EECS 151/251A Homework 10

Due Monday, April 27th, 2021

For this Homework

Please include a short (1-2 sentence) explanation with your answer, unless otherwise noted.

Problem 1: Binary Incrementer

A straightforward way to implement a binary incrementer ($x + 1$) is to use an adder and a register. While this does achieve the result we desire, an adder is a very expensive circuit to implement for such a simple operation. Design an incrementer circuit that performs an N-bit binary increment combinationally.

Solution:

Let's consider a 4-bit counter for now.

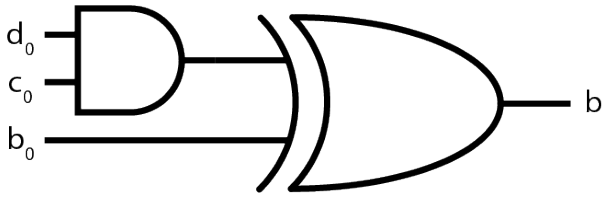
$\mathbf{a_0}$	$\mathbf{b_0}$	$\mathbf{c_0}$	$\mathbf{d_0}$	**a**	**b**	**c**	**d**
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	1	0	1	0
1	0	1	0	1	0	1	1
1	0	1	1	1	1	0	0
1	1	0	0	1	1	0	1
1	1	0	1	1	1	1	0
1	1	1	0	1	1	1	1
1	1	1	1	0	0	0	0

From inspecting this truth table, we can see some patterns emerging. For the two least-significant bits, there are special patterns. For the LSB, the relationship from one count to the next is simply the inversion of the current bit, $d = \overline{d_0}$. This makes sense as a 1-bit increment-by-1 would just manifest as an inverter. The second least significant bit has a slightly more complex pattern, where $c = c_0 \oplus d_0$.

From the third least significant bit and on, the pattern becomes more regular.

$$b = \begin{cases} c_0 \cdot d_0 & b = 0 \\ (c_0 \cdot d_0)' & b = 1 \end{cases}$$

Such an expression can be implemented with an AND gate followed by an XOR gate.



The remaining bit of the output follows the same pattern, only now it depends on a 3-input AND of the 3 less-significant bits. Further bits will follow this same pattern.

