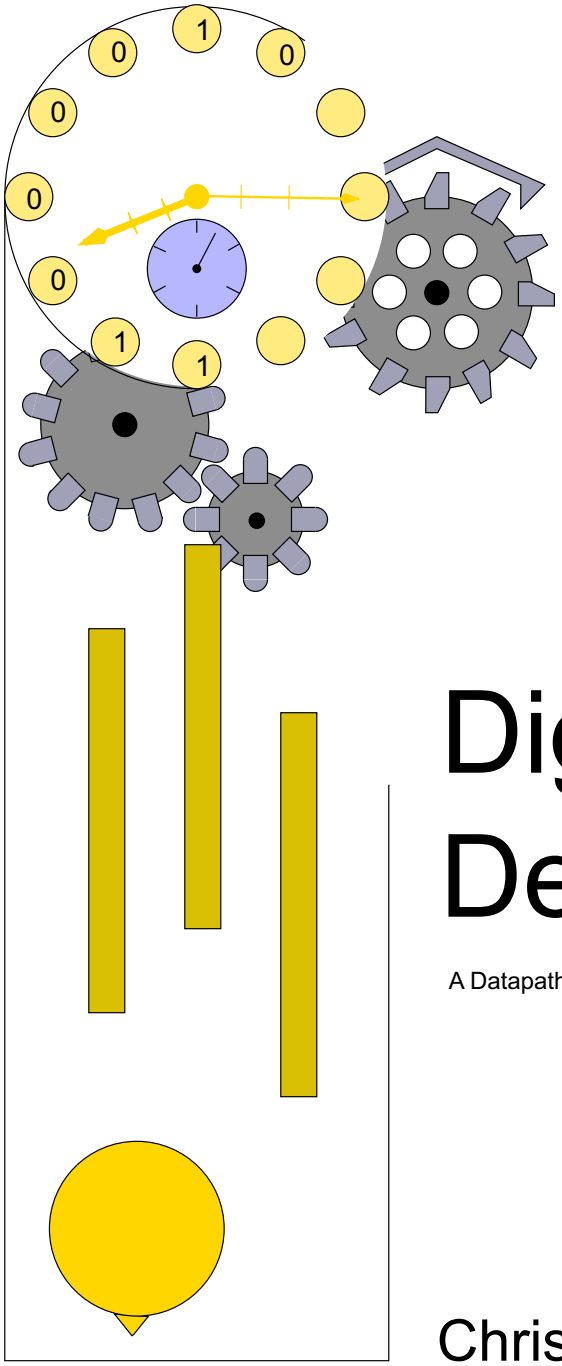


Digital Design

A Datapath and Control Approach

Chris Coulston

October 2, 2024

This document was prepared with LaTeX.

Contents

Chapter 1

Numbering Systems

1.1 Exercises

1. **(1 pt. each)** Syllabus:

   a) What is the late penalty for homework?
   There is a 33% deduction per day.

   b) True or False: Calculators can be used during exams.
   You cannot use calculators at my exams.

   c) True of False: University ID is required during exams.
   I check ID at the exams. After I learn your names its not such a big deal, but bring it to be safe.

   d) What is my thesis regarding grades?

   e) Bob L. Student has the following grades. Determine his final overall course percentage and grade.

Component	Percentage
Homework	60%
Exam 1	90%
Exam 2	80%
Final	70%

Component	*Percentage*	*Weight*
Homework	60%	*60*0.35 = 21*
Exam 1	90%	*90*0.20 = 18*
Exam 2	80%	*80*0.20 = 16*
Final	70%	*70*0.25 = 17.5*
Total	72.5%	*C*

   f) How should you prepare for the 43^{rd} lecture?
   Look over homework problem 8.10, page 165

2. **(1 pt. each)** Convert the following numbers to decimal. Show work, or receive 1/2 credit.

   a) 100_2 $100_2 = 2^2 = 4_{10}$

   b) 1000_2 $1000_2 = 2^3 = 8_{10}$

   c) 10000_2 $10000_2 = 2^4 = 16_{10}$

   d) 100000_2 $100000_2 = 2^5 = 32_{10}$

   e) 111111_2 $111111_2 = 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 63_{10}$

   f) 1000100101000101_2 $1000100101000101_2 = 2^{15} + 2^{11} + 2^8 + 2^6 + 2^5 + 2^0 = 35141_{10}$

   g) $3EA_{16}$ $3EA_{16} = 001111101010 = 2^9 + 2^8 + 2^7 + 2^6 + 2^5 + 2^3 + 2^1 = 1002_{10}$

3. **(1 pt. each)** Convert the following number to binary. Show work, or receive 1/2 credit.

   a) 44_{16} $44_{16} = 01000100_2$

   b) 44_{10} $44_{10} = 32 + 8 = 2^5 + 2^3 = 101100_2$

   c) 1023_{10} $1023_{10} = 512 + 256 + 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 2^9 + 2^8 + 2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 1111111111_2$

4. **(1 pt. each)** Convert the following number to hex. Show work, or receive 1/2 credit.

   a) 101011101_2 $101011101_2 = 15D_{16}$

   b) 77_{10} $77_{10} = 64 + 8 + 4 + 1 = 2^6 + 2^3 + 2^2 + 2^0 = 1001101_2 = 4D_{16}$

5. **(2 pts. each)** Toughies:

   a) Convert 123_5 to base-12 $123_5 = 1 * 5^2 + 2 * 5^1 + 3 * 5^0 = 25 + 10 + 3 = 38_{10} = 3 * 12^1 + 2 * 12^0 = 32_{12}$

   b) Convert 789_{12} to base-5 $789_{12} = 7*12^2+8*12^1+9*12^0 = 1008+96+9 = 1113_{10} = 1 * 5^4 + 3 * 5^3 + 4 * 5^2 + 2 * 5^1 + 3 * 5^0 = 13423_5$

   c) What is the largest base-10 quantity that can be represented using 5 digits in base 12?

   $BBBBB_{12} = 11 * 12^4 + 11 * 12^3 + 11 * 12^2 + 11 * 12^1 + 11 * 12^0 = 248831_{10}$

6. **(1 pt. each)** Perform the following additions, assume a word size of four bits. Determine if overflow occurs.

   a) $0110_2 + 0101_2$ *0110 + 0101 = 1011*

   b) $0010_2 + 0110_2$ *0010 + 0110 = 1000*

   c) $0111_2 + 0011_2$ *0111 + 0011 = 1010*

   d) $0010_2 + 0101_2$ *0010 + 0101 = 0111*

   e) $0010_2 + 1010_2$ *0010 + 1010 = 1100*

   f) $0101_2 + 1011_2$ *0101 + 1011 = 10000 overflow*

   g) $0011_2 + 1001_2$ *0011 + 1001 = 1100*

Chapter 2

Representations of Logical Functions

2.1 Exercises

1. **(2 pts. each)** Given: $F(A,B,C,D) = (AB' + (C + (AD)')(BD))'$

   a) Determine the truth table for $F(A,B,C,D)$

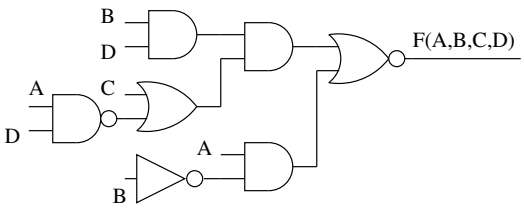
   Solution

   Let T3 = C+(AD)'
   T4 = BD
   *T1 = AB' T5 = T3 * T4*

A	*B*	*C*	*D*	*AB'*	*(AD)'*	*C+(AD)'*	*BD*	*T3*T4*	*T1+T5*	*F*
0	*0*	*0*	*0*	*0*	*1*	*1*	*0*	*0*	*0*	*1*
0	*0*	*0*	*1*	*0*	*1*	*1*	*0*	*0*	*0*	*1*
0	*0*	*1*	*0*	*0*	*1*	*1*	*0*	*0*	*0*	*1*
0	*0*	*1*	*1*	*0*	*1*	*1*	*0*	*0*	*0*	*1*
0	*1*	*0*	*0*	*0*	*1*	*1*	*0*	*0*	*0*	*1*
0	*1*	*0*	*1*	*0*	*1*	*1*	*1*	*1*	*1*	*0*
0	*1*	*1*	*0*	*0*	*1*	*1*	*0*	*0*	*0*	*1*
0	*1*	*1*	*1*	*0*	*1*	*1*	*1*	*1*	*1*	*0*
1	*0*	*0*	*0*	*1*	*1*	*1*	*0*	*0*	*1*	*0*
1	*0*	*0*	*1*	*1*	*0*	*0*	*0*	*0*	*1*	*0*
1	*0*	*1*	*0*	*1*	*1*	*1*	*0*	*0*	*1*	*0*
1	*0*	*1*	*1*	*1*	*0*	*1*	*0*	*0*	*1*	*0*
1	*1*	*0*	*0*	*0*	*1*	*1*	*0*	*0*	*0*	*1*
1	*1*	*0*	*1*	*0*	*0*	*0*	*1*	*0*	*0*	*1*
1	*1*	*1*	*0*	*0*	*1*	*1*	*0*	*0*	*0*	*1*
1	*1*	*1*	*1*	*0*	*0*	*1*	*1*	*1*	*1*	*0*

   b) Draw a schematic of the logic circuit which realizes F as shown, i.e. do not use Boolean Algebra on F.

   Solution



2. **(2 pts. each)** For the circuit in Figure 2.1

   a) Write a Boolean expression for the function.
   Solution *F(A,B,C,D) = (AB+C)D'+ABD'*

   b) Draw the truth table for the function.
   Solution

A	B	C	D	AB+C	(AB+C)D'	ABD'	F
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	1	0	0	1
0	0	1	1	1	1	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	1	0	0
1	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0
1	0	1	0	1	0	0	1
1	0	1	1	1	1	0	0
1	1	0	0	1	0	1	1
1	1	0	1	1	1	0	0
1	1	1	0	1	0	1	1
1	1	1	1	1	0	0	0

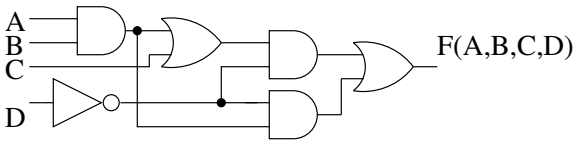


Figure 2.1: The circuit for Problems 2 and 3.

