HW# 4 Solution

Q.1. Consider a technology library containing the following cells:

Cell	Area Cost
INV(x1) = x1'	1
NAND2(x1, x2) = (x1 x2)'	2
NAND3(x1, x2, x3) = (x1 x2 x3)'	3
NOR2(x1, x2) = (x1 + x2)'	2
AOI21(x1, x2, x3) = ((x1 x2) + x3)'	3
OAI21(x1, x2, x3) = ((x1+x2) x3)'	3

(i) Show the **pattern trees** of the library cells using **NAND2** and **INV** as base functions. Assume that symmetric representations do not need to be stored.
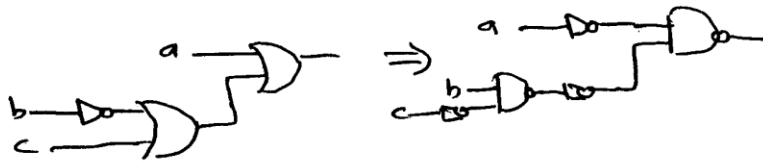
(ii) Decompose the function f = a + b' + c using NAND2 and INV as base functions into all possible **non-symmetric decomposition**s. Then, **map** the decomposed circuits using the given library and determine the decomposition that leads to a **lower area cost**.
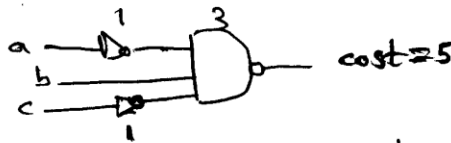
The function $f = a + \bar{b} + c$ can be decomposed into 2-input OR gates by one of the following decompositions:
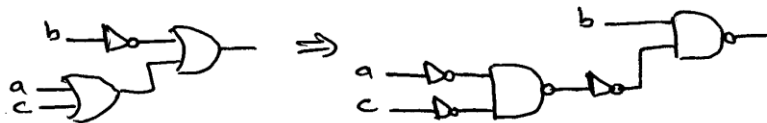

The first and second decompositions have the same structure and will lead to a solution of the same cost. So, we will consider mapping the first and third decompositions.
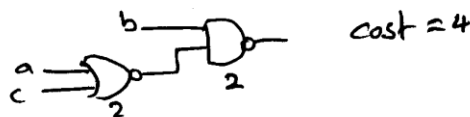

The minimum solution for mapping this decomposition has a cost of 5 as shown below:
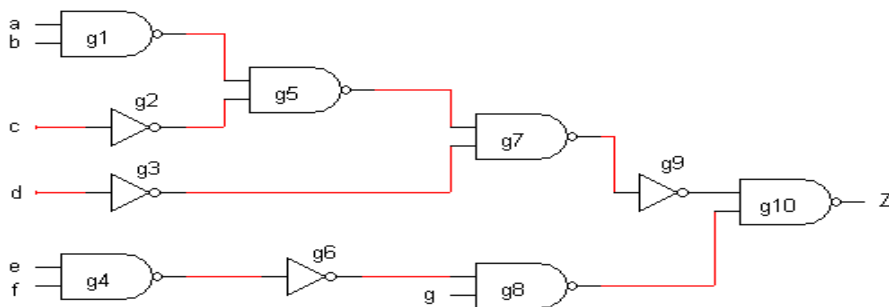

cost = 5

Next, we consider the third decomposition.

The minimum solution for mapping this decomposition has a cost of 4 as shown below:
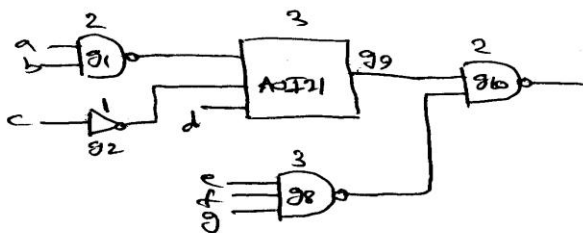

cost = 4

(iii) Using the dynamic programming approach, **map** the circuit given below using the given library into the **minimum area** cost solution. Inputs are *{a, b, c, d, e, f, g}* and output is *{Z}*.

