ECE 30
Introduction to Computer Engineering

Lab #7 Solutions
Spring 2014

1. **Computer Arithmetic**

   (a) What decimal number do these 32 bit 2's complement binary numbers represent? **Note**: You need not simplify your answer.

   - $1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110\ 1111_2$
     $1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110\ 1111_2 =$
     $-\ (0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001\ 0000_2 + 1) =$
     $-\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001\ 0001_2 = -17$

   - $0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110\ 1111_2$
     $0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110\ 1111_2 = 2^{31} - 1 - 16 = 2147483631$

   - $1111\ 1111\ 1111\ 1111\ 1111\ 1110\ 0000\ 1100_2$

     $1111\ 1111\ 1111\ 1111\ 1111\ 1110\ 0000\ 1100_2 =$
     $-\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001\ 1111\ 0011_2 + 1 =$
     $-\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001\ 1111\ 0100_2 = -500$

   - A is an 8-bit integer in the two's complement representation and B is a 4-bit two's complement integer. Which one is larger, if $A = 11011101_2$ and $B = 1101_2$? What is the 8-bit two's complement representation of B?
     Answer:

     $A = 11011101_2 = -(00100010_2 + 1) = -00100011_2 = -35$
     (also: $-2^7 + 2^6 + 2^4 + 2^3 + 2^2 + 2^0$)

     $B = 1101_2 = -(0010_2 + 1) = -0011_2 = -3$
     (also: $-2^3 + 2^2 + 2^0$)

     So, B is larger. In 8-bit two's complement, B would be:

     $B = -3 = -00000011_2 = 11111100_2 + 1 = 11111101_2$

     This could be obtained more directly by "sign extending" B, i.e., copying the MSB of B 4 times to the left.

2. **Logic Design** The *super-majority gate* is a 4-input logic gate whose output is 1 only if three or more of its four inputs are 1.

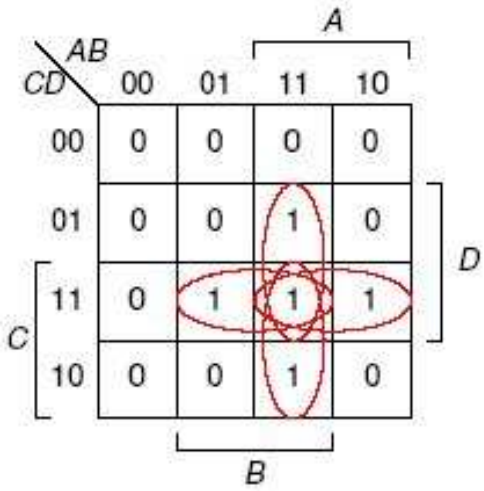
$$F = SM(A, B, C, D)$$

For example, if $A = B = C = 1$, then $F = 1$, but if $C = D = 0$, then $F = 0$.

(a) (2 points) Fill in the truth table below with all those input values for which the output is 1. (The table may have some extra rows. You can leave them blank).

A	B	C	D	F
1	1	1	0	1
0	1	1	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1
-	-	-	-	1
-	-	-	-	1

(b) Draw Karnaugh Map (K-map) for the above truth table.