3. **(2 pts. each)** For the functions F,G,H,I defined by the truth table shown below:

   a) Determine the canonical SOP and POS realization for F, G, H, I.

   Solution

   F(A,B,C) = (A+B+C)(A+B'+C')(A'+B+C')(A'+B'+C) = A'B'C + A'BC' + AB'C' + ABC

   G(A,B,C) = (A'+B+C)(A'+B'+C') =
   A'B'C' + A'B'C + A'BC' + A'BC + AB'C + ABC'

   H(A,B,C) = (A+B'+C)(A+B'+C')(A'+B+C)(A'+B+C')(A'+B'+C') =
   A'B'C' + A'B'C + ABC'

   I(A,B,C) = (A+B+C)(A+B'+C)(A'+B+C)(A'+B'+C) =
   A'B'C + A'BC + AB'C + ABC

   b) Draw the circuit diagram for the canonical SOP and POS realization.

   Solution

Treat each output independently of the other. For example when working with function I, cover up the columns F, G and H.

A	B	C	F	G	H	I
0	0	0	0	1	1	0
0	0	1	1	1	1	1
0	1	0	1	1	0	0
0	1	1	0	1	0	1
1	0	0	1	0	0	0
1	0	1	0	1	0	1
1	1	0	0	1	1	0
1	1	1	1	0	0	1

4. **(2 pts. each)** Prove the validity of the following statements using the laws of Boolean Algebra. For each step of the proof, identify which law was used.

   a) $X'Y' + XY + X'Y = X' + Y$

   Solution

   X'Y' + XY + X'Y = *3D*
   X'Y' + X'Y + XY+ X'Y = *8*
   X'(Y' + Y)+ Y(X+ X')Y= *5*
   X' + Y *QED*

   b) $(X + Y')X'Y' = X'Y'$

   Solution

   (X+Y')X'Y' = *8*
   XX'Y + X'Y'Y' = *5D*
   0+X'Y'= *1*
   X'Y' *QED*

   c) $(X + Y)(X' + Z) = XZ + X'Y$

   Solution

   (X+Y)(X'+Z) = *8*
   (X+Y)X' + (X+Y)Z = *8*
   XX' + YX' + XZ + YZ = *1D,5*
   YX' + XZ + YZ(X+X') = *8*
   YX' + XZ + XYZ + X'YZ = *6*
   X'Y + X'YZ + XZ + XYZ = *1D, 8*
   X'Y(1+Z) + XZ(1+Y) = *2, 1D*
   X'Y + XZ *QED*

   d) $X'Y' + (X + Y)Z = X'Y' + Z$

   Solution

   X'Y' + (X+Y)Z = *8*
   X'Y' + XZ + YZ = *1D,5*
   X'Y'(Z+Z') + XZ + YZ(X+X') =* *8*
   X'Y'Z' + X'Y'Z + XZ + XYZ + X'YZ = *3*
   X'Y'Z' + X'Y'Z + X'Y'Z + XZ + XYZ + X'YZ = *8*
   X'Y'(Z+Z') + XZ(1+Y) + X'Z(Y'+Y) = *5,1D*
   X'Y' + XZ + X'Z = *8*
   X'Y' + Z(X+X') = *5, 1D*
   X'Y' + Z *QED*

e) $A'C + BC + AB = A'C + AB$

Solution

A'C + BC+AB =	*1D, 5*
A'C + (A+A')BC + AB(C+C')	*8*
A'C + ABC + A'BC + ABC + ABC'=	*3*
A'C + A'BC + ABC + ABC'=	*8*
A'C(1+B) + AB(C+C')=	*5, 1D*
A'C + AB	*QED*

f) $A(B + C) = AB + AB'C$

Solution

AB+AB'C=	*1D,5*
AB(C+C') + AB'C=	*8*
ABC+ABC'+AB'C=	*3*
ABC + ABC + ABC' + AB'C=	*6*
ABC + ABC' + ABC + AB'C=	*8*
AB(C+C'+C) + AB'C =	*8*
AB + AB'C=	*QED*

g) $(A + B + C)(A + B + C')(A' + B + C')(A' + B' + C') = (A + B)(A' + C')$

Solution

(A+B+C)(A+B+C')(A'B+C')(A'+B'+C')=	*4*
((A+B+C)(A+B+C')(A'B+C')(A'+B'+C'))"=	*9D*
(A'B'C + A'B'C' + AB'C + ABC)'=	*8*
(A'B'(C+C') + AC(B'+B))'=	*5, 1D*
(A'B' + AC)'=	*9*
(A+B)(A'+C')=	*QED*

5. **(4 pts.)** Design a circuit called MUX2. MUX2 has three bits of input S, y_0, y_1 and one bit of output F. If $S = 0$, then $F = y_0$; else if $S = 1$, then $F = y_1$.

a) Write down the truth table for the MUX2 function.

Solution

S	*y0*	*y1*	*F*
0	*0*	*0*	*0*
0	*0*	*1*	*0*
0	*1*	*0*	*1*
0	*1*	*1*	*1*
1	*0*	*0*	*0*
1	*0*	*1*	*1*
1	*1*	*0*	*0*
1	*1*	*1*	*1*

b) Determine the canonical SOP realization for MUX2; do not simplify.

Solution *F = S'y0y1' + S'y0y1 + Sy0'y1 + Sy0y1*

6. **(6 pts.)** Design a circuit called MUX4. MUX4 has six bits of input $S_1S_0, y_0, y_1, y_2, y_3$ and one bit of output F.
If $S_1S_0 = 00$ then $F = y_0$
else if $S_1S_0 = 01$ then $F = y_1$
else if $S_1S_0 = 10$ then $F = y_2$

else if $S_1S_0 = 11$ then $F = y_3$
Without writing down the truth table determine a SOP expression to realize F by listing all possible inputs which will cause F to equal 1. Then try to simplify your expression using Boolean Algebra.

Solution

The output F only equals one in the following cases.

S1=0 S0=0 and y0=1

S1=0 S0=1 and y1=1

S1=1 S0=0 and y2=1

S1=1 S0=1 and y3=1

With this information we can form four product terms, one for each input, that equal 1 only for that input. ORing together these product terms will give us the solution to the problem.

$F = S_1'S_0'y_0 + S_1'S_0y_1 + S_1S_0'y_2 + S_1S_0y_3$

7. **(4 pts.)** Design a logic circuit called MAJ which has three inputs A, B, C and one output Z. The output equals 1 when a majority of the inputs are equal to 1, otherwise the output is 0.

   a) Write the truth table for the MAJ function.

   Solution

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

   b) Determine the canonical SOP realization for the MAJ function, do not simplify.

   Solution *F = A'BC + AB'C+ABC'+ABC*

8. **(4 pts.)** Let X and Y each be 2-bit signals whose elements are x_1x_0 and y_1y_0 respectively. Determine the $\sum m$ and $\prod M$ expression for a circuit whose 1-bit output z is defined by the following statement.

```
if (X == Y) then z = 1 else z =0
```

Solution

a_1	a_0	b_1	b_0	A	B	z
0	0	0	0	0	0	1
0	0	0	1	0	1	0
0	0	1	0	0	2	0
0	0	1	1	0	3	0
0	1	0	0	1	0	0
0	1	0	1	1	1	1
0	1	1	0	1	2	0
0	1	1	1	1	3	0
1	0	0	0	2	0	0
1	0	0	1	2	1	0
1	0	1	0	2	2	1
1	0	1	1	2	3	0
1	1	0	0	3	0	1
1	1	0	1	3	1	0
1	1	1	0	3	2	0
1	1	1	1	3	3	1

Yielding

$z = \sum m(0, 5, 10, 15) = \prod M(1, 2, 3, 4, 6, 7, 8, 9, 11, 12, 13, 14)$

9. **(4 pts.)** Let X and Y each be 2-bit signals whose elements are x_1x_0 and y_1y_0, respectively. Determine the $\sum m$ and $\prod M$ expressions for a circuit whose 1-bit output z is defined by the following statement.

```
if (X + Y > 3) then z = 0 else z =1
```

Solution

a_1	a_0	b_1	b_0	A	B	z
0	0	0	0	0	0	1
0	0	0	1	0	1	1
0	0	1	0	0	2	1
0	0	1	1	0	3	1
0	1	0	0	1	0	1
0	1	0	1	1	1	1
0	1	1	0	1	2	1
0	1	1	1	1	3	0
1	0	0	0	2	0	1
1	0	0	1	2	1	1
1	0	1	0	2	2	0
1	0	1	1	2	3	0
1	1	0	0	3	0	1
1	1	0	1	3	1	0
1	1	1	0	3	2	0
1	1	1	1	3	3	0

Leading to the answer $z = \sum m(0, 1, 2, 3.4, 5, 6, 8, 12) = \prod M(7, 9, 10, 11, 13, 14, 15)$

10. **(3 pts.)** Determine the canonical SOP and POS expression for $F(A, B, C) = \prod M(0, 1, 4, 5)$ Hint, compose the truth table for F.

    Solution

    F(A,B,C)=A'BC' + A'BC +ABC' +ABC
    F(A,B,C)=(A+B+C)(A+B+C')(A'+B+C)(A'+B+C')

11. **(3 pts.)** Determine the canonical SOP and POS expression for $F(A, B, C, D) = \sum m(0, 4, 12, 15)$ Hint, write out the truth table for F.

    Solution

    F(A,B,C)=A'B'C'D' + A'BC'D' + ABC'D' + ABCD
    F(A,B,C)=(A+B+C+D')(A+B+C'+D)(A+B+C'+D') (A+B'+C+D')(A+B'+C'+D)
    (A+B'+C'+D')(A'+B+C+D)(A'+B+C+D') (A'+B+C'+D)(A'+B+C'+D')(A'+B'+C+D')
    (A'+B'+C'+D)

12. **(4 pts.)** For the function $F(A, B, C) = BC + AB'C'$, draw a timing diagram for an input sequence that follows the same order as the rows of the truth table. Assume a propagation delay for NOT, AND and OR gate are all 10nS.

