E2.11/ISE2.22 – Digital Electronics II

Problem Sheet 6

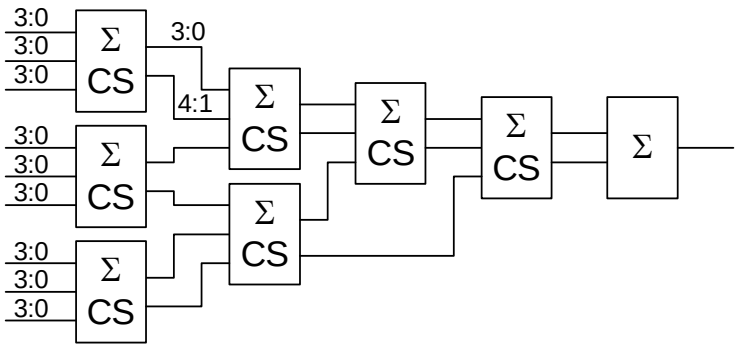
(Question ratings: A=Easy, …, E=Hard. All students should do questions rated A, B or C as a minimum)

1B+ A full-adder is a *symmetric* function of its inputs and is *self-dual*. Define the meaning of the italicised terms. Explain why self-duality implies that any NAND gate implementation of a full-adder can be converted into a NOR gate implementation with exactly the same gate interconnections.

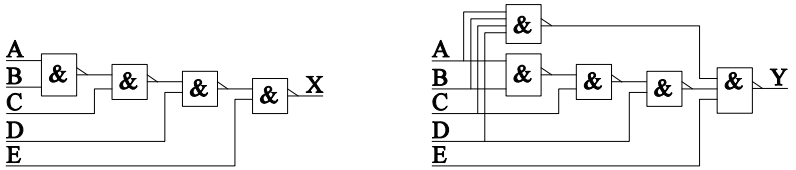
2B A number x lies in the range 0 to 9. Using only a full-adder, create a circuit that generates $y = 3x$ by adding together x and $2x$. Calculate the number of bits needed to represent y and hence use the smallest full-adder possible. How can the number of adder bits be reduced by using a single OR gate as well as the adder ?

3A. If CG = P•Q, CGP = P+Q and CP = P⊕Q, show using Boolean algebra that CG+CGP•CI = CG+CP•CI.

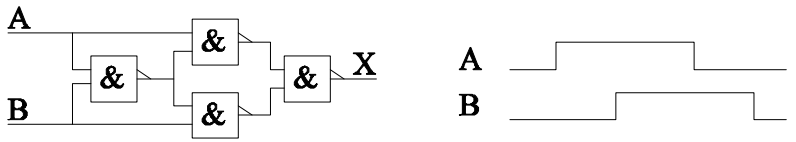
4C. The diagrams below show a possible interconnection for a carry-save addition tree in which each of the nine inputs is a 4-bit unsigned number. Label each of the intermediate and output numbers in the circuit with its range of bit numbers; thus 7:3 would indicate a 5-bit number whose least significant bit has a weight of 2^3. The outputs from the first carry-save block have already been labelled as an illustration.



5C. Show that the two circuits shown below generate the same Boolean function of their inputs. For each circuit, the inputs take the following sequence of values: ABCDE = 00000, 00001, 00011, 00111, 01111, 11111, 01111. For each circuit calculate the sequence of output values and the propagation delay each time the output changes. Each gate has a delay of 1. How is this circuit similar to the carry skip scheme?

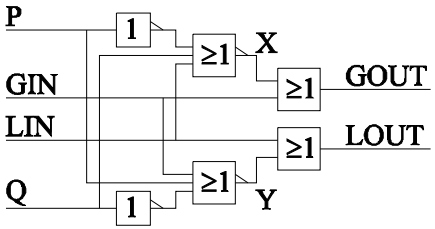


6C. The NAND gates in the following circuit have propagation delays of 5.5 ns when the output goes from low to high and 4 ns when it goes from high to low. If A and B have the waveforms shown, draw the waveform of X and give the propagation delay of the circuit for each output transition.

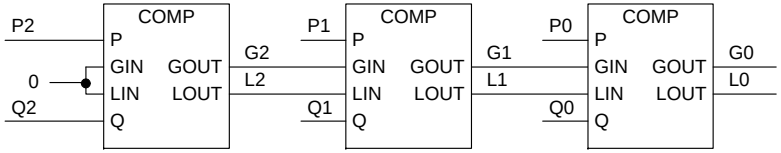


7B. Show how the circuit of a full-adder can be simplified if it is known that one of the input signals is always zero. Your circuit should use only NAND gates.

8C. The diagram shows the circuit of a single stage from a magnitude comparator. Calculate the worst-case propagation delays from each input to the GOUT output. In each case state what values the other inputs must have for the worst-case delay to occur.