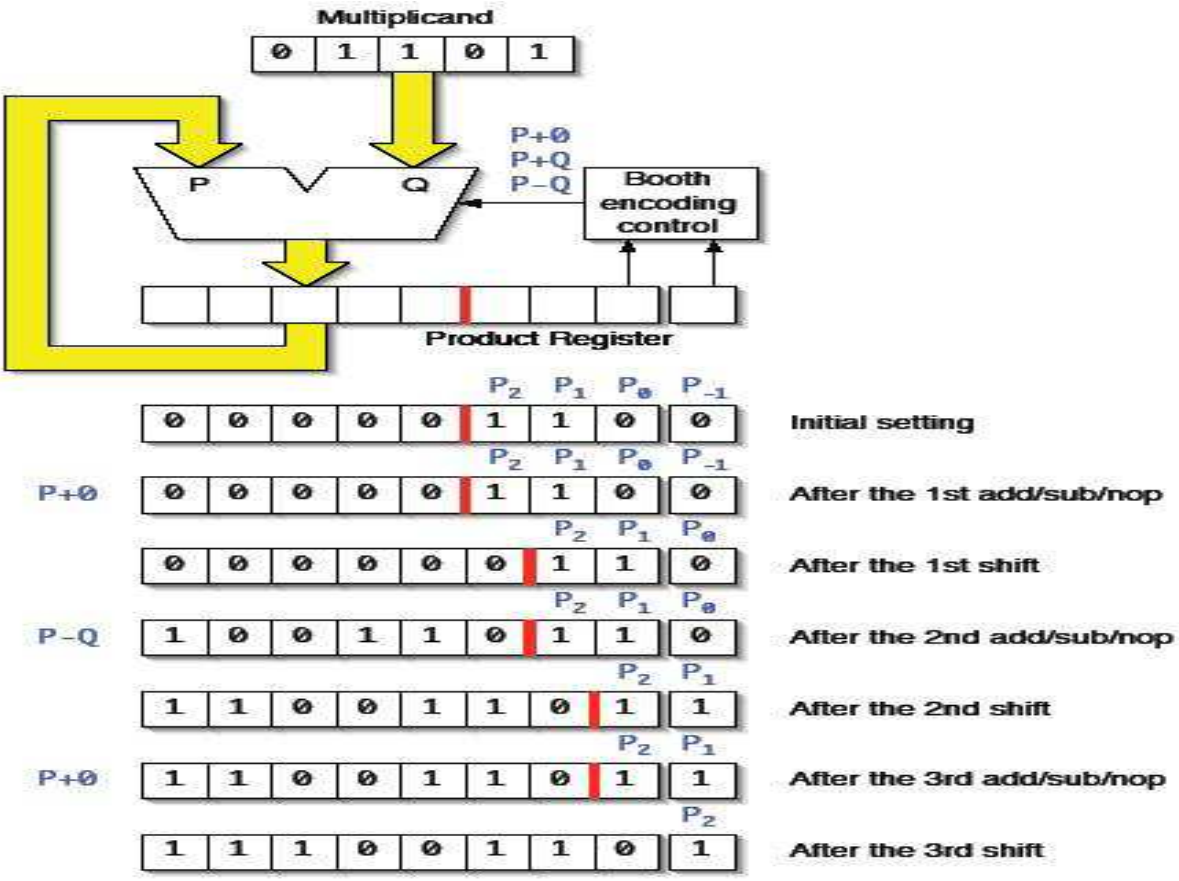


(c) Now, write F in terms of its MINIMAL sum-of-products form.
$F = ABC + ABD + ACD + BCD$

3. **Booth's Algorithm**

Trace the steps of multiplying a 5-bit two's complement **multiplicand** 01101_2 with a 3-bit two's complement **multiplier** 110_2, using Booth's algorithm. Assume that the shift-add multiplier shown below is used. Initially, the 5 most significant bits of the 8-bit product register contain 00000 and the 3 least significant bits contain the multiplier (110_2). When the algorithm completes, the product register must contain the 8-bit product in two's complement representation. Complete the table below.

Answer:

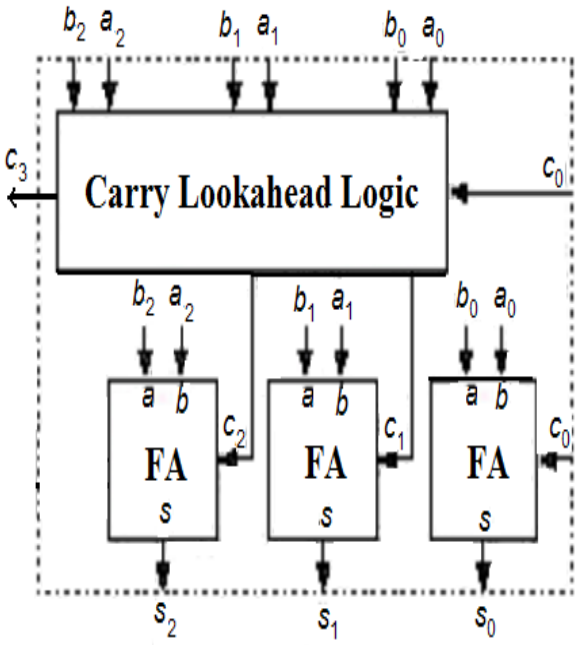

Multiplicand
0 1 1 0 1
P+0
P+Q
P−Q
P
Q
Booth encoding control
Product Register
P2 P1 P0 P−1
0 0 0 0 0 1 1 0 0
Initial setting
P+0
0 0 0 0 0 1 1 0 0
After the 1st add/sub/nop
0 0 0 0 0 0 1 1 0
After the 1st shift
P−Q
1 0 0 1 1 0 1 1 0
After the 2nd add/sub/nop
1 1 0 0 1 1 0 1 1
After the 2nd shift
P+0
1 1 0 0 1 1 0 1 1
After the 3rd add/sub/nop
1 1 1 0 0 1 1 0 1
After the 3rd shift