    Solution *skipped for now*

13. **(4 pts.)** Complete the timing diagram in Figure 2.2 for the functions $F(A, B, C) = AB' + BC + ABC'$ and $G(A, B, C) = (A + B')C + (BC')'$

    Solution

14. **(16 pts.)** Design a circuit to control the water pump of a washing machine. The pump will not pump water if

    The lid is closed and the cycle is not fill

    The cycle is fill and the detergent level is empty

    The detergent is not empty and the lid is open

    The variables for this problem are:

    L = lid is closed

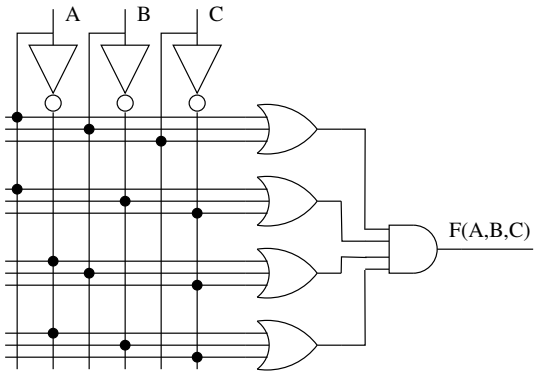
    C = cycle is fill

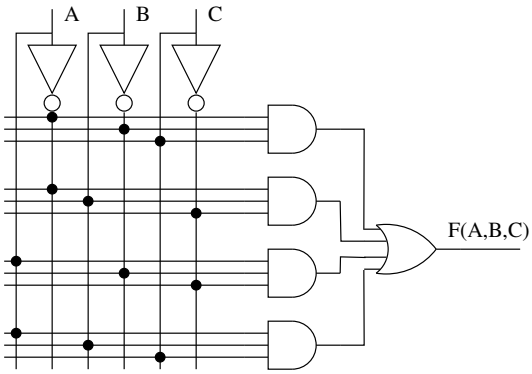
    D = detergent is empty

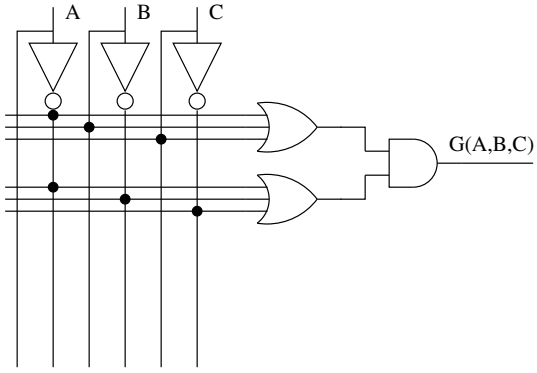
    P = pump will pump water

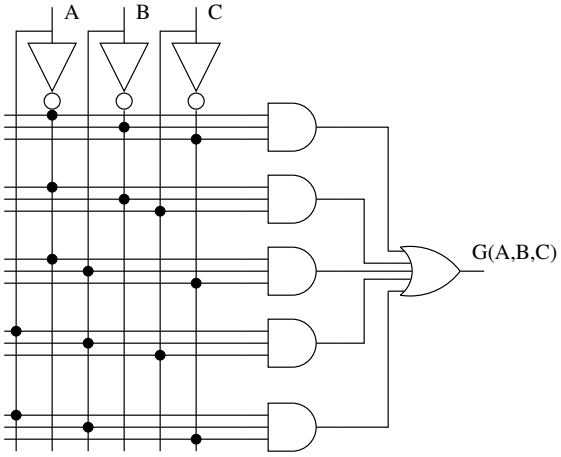
    Create a truth table which describes when the pump will not pump water. Call this output P'. Determine the canonical SOP expression for P'. Use this canonical SOP expression to generate a circuit diagram for P. This can be done by inserting an inverter onto the output of the circuit.

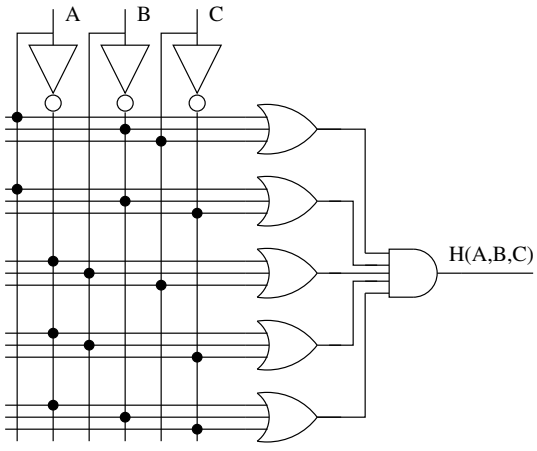
    Take the P' column from truth table and invert all the entries to generate a new output column called P (because the negation of P' is P). Determine the canonical SOP realization for P using this new column.

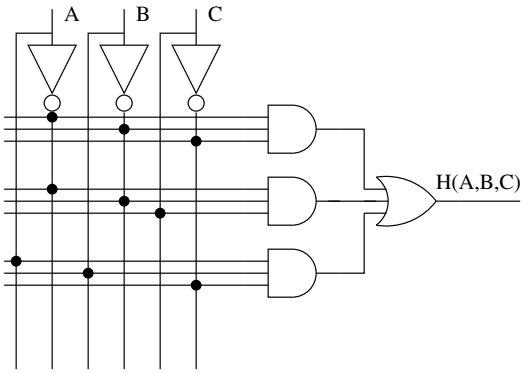

A
B
C
F(A,B,C)

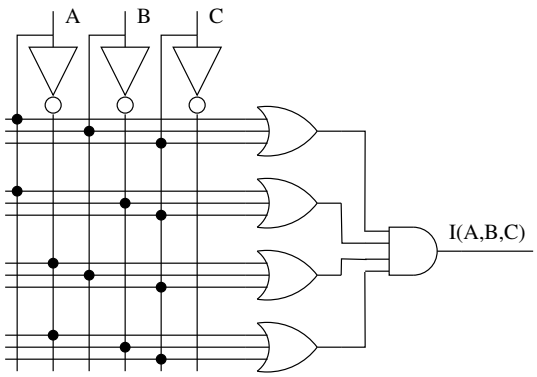

A
B
C
F(A,B,C)

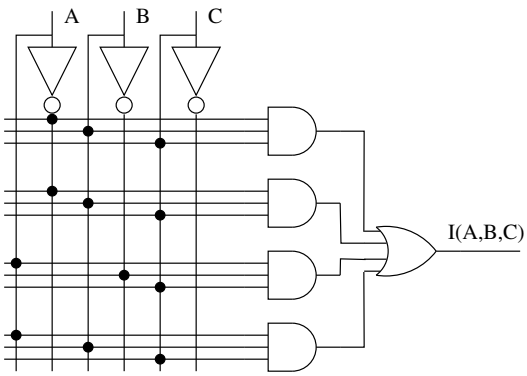

A
B
C
G(A,B,C)


A
B
C
G(A,B,C)


A
B
C
H(A,B,C)


A
B
C
H(A,B,C)


A
B
C
I(A,B,C)


A
B
C
I(A,B,C)

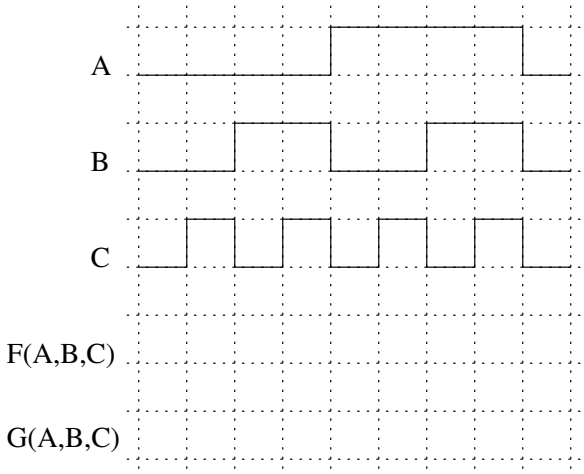
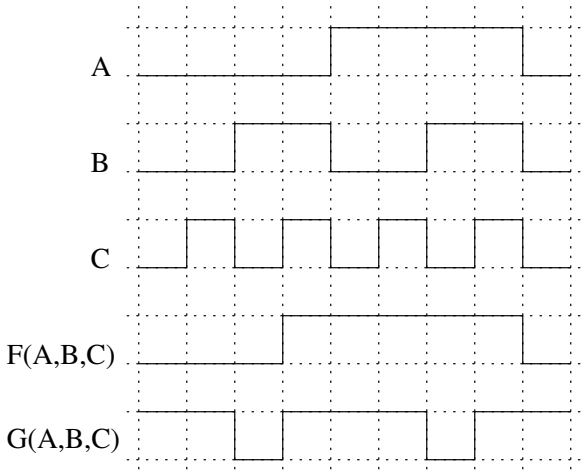


Figure 2.2: The timing diagram for two functions, F and G.



Chapter 3

Minimization of Logical Functions

3.1 Exercises

1. **(6 pts.)** Design a circuit called DECODE. DECODE has two bits of input S, D and two bit of output y_1y_0. If $S = 0$ then $y_0 = D$ and $y_1 = 0$ else if $S = 1$ then $y_0 = 0$ and $y_1 = D$.

   a) Write down the truth table for the DECODE function.

   Solution

S	D	y_1	y_0
0	*0*	*0*	*0*
0	*1*	*0*	*1*
0	*0*	*0*	*0*
1	*1*	*1*	*0*

   b) Determine the SOP $_{\min}$ realization for DECODE.

   Solution $y_0 = S'D$
   $y_1 = SD$

2. **(6 pts.)** Design a circuit called FULLADD. FULLADD has three bits of input a, b, c and two bits of output s_1s_0. The output represents the sum of the three bits.

   a) Write down the truth table for the FULLADD function.