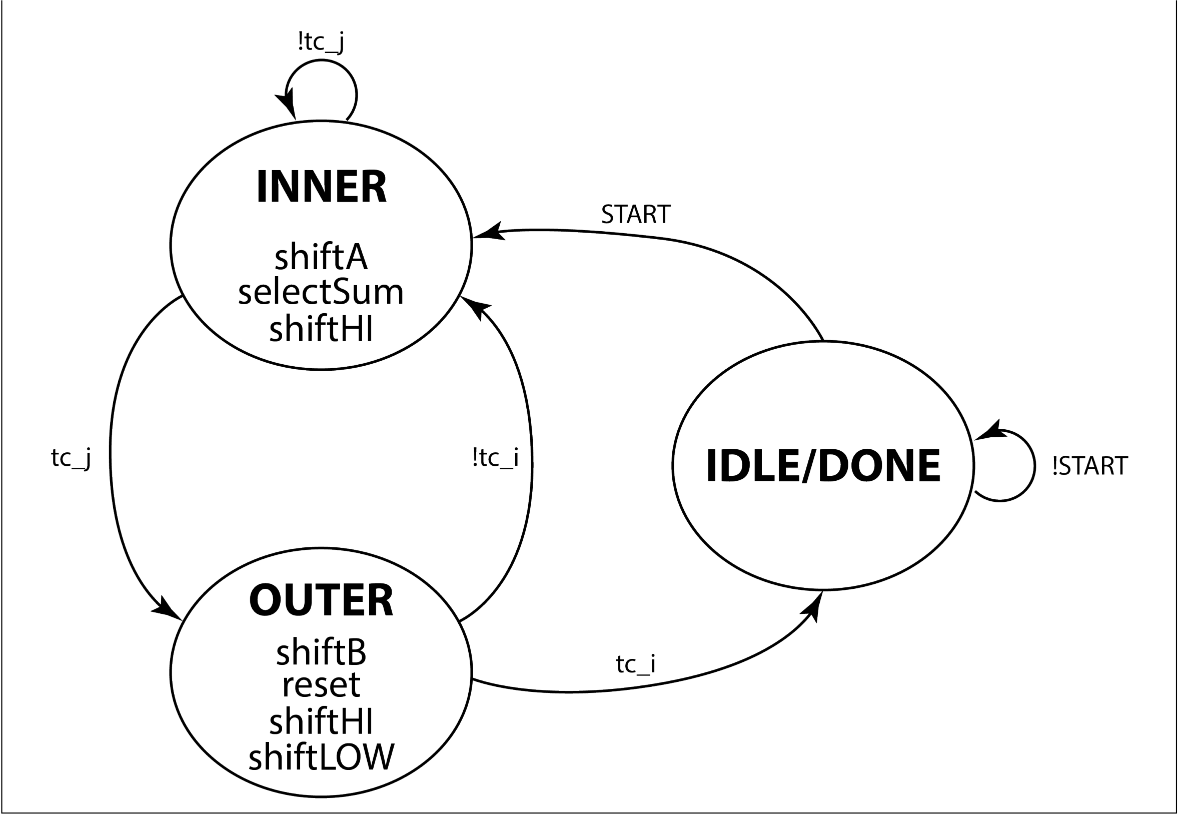
Problem 2: Counters

Using binary counters with a count enable (`ce`) input and terminal count (`tc`) output, design an FSM controller for a 32-bit serial multiplier.

For this problem, turn in

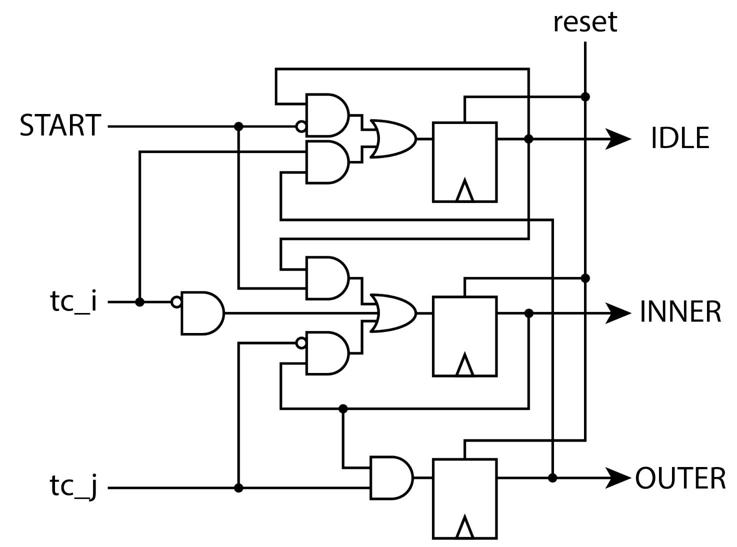
- A state transition diagram

Solution:



- A circuit diagram implementing the next state logic and multiplier. You may represent the counters as blocks.

Solution:

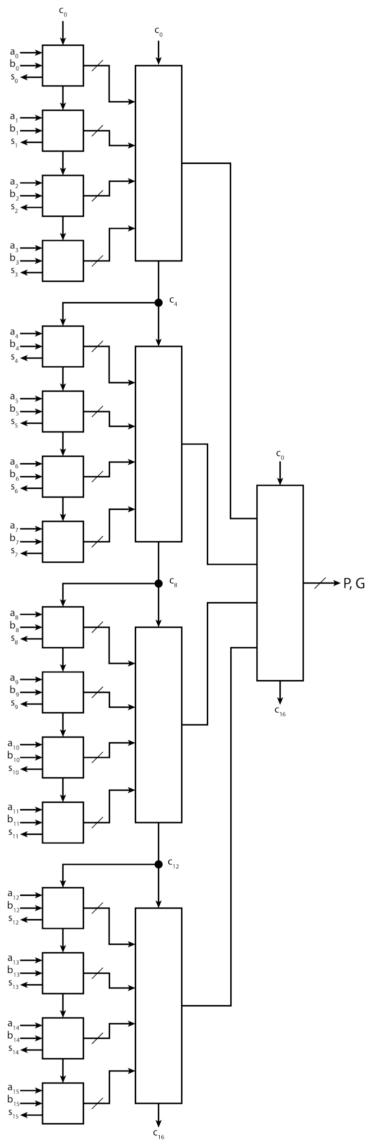


- A short explanation of your design

Problem 3: Adders

Design a 16-bit Carry Look-Ahead adder using 4 4-bit ripple stages. Show a detailed circuit diagram.