4. **Fast Adder Design**

In this problem, you are to design an 8-bit adder using 3-bit carry lookahead adders (CLA) and 1-bit full/half adders (FA/HA) as building blocks and analyze its performance.

(a) An incomplete block diagram of a 3-bit CLA is shown below. Complete the block diagram by adding the sum logic (use 1-bit full adder blocks when constructing the sum logic). Generate the logic equations for c_1, c_2 and c_3 using only g_0, g_1, g_2, p_0, p_1, p_2 and c_0. Note that $g_i = a_i b_i$ and $p_i = a_i + b_i$.

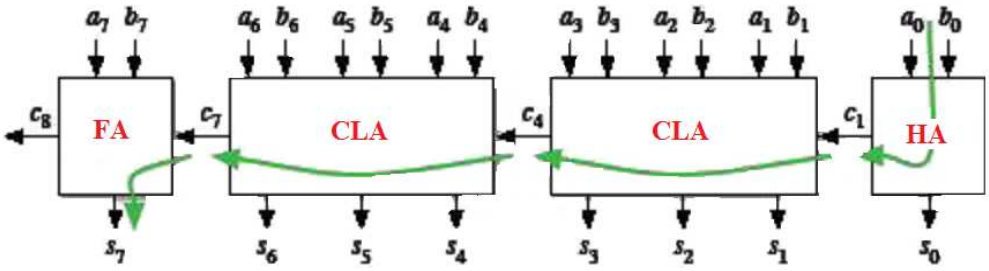
Answer:



- $c_1 = g_0 + p_0 c_0$
- $c_2 = g_1 + p_1 g_0 + p_1 p_0 c_0$
- $c_3 = g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 c_0$

(b) Construct an 8-bit adder using 3-bit CLA blocks as shown below and 1-bit full adders and half adders as necessary. Optimize your design to minimize the worst-case computation delay.

Answer:



(c) Compute the worst-case delay from changes in inputs to the last sum bit changing for the 8-bit adder in (b). Assume that

- all inputs change simultaneously;
- delay from inputs to $g_i/p_i = \mathsf{T}$;
- delay from $g_i/p_i/c_0$ to $c_i = 2\mathsf{T}$ in the carry lookahead logic;
- delay from a_i/b_i to $c_{i+1} = \mathsf{T}$ in the half adder;
- delay from a_i/b_i to $c_{i+1} = 2\mathsf{T}$ in the full adder;
- delay from c_i to $s_i = \mathsf{T}$.

Give an example of two operands which would cause the worst-case delay.

Answer:

- $1\mathsf{T}$ for $a_0/b_0 \rightarrow c_1$,
- $2\mathsf{T}$ for $c_1 \rightarrow c_4$,
- $2\mathsf{T}$ for $c_4 \rightarrow c_7$,
- $1\mathsf{T}$ for $c_7 \rightarrow s_7$.