   Solution

a	b	c	s_1	s_0
0	*0*	*0*	*0*	*0*
0	*0*	*1*	*0*	*1*
0	*1*	*0*	*0*	*0*
0	*1*	*1*	*1*	*1*
1	*0*	*0*	*0*	*0*
1	*0*	*1*	*1*	*1*
1	*1*	*0*	*1*	*0*
1	*1*	*1*	*1*	*1*

   b) Determine the SOP $_{\min}$ realization for FULLADD.

   Solution $s_1 = ab + ac + bc$
   $s_0 = a'b'c + a'bc' + ab'c' + abc$

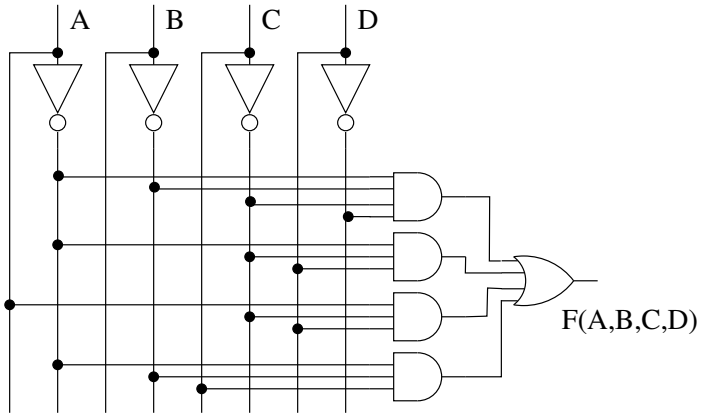
3. **(4 pts.)** Determine SOP $_{\min}$ expression for the following circuit and draw the circuit using the fewest number of gates possible.

   Solution *From the circuit we have: F(A,B,C,D) = A'B'C'D' + A'C'D + AC'D + A'B'C*

$AB\backslash CD$	00	01	11	10
00	1	1	1	1
01		1		
11		1		
10		1		

   From this it follows that F(A,B,C,D) = A'B' + C'D

4. **(8 pts.)** Design a digital system with four bits of inputs $I_3I_2I_1I_0$ and two bits of outputs O_1O_0. At least one of the inputs is always equal to 1. The output encodes the index of the most significant 1 in the input. For example, if $I_3I_2I_1I_0 = 0101$, then the index of the most significant 1 is 2, hence $O_1O_0 = 10$. Submit:



- The truth table.

Solution

I_3	I_2	I_1	I_0	O_1	O_0
0	*0*	*0*	*0*	x	x
0	*0*	*0*	*1*	*0*	*0*
0	*0*	*1*	*0*	*0*	*1*
0	*0*	*1*	*1*	*0*	*1*
0	*1*	*0*	*0*	*1*	*0*
0	*1*	*0*	*1*	*1*	*0*
0	*1*	*1*	*0*	*1*	*0*
0	*1*	*1*	*1*	*1*	*0*
1	*0*	*0*	*0*	*1*	*1*
1	*0*	*0*	*1*	*1*	*1*
1	*0*	*1*	*0*	*1*	*1*
1	*0*	*1*	*1*	*1*	*1*
1	*1*	*0*	*0*	*1*	*1*
1	*1*	*0*	*1*	*1*	*1*
1	*1*	*1*	*0*	*1*	*1*
1	*1*	*1*	*1*	*1*	*1*

- SOP $_{\text{min}}$ expression for O_1 and O_0.

Solution

$I3I2\backslash I1I0$	00	01	11	10
00	x			
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

$O_1 = I_3 + I_2$

$I3I2\backslash I1I0$	00	01	11	10
00	x		1	1
01				
11	1	1	1	1
10	1	1	1	1

$O_0 = I_3 + I_2' I_1$

5. **(8 pts.)** Design a 4-input $a_1a_0b_1b_0$, 4 -output $O_3O_2O_1O_0$ digital system. $A = a_1a_0$ and $B = b_1b_0$ represent 2-bit binary numbers. The output should be the product (multiplication) of the inputs, that is $O = A * B$. In addition to determining the output, determine the number of bits of output. Submit:

- Truth tables.

Solution

a_1	a_0	b_1	b_0	O_3	O_2	O_1	O_0
0	*0*	*0*	*0*	*0*	*0*	*0*	*0*
0	*0*	*0*	*1*	*0*	*0*	*0*	*0*
0	*0*	*1*	*0*	*0*	*0*	*0*	*0*
0	*0*	*1*	*1*	*0*	*0*	*0*	*0*
0	*1*	*0*	*0*	*0*	*0*	*0*	*0*
0	*1*	*0*	*1*	*0*	*0*	*0*	*1*
0	*1*	*1*	*0*	*0*	*0*	*1*	*0*
0	*1*	*1*	*1*	*0*	*0*	*1*	*1*
1	*0*	*0*	*0*	*0*	*0*	*0*	*0*
1	*0*	*0*	*1*	*0*	*0*	*1*	*0*
1	*0*	*1*	*0*	*0*	*1*	*0*	*0*
1	*0*	*1*	*1*	*0*	*1*	*1*	*0*
1	*1*	*0*	*0*	*0*	*0*	*0*	*0*
1	*1*	*0*	*1*	*0*	*0*	*1*	*1*
1	*1*	*1*	*0*	*0*	*1*	*1*	*0*
1	*1*	*1*	*1*	*1*	*0*	*0*	*1*

- Minimal SOP expression for the outputs.

Solution

$a1a0\,\backslash b1b0$	00	01	11	10
00				
01				
11			1	
10				

$O_3 = a_1a_0b_1b_0$

$a1a0\,\backslash b1b0$	00	01	11	10
00				
01				
11				1
10			1	1

$O_2 = a_1a_0'b_1 + a_1b_1b_0'$

Solution

$a1a0\,\backslash b1b0$	00	01	11	10
00				
01			1	1
11		1		1
10		1	1	

$O_1 = a_1b_1'b_0 + a_1a_0'b_0 + a_1'a_0b_1 + a_0b_1b_0'$

$a1a0\,\backslash b1b0$	00	01	11	10
00				
01		1	1	
11		1	1	
10				

$O_0 = a_0b_0$

6. **(8 pts.)** Design a 4-bit Gray-code to binary converter. A 4-bit gray-code is a sequence of 4-bit values where successive values differ by a single bit. For this problem use the sequence: 0000, 0001, 0011, 0010, 0110, 0111, 0101, 0100, 1100, 1101, 1111, 1110, 1010, 1011, 1001, 1000. The index of the 4-bit gray code is its binary value. For example, the 4-bit gray code 0111 is at index 5, therefore when presented with 0111 on its input, the converter should output 0101. Submit:

- A truth table for the converter.

Solution

a_3	a_2	a_1	a_0	f_3	f_2	f_1	f_0
0	*0*	*0*	*0*	*0*	*0*	*0*	*0*
0	*0*	*0*	*1*	*0*	*0*	*0*	*1*
0	*0*	*1*	*0*	*0*	*0*	*1*	*1*
0	*0*	*1*	*1*	*0*	*0*	*1*	*0*
0	*1*	*0*	*0*	*0*	*1*	*1*	*0*
0	*1*	*0*	*1*	*0*	*1*	*1*	*1*
0	*1*	*1*	*0*	*0*	*1*	*0*	*1*
0	*1*	*1*	*1*	*0*	*1*	*0*	*0*
1	*0*	*0*	*0*	*1*	*1*	*0*	*0*
1	*0*	*0*	*1*	*1*	*1*	*0*	*1*
1	*0*	*1*	*0*	*1*	*1*	*1*	*1*
1	*0*	*1*	*1*	*1*	*1*	*1*	*0*
1	*1*	*0*	*0*	*1*	*0*	*1*	*0*
1	*1*	*0*	*1*	*1*	*0*	*1*	*1*
1	*1*	*1*	*0*	*1*	*0*	*0*	*1*
1	*1*	*1*	*1*	*1*	*0*	*0*	*0*

- Four k-maps for the converter.

Solution

$a3a2\backslash a1a0$	00	01	11	10
00				
01				
11	1	1	1	1
10	1	1	1	1

$f_3 = a_3$

$a3a2\backslash a1a0$	00	01	11	10
00				
01	1	1	1	1
11				
10	1	1	1	1

$f_2 = a_3a_2' + a_3'a_2$

Solution

$a3a2\backslash a1a0$	00	01	11	10
00			1	1
01	1	1		
11	1	1		
10			1	1

$f_1 = a_2a_1' + a_2'a_1$

$a3a2\backslash a1a0$	00	01	11	10
00		1		1
01		1		1
11		1		1
10		1		1

$f_0 = a_1'a_0 + a_1a_0'$

- SOP $_{\text{min}}$ expression for the outputs, no product sharing please (use the `-Dso` command line option).
- Espresso file for the converter
- Espresso output in PLA format
- Compare the number of gates required in your solution versus the number of gates required by Espresso.