9D. A 3-bit comparator for unsigned numbers is made by combining 3 copies of the circuit in the previous question:

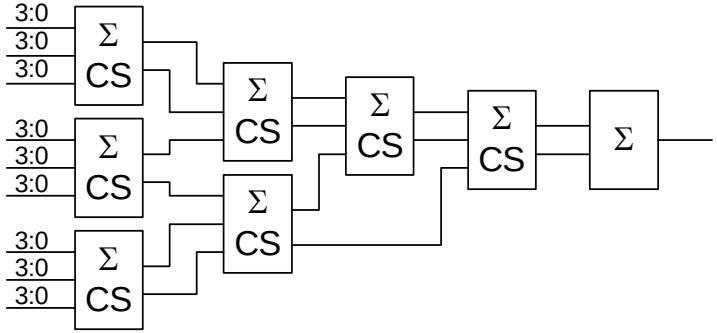


(a) Show that G0=1 if and only if P2:0 > Q2:0.

(b) Determine the worst-case delay from P2 to G0 and give an example of when it occurs (i.e. give the initial values of P2:0 and Q2:0 before P2 changes).

10D. The circuit shows a carry-save tree for adding together nine 4-bit unsigned numbers. Calculate the propagation delay of the circuit and give an example where this delay occurs. Each carry-save bit is implemented using the nine NAND gate circuit from the notes. The final stage is a carry-lookahead adder with a delay of 6.

Show how the delay may be reduced by inserting inverters between the columns of carry-save adder modules and merging the resultant AND gates into the following stages.



E2.11/ISE2.22 – Digital Electronics II

Solution Sheet 6

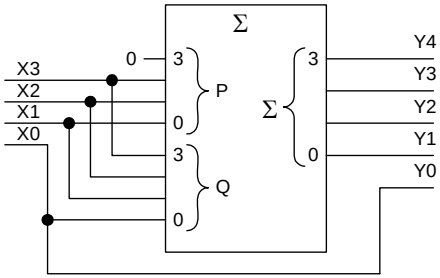
(Question ratings: A=Easy, …, E=Hard. All students should do questions rated A, B or C as a minimum)

1B+ *Symmetric* means that the inputs can be permuted arbitrarily without affecting any of the outputs.

Self-dual means that inverting all the inputs will cause all the outputs to be inverted.

It is possible to re-implement the circuit using *negative* logic in which a logic 1 is a low voltage and logic 0 is a high voltage. In negative logic a NAND function is performed by a positive logic NOR gate: thus wherever the original circuit had a NAND, you now need a NOR (and vice-versa). The input and output signals to the circuit still use positive logic, so they undergo logical inversion; the nett effect is therefore a full adder with inverters at the inputs and outputs. Since the adder is self-dual this is the same as a full adder.

2B *y* lies in the range 0 to 27 and therefore needs a 5 bit unsigned number. We generate 2*x* just by shifting the bits one space to the left; this doesn't need a shift register or any other circuitry – it just involves relabelling the bits. We don't need any adder stage for the LSB since X0 is always equal to Y0.

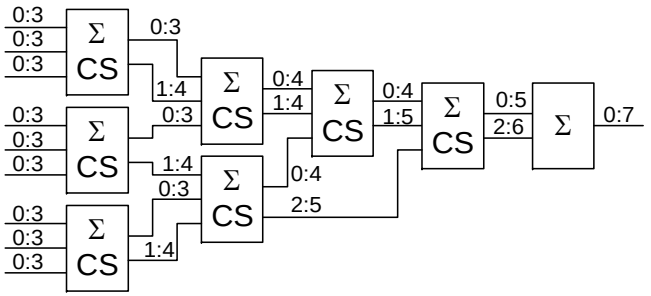


The Y4 output is the addition of Q3 and P3 together with the carry from the previous stage. Since P3=0 you can generate this by XORing Q3 with the carry. Since we know the result can never exceed 5 bits, we know that there can never be a carry out o bit 4 and hence that Q3 and the carry from stage 3 can never both be high at the same time. It follows that we can use an OR gate instead of the XOR that we would normally need. Hence we use a 3-bit full adder and OR X3 with the carry out of the adder to generate Y4.

3A. CGP = P+Q = P•(Q+!Q) + Q•(P+!P) = P•Q + P•!Q + !P•Q = CG + CP

CG + CGP•CI = CG + (CG+CP)•CI = CG•(1+CI) + CP•CI = CG + CP•CI

4C. The maximum output from the circuit is 9×15=135 which needs 8 bits.



The SUM output from a CS adder must go from the lowest bit position of any input to the highest bit position of any input. The CARRY output must go from one more than the lowest bit position with at least two inputs to one more than the highest bit position with at least two inputs. This is because we need at least two inputs to generate a carry.

Note that several of the carry-save modules have bit positions in which either one or two of the inputs are missing: in these cases the circuitry becomes either simpler or completely unnecessary. In the latter case, no carry output is possible at all.

We can actually save a small amount of circuitry by rearranging the connections between the first two stages so that all the 4:1 signals go to one carry-save module and all the 3:0 signals go to the other. This increases the number of columns that need no circuitry at all.