Solution:



Problem 4: More Adders

Write out the expression for the MSB of the sum of an 8-bit Brent-Kung adder, in terms of the sub-expressions of the other nodes in the adder.

Solution:

$$G_{7,0} = g_7 + g_6 p_7$$

$$G_{5,0} = g_5 + g_4 p_5$$

$$P_{7,0} = p_7 p_6$$

$$G_{3,0} = g_3 + g_2 p_3$$

$$G_{1,0} = g_1 + g_0 p_1$$

$$P_{3,0} = p_3 p_2$$

$$P_{1,0} = p_1 p_0$$

$$P_{3,1} = P_{3,0} P_{1,0}$$

$$P_{7,1} = P_{7,0} P_{3,1}$$

$$G_{7,1} = G_{7,0} + G_{5,0} P_{7,0}$$

$$G_{3,1} = G_{3,0} + G_{1,0} P3, 0$$

$$c_8 = G_{7,1} + G_{3,1} P_{7,1}$$

Problem 5: Multipliers

For the following multiplier/muliplicand pairs, show the long-hand calculation for mulitplication using the method(s) indicated. All multiplications are 4-bit.

(a) Traditional long multiplication

- 5×3

Solution:

$$\begin{array}{cccccccc}
 & & & & 0 & 1 & 0 & 1 \\
 & & \times & & 0 & 0 & 1 & 1 \\
\hline
 & & & & 0 & 1 & 0 & 1 \\
 & & & 0 & 1 & 0 & 1 & \\
 & & 0 & 0 & 0 & 0 & & \\
+ & 0 & 0 & 0 & 0 & & & \\
\hline
 & 0 & 0 & 0 & 1 & 1 & 1 & 1
\end{array}$$

- -3×6

Solution:

$$\begin{array}{ccccccccc}
 & & & & & 1 & 1 & 0 & 1 \\
 & & & & \times & 0 & 1 & 1 & 0 \\ \cline{5-9}
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & \\
+ & 1 & 1 & 1 & 1 & 0 & 1 & & \\
- & 0 & 0 & 0 & 0 & 0 & & & \\ \hline
 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0
\end{array}$$

- 5×-4

Solution:

$$\begin{array}{ccccccccc}
 & & & & & 0 & 1 & 0 & 1 \\
 & & & & \times & 1 & 1 & 0 & 0 \\ \cline{4-8}
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\
+ & 0 & 0 & 0 & 1 & 0 & 1 & & \\
- & 0 & 0 & 1 & 1 & 1 & & & \\ \hline
 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0
\end{array}$$

- -4×-2

Solution:

$$\begin{array}{ccccccccc}
 & & & & & 1 & 1 & 0 & 0 \\
 & & & & \times & 1 & 1 & 1 & 0 \\ \cline{5-9}
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & \\
+ & 1 & 1 & 1 & 1 & 0 & 0 & & \\
- & 1 & 1 & 1 & 0 & 0 & & & \\ \hline
 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0
\end{array}$$

(b) Booth Recoding

- 7×5

Solution:

Let's begin by reiterating the Booth Recoding table:

B_{k+1}	B_k	B_{k-1}	Action
0	0	0	Add 0
0	0	1	Add A
0	1	0	Add A
0	1	1	Add 2A
1	0	0	Sub 2A
1	0	1	Sub A
1	1	0	Sub A
1	1	1	Add 0