7. **(4 pts. each)** Determine SOP $_{\text{min}}$ expression for:

a) $F(A, B, C) = \sum m(0, 1, 3, 4, 5)$

Solution

$A\backslash BC$	00	01	11	10
0	1	1	1	
1	1	1		

F(A,B,C)= B'+A'C

b) $F(A, B, C, D) = \sum m(1, 5, 6, 7, 11, 12, 13, 15)$

Solution

$AB\backslash CD$	00	01	11	10
00		1		
01		1	1	1
11	1	1	1	
10			1	

F(A,B,C)= ABC'+A'C'D+ACD+A'BC

c) $F(A,B,C,D) = \sum m(0,2,5,6,8,11,12,13,14,15)$

Solution

$AB\backslash CD$	00	01	11	10
00	1			1
01		1		1
11	1	1	1	1
10	1		1	

F(A,B,C,D) = AB + BC'D + ACD + B'C'D' + A'CD'

d) $F(A,B,C,D,E) = \sum m(0,8,9,10,13,15,22,26,29,30,31)$

Solution

$BC\backslash DE$	00	01	11	10
00	1			
01				
11		1	1	
10	1	1		1

A=0

$BC\backslash DE$	00	01	11	10
00				
01				1
11		1	1	1
10				1

A=1

F(A,B,C,D,E) = BCE + A'C'D'E' + BC'DE + ACDE' + A'BD'E or
F(A,B,C,D,E) = BCE + A'C'D'E' + BC'DE + ACDE' + A'BC'D'

e) $F(A,B,C,D,E) = \sum m(0,2,4,5,7,10,13,15,18,21,24,26,28,29)$

Solution

$BC\backslash DE$	00	01	11	10
00	1			1
01	1	1	1	
11		1	1	
10				1

A=0

$BC\backslash DE$	00	01	11	10
00				1
01		1		
11	1	1		
10	1			1

A=1

F(A,B,C,D,E) = A'B'D'E' + CD'E+C'DE'+ABD'E'+A'CE

8. **(4 pts. each)** Determine SOP $_{\min}$ expression for:

a) $F(A,B,C,D) = \sum m(4,7,9,12,13,15) + \sum d(0,1,2,3,10,14)$

Solution

$AB\backslash CD$	00	01	11	10
00	x	x	x	
01	1		1	
11	1	1	1	x
10		1		x

F(A,B,C,D) = BC'D'+AC'D+BCD

b) $F(A,B,C,D) = \sum m(0,1,5,7,10,14,15) + \sum d(2,8)$

Solution

$AB\backslash CD$	00	01	11	10
00	1	1		x
01		1	1	
11			1	1
10	x			1

F(A,B,C,D) = B'D'+A'C'D+BCD+ABC

c) $F(A,B,C,D) = \sum m(0,1,3,4,15) + \sum d(10,12)$

Solution

$AB\backslash CD$	00	01	11	10
00	1	1	1	
01	1			
11	x		1	
10				x

F(A,B,C,D)=ABCD+A'C'D'+A'B'D

d) $F(A,B,C,D,E) = \sum m(2,3,5,7,11,13,17,19,29,31) + \sum d(1,4,9,16,25)$

Solution

$BC\backslash DE$	00	01	11	10
00		x	1	1
01	x	1	1	
11		1		
10		x	1	

A=0

$BC\backslash DE$	00	01	11	10
00	x	1	1	
01				
11		1	1	
10		x		

A=1

F(A,B,C,D,E)=A'D'E+A'C'E+A'B'C'D+B'C'E+ABCE +A'B'E

e) $F(A,B,C,D,E) = \sum m(2,3,6,10,12,13,14,18,25,26,28,29) + \sum d(11,27)$

Solution

$BC\backslash DE$	00	01	11	10
00			1	1
01				1
11	1	1		1
10			x	1

A=0

$BC\backslash DE$	00	01	11	10
00				1
01				
11	1	1		
10		1	x	1

A=1

F(A,B,C,D,E)=A'C'D+C'DE'+A'DE' +BCD' + ABC'E

9. **(8 pts. each)** Determine SOP $_{\min}$ and POS $_{\min}$ expressions for:

a) $F(A,B,C,D) = \sum m(0,1,2,5,8,10,13,15)$

Solution

$AB\backslash CD$	00	01	11	10
00	1	1		1
01		1		
11		1	1	
10	1			1

F

$AB\backslash CD$	00	01	11	10
00			1	
01	1		1	1
11	1			1
10		1	1	

F'

SOP $_{min}$ F(A,B,C,D) = B'D'+A'C'D+ABD

POS $_{min}$ F(A,B,C,D) = (B'+D)(A+C'+D')(A'+B+D')

b) $F(A,B,C,D) = \prod M(0,4,6,10,11,12)$

Solution

$AB\backslash CD$	00	01	11	10
00		1	1	1
01		1	1	
11		1	1	1
10	1	1		

F

$AB\backslash CD$	00	01	11	10
00	1			
01	1			1
11	1			
10			1	1

F'

SOP $_{min}$ F(A,B,C,D) = AB'C'+A'B'C+ABC+C'D+A'D or

SOP $_{min}$ F(A,B,C,D) = AB'C'+A'B'C+ABC+C'D+BD or
SOP $_{min}$ F(A,B,C,D) = AB'C'+A'B'C+ABC+A'D+BD
POS $_{min}$ F(A,B,C,D) = (A+C+D)(B'+C+D)(A+B'+D)(A'+B+C')

c) $F(A, B, C, D) = \sum m(0, 5, 7, 10, 11, 14) + \sum d(3, 12, 15)$

Solution

$AB\backslash CD$	00	01	11	10
00	1		x	
01		1	1	
11	x		x	1
10			1	1

F

$AB\backslash CD$	00	01	11	10
00		1	x	1
01	1			1
11	x	1	x	
10	1	1		

F'

SOP $_{min}$ F(A,B,C,D) = A'B'C'D'+A'BD+AC
POS $_{min}$ F(A,B,C,D) = (A'+C)(A+C'+D)(A+B+D')(B'+C+D) or
POS $_{min}$ F(A,B,C,D) = (A'+C)(A+C'+D)(B+C+D')(B'+C+D) or
POS $_{min}$ F(A,B,C,D) = (A'+C)(A+C'+D)(A+B+D')(A+B'+D) or
POS $_{min}$ F(A,B,C,D) = (A'+C)(A+C'+D)(B+C+D')(A+B'+D)

d) $F(A, B, C, D) = \prod M(2, 6, 7, 9, 15) * \prod d(4, 12, 13)$

Solution

$AB\backslash CD$	00	01	11	10
00	1	1	1	
01	x	1		
11	x	x		1
10	1		1	1

F

$AB\backslash CD$	00	01	11	10
00				1
01	x		1	1
11	x	x	1	
10		1		

F'

SOP $_{min}$ F(A,B,C,D) = A'C'+AD'+A'CD
POS $_{min}$ F(A,B,C,D) = (A+C'+D)(B'+C'+D')(A'+C+D')

e) $F(W, X, Y, Z) = WX'Z' + X'YZ + W'Y'Z + XYZ + WXY'$

Solution

$WX\backslash YZ$	00	01	11	10
00		1	1	
01		1	1	
11	1	1	1	
10	1		1	1

F

$WX\backslash YZ$	00	01	11	10
00	1			1
01	1			1
11				1
10		1		

F'

SOP $_{min}$ F(W,X,Y,Z) = W'Z+XZ+WY'Z'+WX'Y
POS $_{min}$ F(W,X,Y,Z) = (W+Z)(X'+Y'+Z)(W'+X+Y+Z')

f) $F(W, X, Y, Z) = (W + X' + Y')(W' + Z')(W + Y')$

Solution

We have that $F'(W, X, Y, Z) = W'XY + WZ + W'Y$

$WX\backslash YZ$	00	01	11	10
00			1	1
01			1	1
11		1	1	
10		1	1	

F'

$WX\backslash YZ$	00	01	11	10
00	1	1		
01	1	1		
11	1			1
10	1			1

F

SOP $_{min}$ F(W,X,Y,Z) = W'Y'+WZ'
POS $_{min}$ F(W,X,Y,Z) = (W'+Z')(W+Y')

Hint, the negation of a "Don't care" is a "Don't care".

10. **(3 pts.)** While grading homework for a digital design class the following question/answer pair is encountered. What is the problem with the answer given?

    Question: Generate the POS $_{\min}$ expression for $F(A,B,C) = \sum m(2,3,4,5)$
    Answer: $F(A,B,C) = (A + B')(A' + B)$

11. **(6 pts.)** Determine the SOP $_{\min}$ realization of the following function.

A	B	C	D	F(A,B,C,D)
x	1	1	x	0
0	x	0	1	0
x	x	0	0	x
x	0	1	x	1
1	x	0	1	1

Solution

$AB\backslash BC$	00	01	11	10
00	x	0	1	1
01	x	0	0	0
11	x	1	0	0
10	x	1	1	1

F(A,B,C) = AC'+B'C

12. **(6 pts.)** What is the worst function SOP $_{\min}$ of 3 variable that can be created? That is, define a function whose minimal SOP form has the largest possible number of product terms. What is the largest number of product terms that a 4-variable SOP $_{\min}$ expression can have? How about N variables?

    Solution *The worst function of three variable is shown in the Kmap below:*

$A\backslash BC$	00	01	11	10
0	1		1	
1		1		1

This function requires four product terms. Any additional minterm added to this Kmap would create a grouping with neighboring minterms, perhaps not decreasing the the number of product terms, but certainly making the product terms simpler. Its important to note that this configuration looks like a checkerboard. The worst function of four variables looks like:

$AB\backslash BC$	00	01	11	10
00	1		1	
01		1		1
11	1		1	
10		1		1

This function requires 8 product terms, each containing every variable. Boy that's one bad realization. In a general, the worst realization of an N variable function requires 2^{N-1} *product terms. This is because the checkerboard configuration can be applied to any Kmap, with half of the cells containing 1's. Since a function with N variables has* 2^N *cells, then 1/2 of these cells works out to* 2^{N-1}.

13. **(16 pts.)** Sometimes a logic circuit needs to output a logic 0 in order to produce some behavior. For example, an LED can be attached to a digital circuit output so that it

lights up when the circuit outputs a 0. This response is called an active low output; the output device is *active* then the digital output is *low*.

Build a digital circuit that takes as input two 2-bit numbers, A and B. The circuit has three outputs which drive three LEDs labeled G, L, and E. The G LED should be illuminated when A>B. The L LED should be illuminated when A<B. The E LED should be illuminated when A=B. The LEDs are illuminated when the circuit outputs a 0, otherwise they are turned off.

Determine SOP $_{\text{min}}$ expression for the G, L and E outputs. Determine POS $_{\text{min}}$ expression for the G, L and E outputs.