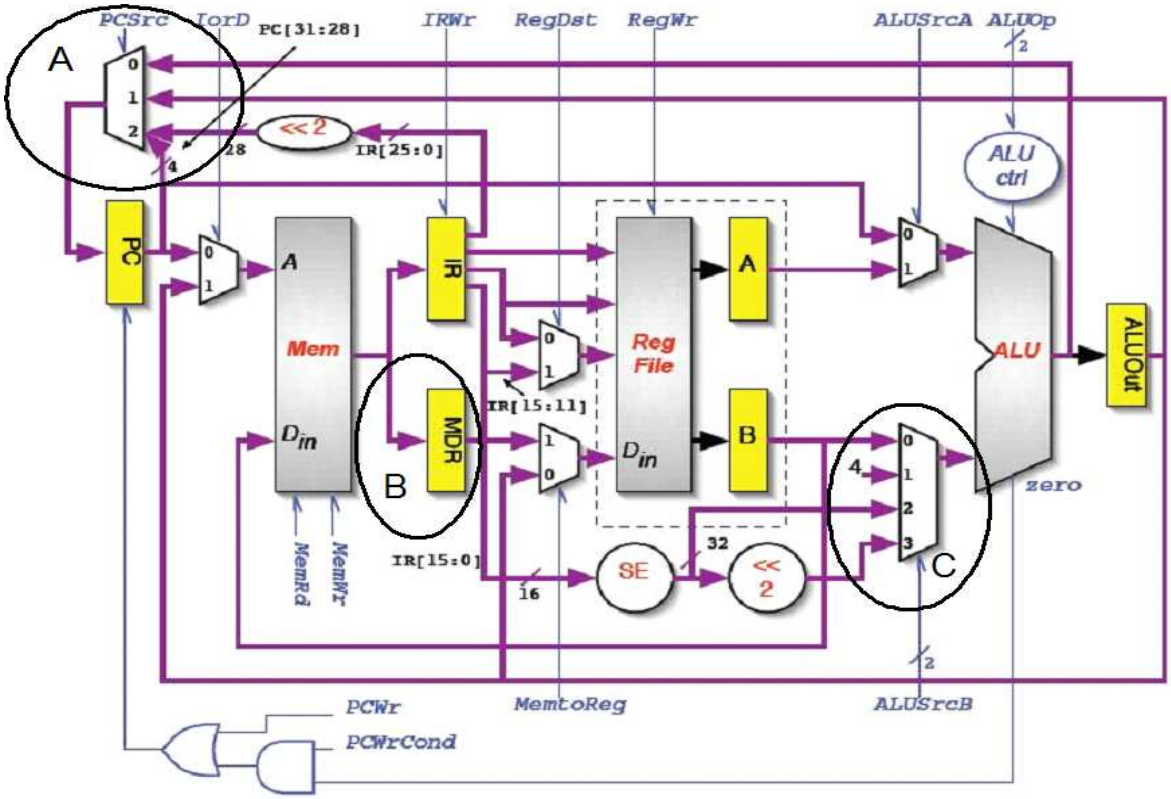
Therefore, the worst-case delay is $1\mathsf{T} + 2\mathsf{T} + 2\mathsf{T} + 1\mathsf{T} = 6\mathsf{T}$.

Two operands which would cause the maximum delay are 0111 1111 and 0000 0001.

5. The updated CPU diagram is shown further below.

- Circle A:
  - Connected ALUOut to the PC multiplexer.
  - `beq`: EX
- Circle B:
  - Connected memory output to MDR.
  - `lw`: DM

- Circle C:
  - Provide the value 4 as an input to the ALU multiplexer.
  - `j, beq, add/sub, lw, sw`: IF



6.

	0	1	2	3
IorD	0	-	-	1
MemRd	1	-	-	0
MemWr	0	0	0	1
IRWr	1	0	0	0
ALUSrcA	0	0	1	-
ALUSrcB	01	11	10	-
ALUOp	00	00	00	-
PCSrc	00	-	-	-
PCWr	1	0	0	0
PCWrCond	-	0	0	0
RegWr	0	0	0	0
RegDst	-	-	-	-
MemtoReg	-	-	-	-
	IF	ID/RF	EX	DM

7.

Instruction type	No. of clock cycles	Phases (IF, ID/RF, EX, DM, WB)
`lb`	5	IF, ID/RF, EX, DM, WB
`sb`	4	IF, ID/RF, EX, DM
`beq`	3	IF, ID/RF, EX
`j`	3	IF, ID/RF, EX

8. $\mathsf{CPI}_{\text{multi}} = 0.3 \times 4 + 0.2 \times 5 + 0.25 \times 4 + 0.15 \times 3 + 0.1 \times 3 = 3.95$.

   Let the clock period of the multi-cycle implementation be T. Then, the clock period of the single-cycle implementation is $5T$ because `lw` (which is the instruction that takes the maximum number of clock cycles to execute) takes 5 clock cycles. Let I be the number of instructions.

   $\mathsf{ExecutionTime}_{\text{single}} = \mathsf{CPI}_{\text{single}} \times I \times 5T = 5IT$.
   $\mathsf{ExecutionTime}_{\text{multi}} = \mathsf{CPI}_{\text{multi}} \times I \times T = 3.95IT$.

   Speed Up $= 5/3.95 = 1.27$.