5C.

$$X = \overline{E} + D \cdot \left(\overline{C} + A \cdot B\right) = \overline{E} + \overline{C} \cdot D + A \cdot B \cdot D$$

$$Y = \overline{E} + \overline{C} \cdot D + A \cdot B \cdot D + A \cdot B \cdot C \cdot D = X$$

ABCDE	X=Y	X delay	Y delay	
00000	1			
00001	0	1	1	
00011	1	2	2	
00111	0	3	3	
01111	0			
11111	1	4	2	← Speedup
01111	0	4	4	← No Speedup

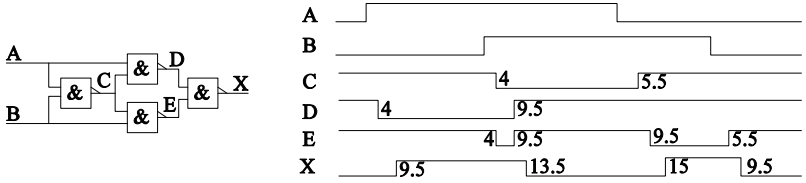
The circuit is similar to carry skip in that the situation causing the worst-case delay is recognised by the 4-input NAND gate and the chain of three NAND gates is then bypassed causing a speedup.

The circuit differs from carry skip in two respects:

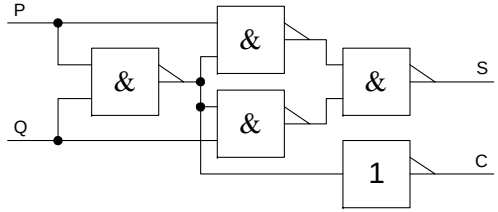
(a) Carry skip uses a multiplexer to switch between the fast and slow paths which means that it works for both rising and falling edges of its output. The use of a NAND gate in this circuit to combine the fast and slow paths means that the speedup is only effective for rising output transitions.

(b) Carry skip improves the worst-case delay at the expense of making the delay greater under most other conditions. This circuit doesn't make any delays worse.

6C. In the timing diagram, I have marked each edge with the time delay from the A or B transition that caused it. The circuit is an XOR function.



7B. Because a full-adder is symmetrical, it doesn't matter which of the inputs we take to be zero. Here I have assumed that it is the CI input.



8C. P→GOUT = 3 when !Q•!LIN•!GIN =1
GIN→GOUT = 1 when !P+Q+LIN=1
LIN→GOUT = 2 when P•!Q•!GIN=1
Q→GOUT = 2 when P•!LIN•!GIN =1

9D. The circuit compares the two numbers beginning at the most significant bit. If Px = Qx for all the bits down to and including bit n, then Gn=Ln=0. Otherwise, either Gn or Ln is high according to whether P is greater or less than Q. Pn and Qn are never high together.

Looking at the circuit of the previous question, we can see that GOUT is high either if GIN=1 (i.e. if higher order bits have determined that P>Q) or else if P=1 and Q=0 and LIN=0 (i.e. previous bits are identical but P>Q because of this bit).

The longest delay is P2→G0 = 7 when Q=2 and P changes from 5 to 1.

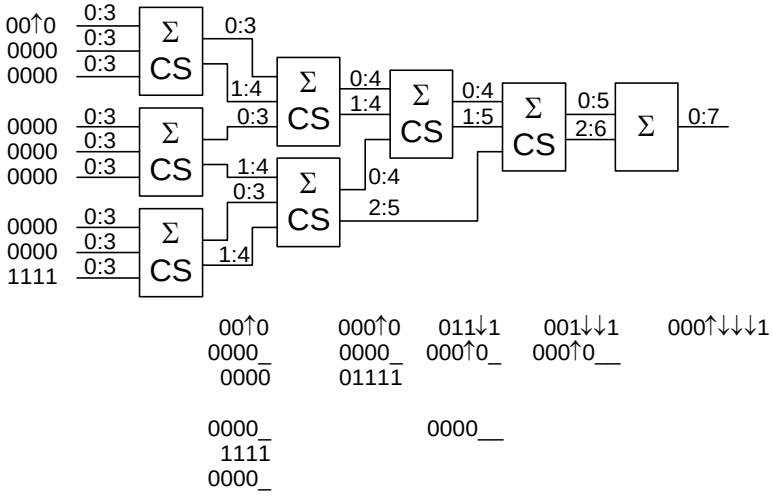
Time 0: P2↓
Time 1: !P2↑
Time 2: X2↓
Time 3: G2↓
Time 4: Y1↑
Time 5: L1↑
Time 6: X0↓
Time 7: G0↓

10D. Initial delay = 4×3 + 6 = 18.

The delay to the S output of the CS adder is 3 in any bit position where neither the initial or final states have all 3 inputs high. The delay of the final adder is 6 if any of its carry signals change.

Carry-save bits with only one active input (such as the least significant bit of the last carry-save module) require no circuitry and hence have no delay. To avoid these bits, we alter bit 1 rather than bit 0 in the example below.

The numbers written below the diagram show the value of each bit of each number. As indicated by the up-arrow, we are changing bit 1 of the uppermost input number from 0 to 1 (i.e. increasing the number's value by 2). The resultant changes in other values are indicated by the up or down arrows in the diagram.



By inserting inverters between stages, we can change the delay to 4×2+6 = 14. We should not insert an inverter in the signal path that skips a stage.