Chapter 4

Combinational Logic Building Blocks

4.1 Exercises

1. **(2 pts. each)**Short answer:

   a) How many 3:8 decoders would it take to build a 9:512 decoder?
   Solutions *A total of 73 decoders are required. There are 512/8 = 64 on the output layer, 64/8 = 8 in the middle layer and 1 on the input.*

   b) How many AND gates are there in a 2^N:1 mux?
   Solutions *Each input requires 1 AND gate, hence 2^N AND gates.*

   c) How many AND gates are there in a $2^N : 1$ mux which is constructed out of 2:1 muxes?
   Solutions *A 2^N Mux requires the following number of 2:1 muxes:*

$$2^N/2 + 2^N/4 + ... + 2^N/2^N =$$
$$2^N(1/2 + 1/4 + ... + 1/2^N) =$$
$$2^N(1 - 1/2^(N + 1)) =$$
$$2^N - 1$$

   Since each 2:1 mux contains 2 AND gates, the total number of AND gates is $2^(N + 1) - 2$.

   d) How many AND gates are there in a 2^N:1 mux which is constructed out of 2^L:1 muxes, assume that 2^N is an integer multiple of 2^L?
   Solutions *A 2^N Mux requires the following number of $2^L : 1$ muxes:*

$$2^N/2^L + 2^N/2^(2L) + ...2^N/2^(kL)$$
$$2^N(1/2^L + 1/2^(2L) + ...1/2^(kL))$$

   Where $k = N/L$. Each $2^L : 1$ mux requires 2^L AND gates for its construction, so the number of AND gates is the product of the number of $2^L : 1$ muxes and the number of AND gates in a single $2^L : 1$ mux, or:

$$2^N * 2^L(1/2^L + 1/2^(2L) + ...1/2^(kL))$$
$$2^N(1 + 1/2 + ...1/2^k)$$
$$2^N(2 - 1/2^(k + 1))$$
$$2^N(2 - 1/2^(N/L + 1)$$
$$2^(N + 1) - 2^(N - N/L - 1)$$

2. **(6 pts.)**Determine the SOP $_{\min}$ expression for each of the three outputs of a bit-slice of the comparator.

Solutions *The following five variable Kmap describes E_{out}*

$L_{in}G_{in}\backslash xy$	00	01	11	10
00	x	x	x	x
01	0	0	0	0
11	x	x	x	x
10	0	0	0	0

$E_{in} = 0$

$L_{in}G_{in}\backslash xy$	00	01	11	10
00	1	0	1	0
01	x	x	x	x
11	x	x	x	x
10	x	x	x	x

$E_{in} = 1$

$$E_{out} = L'_{in}G'_{in}x'y' + L'_{in}G'_{in}xy$$

The following five variable Kmap describes G_{out}

$L_{in}G_{in}\backslash xy$	00	01	11	10
00	x	x	x	x
01	1	1	1	1
11	x	x	x	x
10	0	0	0	0

$E_{in} = 0$

$L_{in}G_{in}\backslash xy$	00	01	11	10
00	0	0	0	1
01	x	x	x	x
11	x	x	x	x
10	x	x	x	x

$E_{in} = 1$

$$G_{out} = G_{in} + L'_{in}xy'$$

The following five variable Kmap describes L_{out}

$L_{in}G_{in}\backslash xy$	00	01	11	10
00	x	x	x	x
01	0	0	0	0
11	x	x	x	x
10	1	1	1	1

$E_{in} = 0$

$L_{in}G_{in}\backslash xy$	00	01	11	10
00	0	1	0	0
01	x	x	x	x
11	x	x	x	x
10	x	x	x	x

$E_{in} = 1$

$$L_{out} = L_{in} + G'_{in}x'y$$

3. **(2 pts.)**Show how to connect together four 4-bit comparators to construct a 16-bit comparator.

   Solutions *Figure forthcoming*

4. **(2 pts.)**Determine the circuitry for the overflow detection circuit for a 2's-complement adder subtractor. See page **??**.

   Solutions

$c_{in}\backslash c_{out}$	0	1
0		1
1	1	

   Thus $ovf = c'_{in}c_{out} + c_{in}c'_{out} = c_{in} \oplus c_{out}$

5. **(10 pts.)**Build a BCD to 7-Segment Display converter using Espresso.

Nomenclature:	BCD to 7-segment converter
Data Input:	4-bit vector $D = d_3d_2d_1d_0$
Data Output:	7-bit vector $Y = y_6 \ldots y_1y_0$
Control:	none
Status:	none
Behavior:	The output drives a 7-segment display pattern representing the BCD digit.

   A binary coded digit (BCD) is a 4-bit binary number that is constrained to assume the values of 0-9. That is, 1010 ... 1111 are illegal BCD digits.

   A 7-segment display is a box with seven inputs and seven output LED bars. Each input is wired to an LED bar that is illuminated when a 1 is applied to its input. Each of the seven LED segments is numbered according to the pattern shown on the left-hand side of Figure 4.1.

   The pattern of LEDs to illuminate for each BCD digit is shown on the right-hand side of Figure 4.1. A BCD to 7-segment converter has four inputs, $d_3d_2d_1d_0$ and seven outputs $S_7 \ldots S_1$. Complete the design using Espresso. Make sure to include "Don't cares" in the truth table specification.

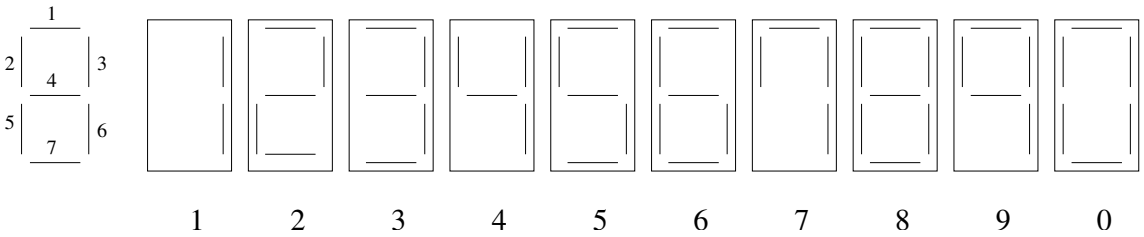


Figure 4.1: The numbering of the segments in a 7-segment display. The patterns of the BCD digits.

a) Use Espresso to determine the SOP $_{\min}$ expression for the outputs $S_7 \ldots S_1$. Underline product terms that are shared. Submit the Espresso source file.

Solutions *The following is the source file for the BCD to 7-segment converter.*
The following is the output from espresso on the BCD to 7-segment converter.

b) Use Espresso to determine the POS $_{\min}$ expression for the outputs $S_7 \ldots S_1$. Underline sum terms that are shared. Submit the Espresso source file **Solutions** *The following output was generated by using the same file as the solution in the previous part; but using the epos option in espresso.*
Remember that these are the negation of the output variables, hence we have to use DeMorgan's to put them into POS $_{min}$ form. Symbolically we have: s7=(b3+b2+b1+b0')(b2'+
s6=(b2+b1'+b0);
s5=(b2'+b1+b0')(b3+b2+b1+b0')(b2+b1+b0')(b2'+b1+b0)(b3'+b0')(b2'+b1'+b0');
s4=(b3+b2+b1)(b2'+b1'+b0');
s3=(b2'+b1'+b0)(b2'+b1+b0');
s2=(b2+b1'+b0)(b3+b2+b1+b0')(b2+b1'+b0');
s1=(b3+b2+b1+b0')(b2'+b1+b0);

6. **(10 pts.)** Build a box which has one 4-bit input called A and one 4-bit output called T. The output T is the 2's-complement value of the input A. Use the bit slice paradigm to solve this problem. That is, create a building block for one bit of the problem then string four of them together to solve the problem. For the problem at hand this can be done as follows:

   a) Start at the LSB of A.
   b) If this is the first, least significant, 1, flip all bits to the left.
   c) If this is not the first 1, leave the bit alone.
   d) Move one bit to the left.
   e) Goto Step b.

   A bit-slice should communicate whether there has been a 1 to the right, to the more significant bit. Submit:

   - How the above "algorithm" behaves when presented with the inputs A=1100
   - The truth table for one bit slice
   - SOP $_{\min}$ expression and circuit diagram for a bit slice.

A	c_{in}	T	c_{out}	comment
0	0	0	0	There is no 1 to the right
0	1	1	1	There is a 1 to the right, flip A
1	0	1	1	There is is no 1 to the right, but we've created one
1	1	0	1	There is a 1 to the right, flip A

- The organization of four bit slices to solve the problem

Solutions *The key to the begin solution is to figure out the structure of the begin solution and then give meaning to the signals involved. The problem will be sliced into four bit-slices; each handled but its own complement box. Thus, there will be four complement box in the begin solution. Each box will have 2 inputs, one being a bit of A and the other being a "carry in" from the less significant less, immediately to the right. Each box will have two bits of output, a bit of T and a "carry out". The carry bits (both into and out of a box will convey information regarding the rightmost 1 in the number A.*

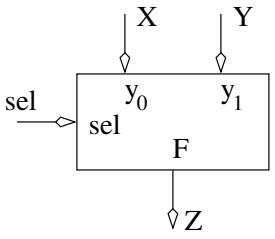
If the carry in is equal 1 then there is a one to the right. If the carry in is equal 0 then there is not a one to the right. The truth table for a box is then

From this it follows that:
$T = A'c_{in} + Ac'_{in} = A \oplus c_{in}$
$c_{out} = A + c_{in}$

7. **(4 pts.)** Build a 7:128 decoder using a minimum number of 4:16, 2:4 and 1:2 decoders. Describe the wiring of the select lines. **Solutions**

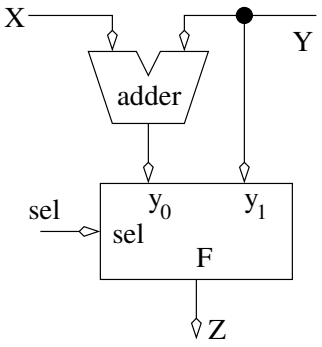
8. **(4 pts. each)** Design a circuit with two 8-bit inputs X, Y, an 8-bit output Z and a 1-bit input *sel*. Construct a circuit that yields the correct value of Z using only the basic building blocks presented in this chapter; do NOT show the internal organization of these building blocks. If a mux is used, denote which input is the y_0 and which is y_1. If a comparator is used denote which input is X and which is Y. Do not use any AND or OR gates; it will tempting in the later problems.