Table 1: Booth Recoding Actions

Apply to the calculation,

$$
\begin{array}{llccccccc}
 & & & & & 0 & 1 & 1 & 1 \\
 & & & & \times & 0 & 1 & 0 & 1 \\
\cline{5-9}
\text{ADD A} & + & & & & 0 & 1 & 1 & 1 \\
\text{ADD A} & + & & 0 & 1 & 1 & 1 & & \\
\hline
 & & 0 & 1 & 0 & 0 & 0 & 1 & 1
\end{array}
$$

(c) Baugh-Wooley Method

- 5×3

Solution:

No difference in positive-positive Baugh-Wooley

$$
\begin{array}{cccccccc}
 & & & & 0 & 1 & 0 & 1 \\
 & & \times & & 0 & 0 & 1 & 1 \\
\cline{4-8}
 & & & & 0 & 1 & 0 & 1 \\
 & & & 0 & 1 & 0 & 1 & \\
 & & 0 & 0 & 0 & 0 & & \\
+ & 0 & 0 & 0 & 0 & & & \\
\hline
 & 0 & 0 & 0 & 1 & 1 & 1 & 1
\end{array}
$$

- -3×6

Solution:

First, sign extend like in long division,

$$
\begin{array}{ccccccccc}
 & & & & & 1 & 1 & 0 & 1 \\
 & & & & \times & 0 & 1 & 1 & 0 \\
\cline{5-9}
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & \\
+ & 1 & 1 & 1 & 1 & 0 & 1 & & \\
- & 0 & 0 & 0 & 0 & 0 & & & \\
\hline
\end{array}
$$

Add 1 to every sign bit to cancel sign extension,

$$\begin{array}{ccccccccc}
 & & & & & 1 & 1 & 0 & 1 \\
 & & & & \times & 0 & 1 & 1 & 0 \\
\cline{5-9}
 & & & & & 1 & 0 & 0 & 0 \\
 & & & & 0 & 1 & 0 & 1 & \\
+ & & & 0 & 1 & 0 & 1 & & \\
- & & 1 & 0 & 0 & 0 & & & \\
\hline
\end{array}$$

Subtract additional constants at the end (note in this case since the last partial product is 0, we don't need to do a 2's complement inversion when subtracting it)

$$\begin{array}{ccccccccc}
 & & & & & 1 & 1 & 0 & 1 \\
 & & & & \times & 0 & 1 & 1 & 0 \\
\cline{5-9}
 & & & & & 1 & 0 & 0 & 0 \\
 & & & & 0 & 1 & 0 & 1 & \\
+ & & & 0 & 1 & 0 & 1 & & \\
- & & 1 & 0 & 0 & 0 & & & \\
- & & 1 & 1 & 1 & 1 & & & \\
\hline
 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0
\end{array}$$

- 5×-4

Solution:

$$\begin{array}{ccccccccc}
 & & & & & 0 & 1 & 0 & 1 \\
 & & & & \times & 1 & 1 & 0 & 0 \\
\cline{5-9}
 & & & & & 1 & 0 & 0 & 0 \\
 & & & & 1 & 0 & 0 & 0 & \\
+ & & & 1 & 1 & 0 & 1 & & \\
- & & 1 & 1 & 0 & 1 & & & \\
- & & 1 & 1 & 1 & 1 & & & \\
\hline
 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0
\end{array}$$

- -4×-2

Solution:

$$\begin{array}{ccccccccc}
 & & & & & 1 & 1 & 0 & 0 \\
 & & & & \times & 1 & 1 & 1 & 0 \\
\cline{5-9}
 & & & & & 1 & 0 & 0 & 0 \\
 & & & & 0 & 1 & 0 & 0 & \\
+ & & & 0 & 1 & 0 & 0 & & \\
- & & 0 & 1 & 0 & 0 & & & \\
- & & 1 & 1 & 1 & 1 & & & \\
\hline
 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0
\end{array}$$

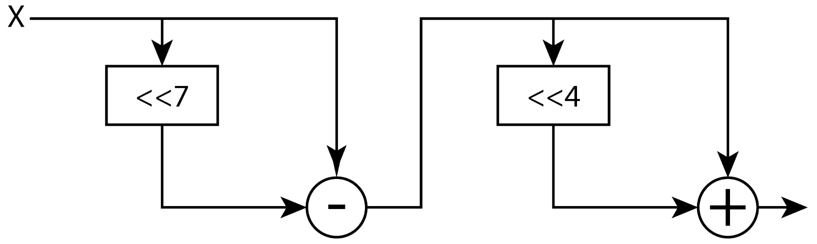
You may explain your steps for the Booth and Baugh-Wooley calculations in place of the short explanation.

