
1	0	X	X	0	0

TABLE State transition table with binary encodings

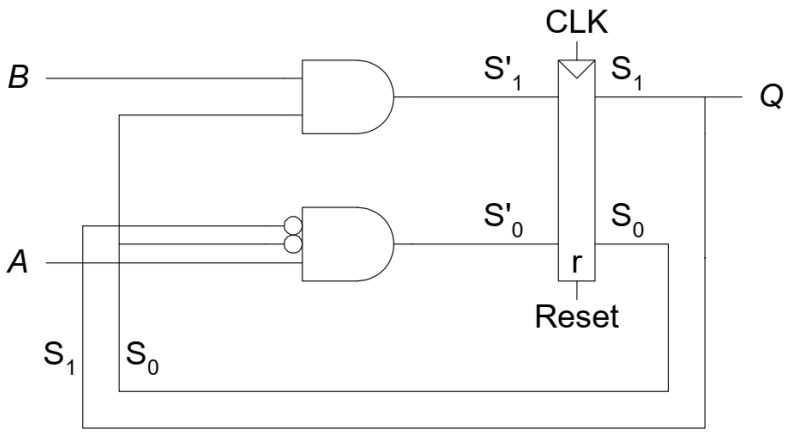
current state		output
s_1	s_0	q
0	0	0
0	1	0
1	0	1

TABLE Output table with binary encoding

$$S'_1 = S_0 B$$

$$S'_0 = \overline{S_1}\,\overline{S_0} A$$

$$Q = S_1$$


CLK
B
S'1
S1
Q
S'0
S0
A
r
Reset
S1
S0