a) `if (sel==0) then Z = X else Z = Y`



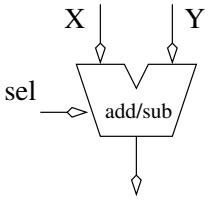
Solutions

b) `if (sel==0) then Z = X+Y else Z = Y`



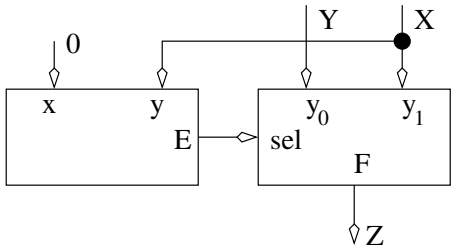
Solutions

c) `if (sel==0) then Z = X+Y else Z = X-Y`



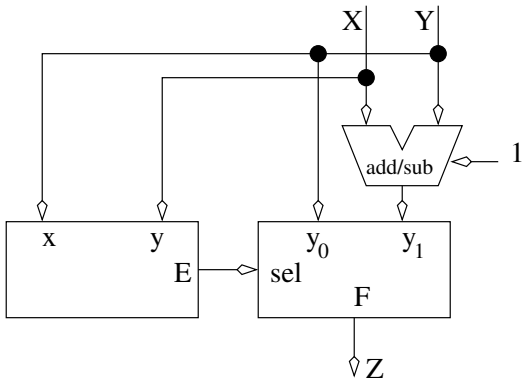
Solutions

d) `if (X==0) then Z = X else Z = Y`



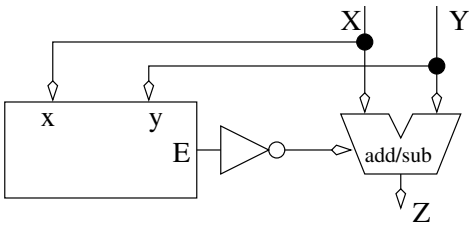
Solutions

e) `if (X==Y) then Z = X-Y else Z = Y`



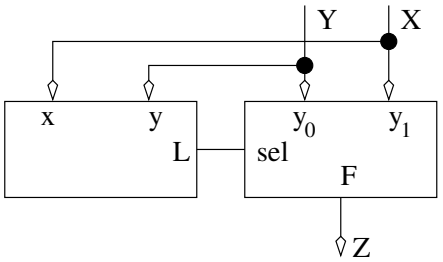
Solutions

f) `if (X==Y) then Z = X+Y else Z = X-Y`



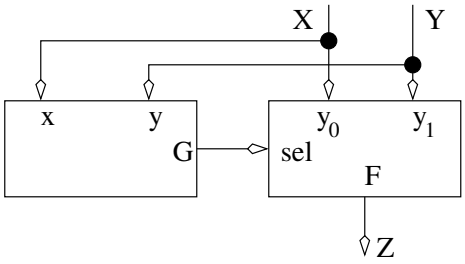
Solutions

g) `if (X < Y) then Z = X else Z = Y`



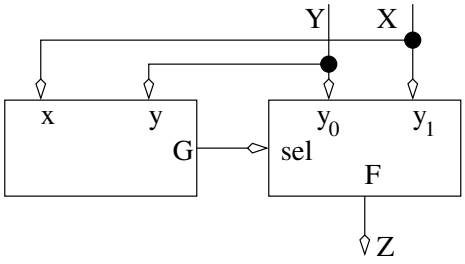
Solutions

h) `if (X <= Y) then Z = X else Z = Y`



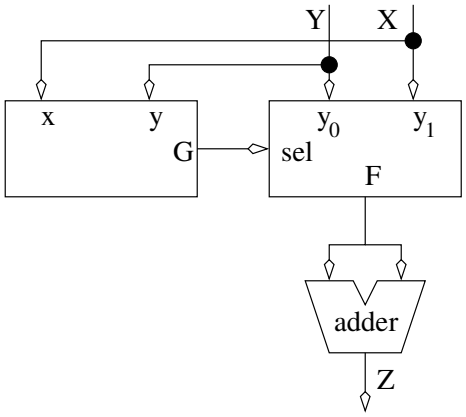
Solutions

i) `if (X > Y) then Z = X else Z = Y`



Solutions

j) `if (X > Y) then Z = X+X else Z = Y+Y`



Solutions

9. **(10 pts.)** Build a 4-bit priority encoder.

Nomenclature:	N-bit priority encoder
Data Input:	N-bit vectored $D = d_{N-1} \ldots d_1 d_0$
Data Output:	$\log_2(N)$-bit vector $Y = y_{log_2(N)} \ldots y_1 y_0$
Control:	none
Status:	none
Behavior:	$F = i$ where i is the highest indexed input which equals 1. When all inputs equal 0, the output is a "don't care".

The idea is for the outputs to represent (in binary code) the highest input index which equals 1. For example, a 4-bit priority encoder with input $D = 1010$ has inputs $d_3 = 1$ and $d_0 = 1$. Of these two inputs, the index of d_3 is greater than the index of d_0 so the output, F is equal to 3, or in binary 11. If the input were $D = 0111$ then $F = 10$.

a) Write down the truth table for a 4-bit priority encoder. Hint, the truth table could be structured so that it contains only five rows by using "don't cares" on the inputs.

Solutions

d_3	d_2	d_1	d_0	f_1	F_0
0	*0*	*0*	*0*	x	x
0	*0*	*0*	*1*	*0*	*0*
0	*0*	*1*	x	*0*	*1*
0	*1*	x	x	*1*	*0*
1	x	x	x	*1*	*1*

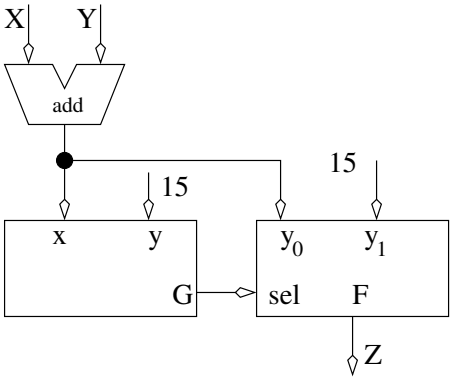
b) An SOP $_{\text{min}}$ realization of the circuit. **Solutions** $f_1 = d_3 + d_2$
$f_0 = d_3 + d_2' d_1$

10. **(10 pts.)** Build a 4-bit saturation adder. A saturation adder performs normal 4-bit addition when the resulting sum is less than 15. If the sum is greater than 15, the saturation adders outputs 15. The following table summarizes.

Nomenclature:	4-bit saturation adder
Data Input:	2, 4-bit vectors `A, B`
Data Output:	4-bit vector `sum`
Control:	none
Status:	none
Behavior:	`if (A+B > 15) sum = 15` `else sum = A+B`

Submit a schematic showing the basic building blocks, their data status, and control interconnections. Show any truth tables used to build glue logic.

Solutions *All we need to do is to determine when the sum is greater then 15 and output 15 when it is. The comparator/mux combo mentioned several times in the chapter should do the trick.*



11. **(10 pts.)** Build a mod-6 adder. The mod-6 adder takes as input two 3-bit (mod 6) numbers and adds them together modulus 6.

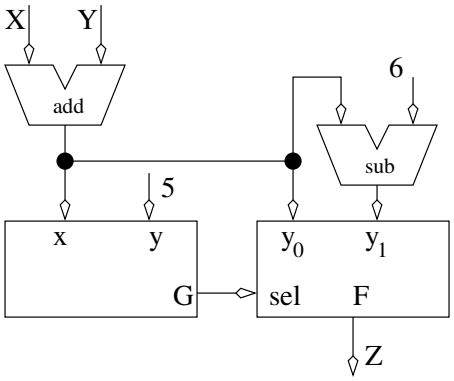
Modular arithmetic only operates with a limited portion of the integers. The range of numbers is $\{0, 1, 2, \ldots, m-1\}$ where m is called the *modulus*; note there are m different integers because counting started at 0. For example, when working in mod-6 arithmetic use the integers $\{0, 1, 2, 3, 4, 5\}$. To solve any addition problem in modular arithmetic, it is only necessary to perform regular addition with the special rule that the addition process rolls over from the largest number, $m-1$ to 0 when the result is larger than $m-1$. For example, in mod-6 arithmetic $(5+1) \mod 6 = 0$. The statement " mod 6" is always included in the addition problem to indicate to the reader that mod-6 arithmetic is being performed. Here are a few more examples to help

$2 + 3 \mod 6 = 5$
$3 + 3 \mod 6 = 0$
$4 + 3 \mod 6 = 1$
$5 + 5 \mod 6 = 4$

Nomenclature:	3-bit mod 6 adder
Data Input:	two, 3-bit (mod-6) vectors `A, B`
Data Output:	3-bit (mod-6) vector `sum`
Control:	none
Status:	none
Behavior:	`sum = A+B mod 6`

Submit a schematic showing the basic building blocks, their data status, and control interconnections. Show any truth tables used to build glue logic. Be careful that the word size of the result is handled correctly.

Solutions *Since the inputs are mod 6 numbers then the inputs can be in the range [0-5]. Adding two such values will yield a value in the range [0-10]. Hence a simple adjustment of the sum when its larger that 5 is required.*



12. **(1pt. each)**Convert the following to 2's-complement assuming a word size of eight bits.

    a) -35

    Solutions $35 = 32 + 2 + 1 = 100011 = 00100011$*, thus* $-35 = 11011101$

    b) -128

    Solutions *This is a special case, see page 10 for more information.* $-128 = 10000000$

    c) 67

    Solutions $67 = 64 + 2 + 1 = 1000011 = 01000011$

    d) 128

    Solutions *There are not enough bits to represent this positive number; hence the 8-bit representation does not exist.*

13. **(1 pt. each)** Perform the following operations for the given 2's-complement numbers. Assume a word size of eight bits in all cases. Indicate where overflow occurs. If there is no overflow, convert the result to decimal.

    a) 01011101 + 00110111

    Solutions *01011101 + 00110111 = 10010100 overflow*

b) 11101011 + 11110001
Solutions *11101011 + 11110001 = 11011100*

c) 01011101 + 10101011
Solutions *01011101 + 10101011 = 00001000*

d) 10111011 - 11110001
Solutions *10111011 - 11110001 =*
10111011 + 00001111 = 11001010

e) 01011101 - 00110111
Solutions *01011101 - 00110111 =*
01011101 + 11001001 = 00100110

f) 01011101 - 10101111
Solutions *01011101 - 10101111 =*
01011101 + 01010001 = 10101110, overflow

14. **(5 pts.)** Build a 4-bit bus transceiver. A bus transceiver is defined by the following truth table.