Problem 6: Constant Value Multiplication

Design a circuit using full adder cells for an unsigned multiplication that computes $Y = C \cdot X$, where $C = 2159$ and X is an input. Use a minimum number of only full adders, logical shifters, and inverters. (*Hint: Consider the CSD and KCM methods detailed in the lecture*)

Solution:

The KCM solution is given below.

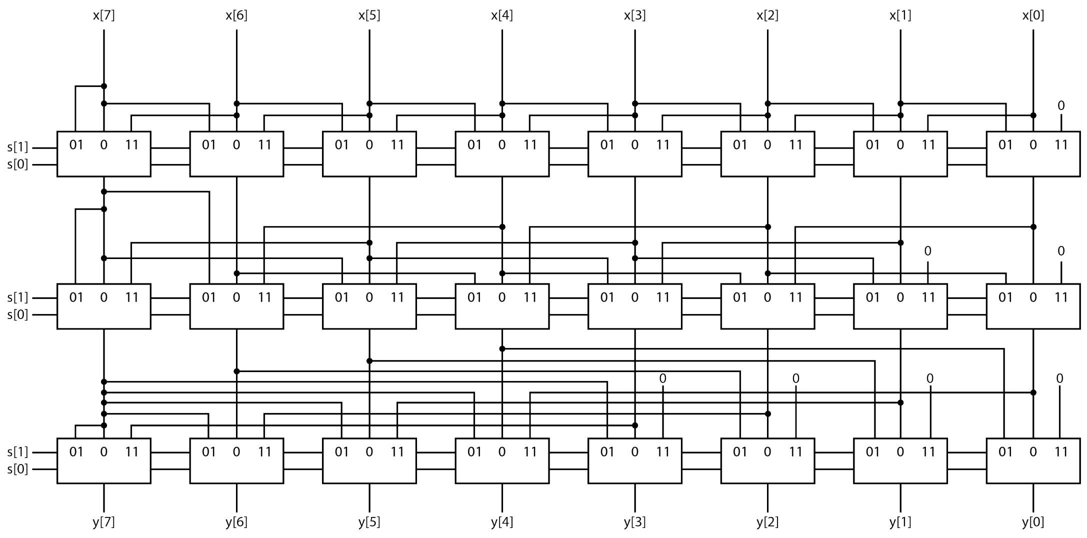


Problem 7: Shifters

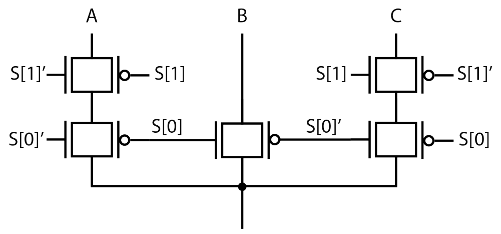
Design an 8-bit wide logarithmic shifter that performs an arithmetic shift left or right by 0-7 places. The circuit has a 4-bit input: 3 bits to specify the number of places to shift, and 1 bit to specify the direction. You are allowed to use multiplexers implemented using transmission gate logic to create the shifter. Provide a circuit diagram for the overall shifter, as well as a more detailed schematic of the multiplexers that you use. You may represent the multiplexers in your shifter diagram as symbols, but for each type of multiplexer you use, please show separately their implementation at the transistor level.

Solution:

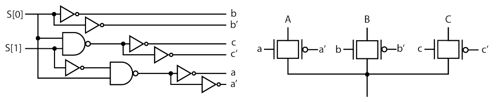
The log shifter can be implmented as shown,



The 3-input MUXes can be made in one of two ways. The first line of muxes should be made with minimal select logic to minimize the select delay. The second line of muxes can be made with more logic on the select lines to allow for minimal propagation delay through the transmission gates in the mux.



Mux circuit A



Mux circuit B