Nomenclature:	N-bit bus transceiver.
Data:	two bidirectional N-bits vectors $X = x_{N-1} \ldots x_1 x_0$. $Y = y_{N-1} \ldots y_1 y_0$.
Control:	1-bit F and R
Status:	none
Behavior:	F R X Y comment 0 0 Z Z X and Y tristate 0 1 Y Y X = Y 1 0 X X Y = X 1 1 x x never applied

Flow is determined by the F and R signals denoting forward and reverse respectively. When $F = 1$, data flows from X to Y. In this case, X is acting like and input and Y is acting like an output. When $R = 1$, data flows from the Y input to the X output. This design relies heavily on tristate buffers.

Solutions

15. **(3 pts.)** Build a 2:1 mux using some tristate buffers and an inverter.

16. **(3 pts.)** Build a 4:1 mux using some tristate buffers and two inverters.

Solutions

17. **(10 pts.)** Build a flip box. A flip box is defined by the following input, output, and behavior definition.

Nomenclature:	8-bit flip box.
Data Input:	8-bit $D = d_7 \ldots d_0$
Data Output:	8-bit $F = f_7 \ldots f_0$
Control:	3-bit $S = s_2 s_1 s_0$
Status:	none
Behavior:	The output is the same as the input except for one bit which is inverted. The index of the inverted bit is given by S.

Key	scancode	Key	scancode	Key	scancode	Key	scancode
0	45_{16}	1	16_{16}	2	$1E_{16}$	3	26_{16}
4	25_{16}	5	$2E_{16}$	6	36_{16}	7	$3D_{16}$
8	$3E_{16}$	9	46_{16}	A	$1C_{16}$	B	32_{16}
C	21_{16}	D	23_{16}	E	24_{16}	F	$2B_{16}$
P	$4D_{16}$	L	$4B_{16}$	M	$3A_{16}$	I	43_{16}

Table 4.1: Some keyboard scancodes.

The flip box takes the 8-bit data input, flips a single bit identified by S, then sends the new 8-bit value to the output. For example, if $D = 11110000$ and $S = 010$ then $F = 11110100$. If $D = 11110000$ and $S = 101$ then $F = 11010000$. The solution should rely heavily on the basic building blocks.

Solutions *Arrange 8, 2:1 muxes with d_i and d_i' going into the data inputs. Run the select into a 3:8 decoder and route the data outputs to the individual selects of the 2:1 muxes.*

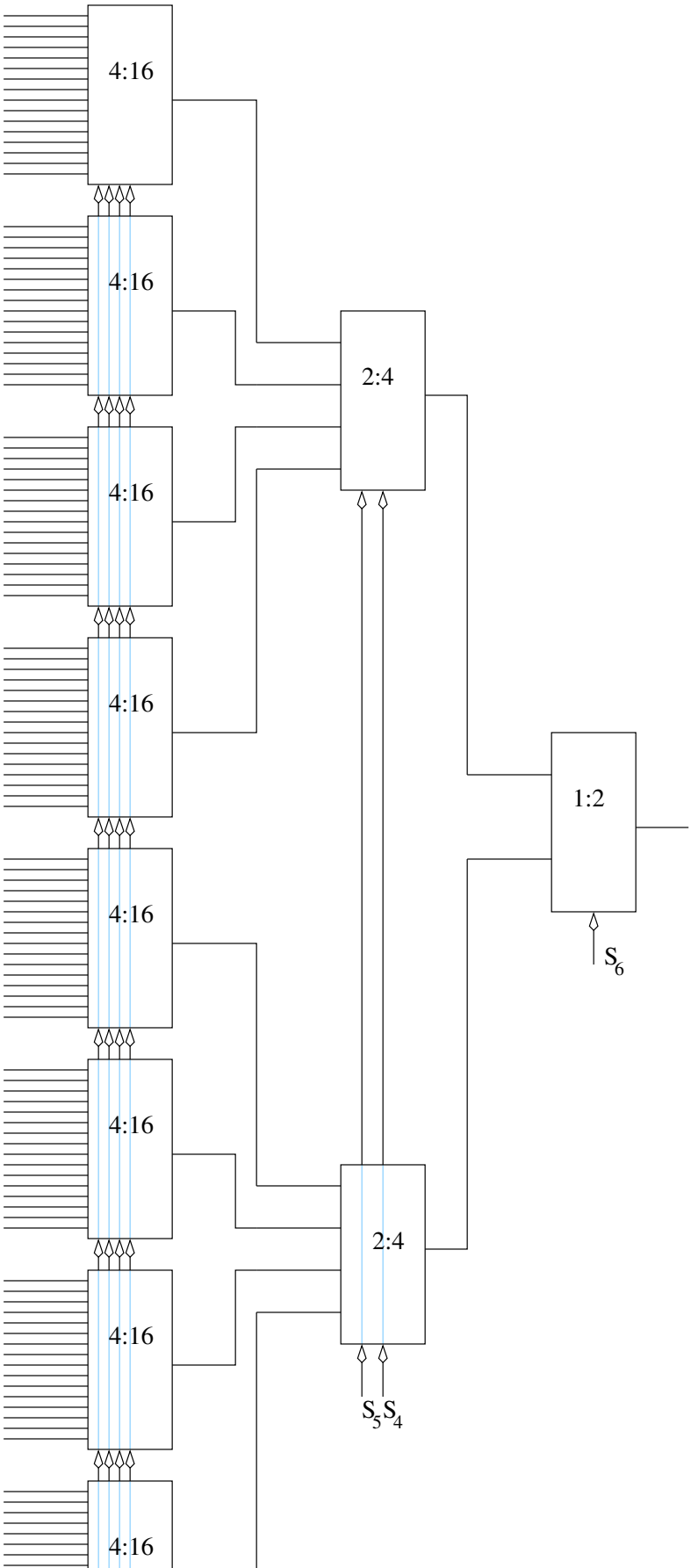
18. **(10 pts.)** Build a box which recognizes some keyboard scancode. When a key is pressed on a keyboard, the keyboard transmits (among other things) an 8-bit scancode of the pressed key. Each key has its own scancode listed in Table 4.1. The relationship between the keys and their scancode is not based on ASCII.

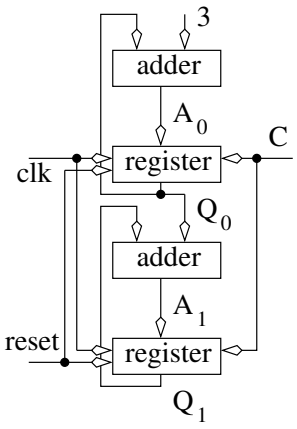
Nomenclature:	scancode classifier
Data Input:	8-bit $D = d_7 \ldots d_0$
Data Output:	IsP, IsL, IsM, IsI, IsS
Control:	none
Status:	none
Behavior:	IsP =1 when D is the scan code for the letter "P". IsL =1 when D is the scan code for the letter "L". IsM =1 when D is the scan code for the letter "M". IsI =1 when D is the scan code for the letter "I". IsS =1 when D is the scan code for the letter "S".

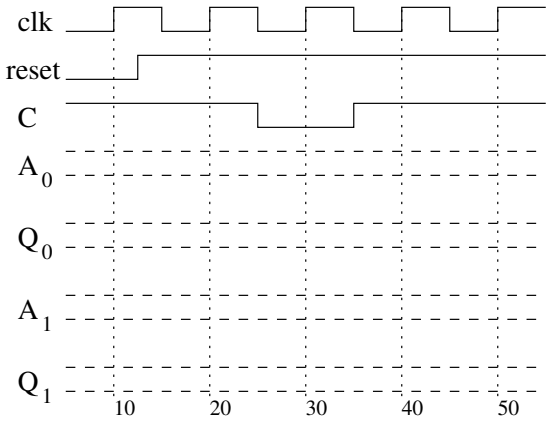
19. **(10 pts.)** Build a box which converts an 8-bit scancode for a hexadecimal digit into a 4-bit hexadecimal values.

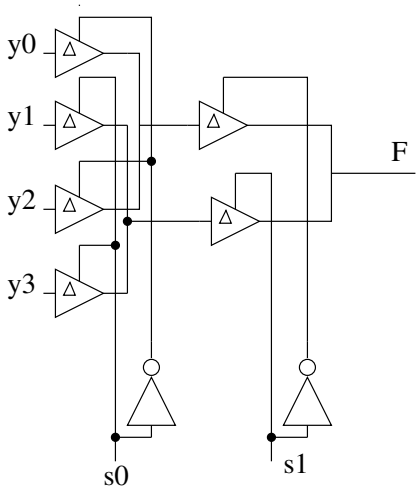
Nomenclature:	scancode classifier
Data Input:	8-bit $D = d_7 \ldots d_0$
Data Output:	4-bit $H = h_3 h_2 h_1 h_0$
Control:	none
Status:	none
Behavior:	Converts the scancode D, representing a the key of a hexadecimal character, into its 4-bit value H.

For example, if $D = 25_{16}$, the scancode for the "4" key, then the converter should output $H = 0100_2$. Assume that the inputs are always legal hexadecimal scancodes.


4:16
4:16
4:16
4:16
4:16
4:16
4:16
4:16
2:4
2:4
1:2
S6
S5S4


3
adder
A0
C
clk
register
Q0
adder
A1
reset
register
Q1


clk
reset
C
A0
Q0
A1
Q1
10
20
30
40
50


y0
y1
y2
y3
F
s0
s1