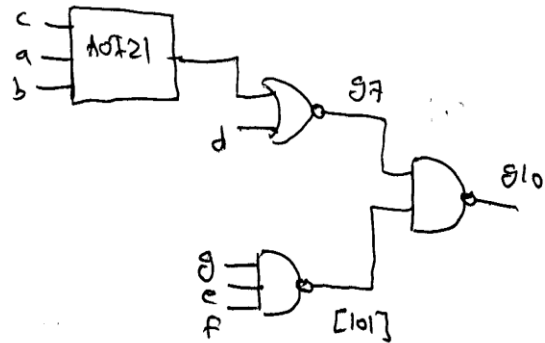| vertex	Match	gate	cost
g_1	t_2	$Nand_2(a,b)$	2
g_2	t_1	$INV(c)$	1
g_3	t_1	$INV(d)$	1
g_4	t_2	$Nand_2(e,f)$	2
g_5	t_2	$Nand_2(g_1,g_2)$	$2+2+1 = 5$
g_6	t_1	$INV(g_4)$	$1+2 = 3$
g_7	t_2	$Nand_2(g_5,g_3)$	$2+5+1 = 8$
g_8	t_2	$Nand_2(g_6,g)$	$2+3 = 5$
	t_3	$Nand_3(e,f,g)$	$\underline{3}$
g_9	t_1	$INV(g_7)$	$1+8 = 9$
	t_5	$AoI_{21}(g_1,g_2,d)$	$3+2+1 = \underline{6}$
g_{10}	t_2	$Nand_2(g_9,g_8)$	$2+6+3 = \underline{11}$
	t_3	$Nand_3(g_5,g_3,g_8)$	$3+5+1+3 = 12$

Thus, the minimum cost is $\underline{11}$ and the mapped solution is shown below:

(iv) Using the given library, use the SIS command *read_libray* **q1.lib** to read the library. Then, map the circuit to the library using the sis command *map –s –m 0*. Compare your solution to the solution obtained in (iii). You can save the mapped circuit using the sis command *write_blif –n*. Why do you think the solution obtained by SIS is better than your solution?

The solution achieved by SIS is given below with a total area of 10.

.model hw4q1_081.blif
.inputs a b c d e f g
.outputs g10
.default_input_arrival 0.00 0.00
.default_output_required 0.00 0.00
.default_input_drive 0.20 0.20
.default_output_load 1.00
.default_max_input_load 999.00
.gate NAND3 a=g b=e c=f O=[101]
.gate AOI21 a=c b=a c=b O=g5
.gate NOR2 a=d b=g5 O=g7
.gate NAND2 a=[101] b=g7 O=g10
.end

The obtained solution is better than the solution we obtained because SIS uses an additional optimization technique by inserting pairs of inverters at each line in the subject graph and then finding an optimal mapping of the subject graph. Any mapped pair of inverters can then be eliminated. Using this technique it is possible to obtain a better mapping of the given network.

(v) Assuming **Boolean matching**, determine the <u>number</u> of ROBDD's that need to be stored in the cell library for each of the following cells. <u>Justify your answer</u>.

a. $f = a b + a c + b c$

Since   a, b, and c are all   symmetric
$C_3 = \{ (a,b,c) \} \Rightarrow$ # ROBDD's = 1

b. $f = a b c + a' b' d$

we   have   2 binate variables $\{a, b\}$
and   2 unate variables $\{c, d\}$
variables   a and b are symmetric.
$\Rightarrow$ # ROBDD's = 2

Q.2. Consider the incompletely-specified FSM that has 6 states, two inputs and one output, represented by the following state table:

Present State	Next State, Output			
	00	**01**	**11**	**10**
S1	S2, 0	–, –	S5, 1	–, –
S2	S1, 0	S3, –	–, –	–, –
S3	S3, –	S1, 1	S5, –	S4, 1
S4	–, –	S2, –	S1, –	–, –
S5	S3, –	S2, 1	–, 0	S6, –
S6	S6, 1	S1, –	S2, –	S5, –

(i) Determine the incompatible and the compatible states along with their implied pairs.

Compatibility Table

The incompatible states are:
(S1, S4), (S1, S5), (S1, S6), (S2, S6), (S3, S4)

The compatible states with their implied pairs:

(S1, S2)
(S1, S3) ⟵ (S2, S3)
(S2, S3) ⟵ (S1, S3)
(S2, S4) ⟵ (S2, S3)
(S2, S5) ⟵ (S1, S3) , (S2, S3)
(S3, S5) ⟵ (S1, S2) , (S4, S6)
(S3, S6) ⟵ (S2, S5) , (S4, S5)
(S4, S5)
(S4, S6) ⟵ (S1, S2)
(S5, S6) ⟵ (S1, S2) , (S3, S6)

(ii) Compute the maximal compatible classes along with their implied state pairs.

Maximal compatible classes:

We assign for every state S_i variable X_i.
From the incompatible state pairs we have

$$(\bar{x_1}+\bar{x_4})(\bar{x_1}+\bar{x_5})(\bar{x_1}+\bar{x_6})(\bar{x_2}+\bar{x_6})(\bar{x_3}+\bar{x_4})$$

$$= (\bar{x_1}+\bar{x_4}\bar{x_5})(\bar{x_1}+\bar{x_6})(\bar{x_2}+\bar{x_6})(\bar{x_3}+\bar{x_4})$$

$$= (\bar{x_1}+\bar{x_4}\bar{x_5}\bar{x_6})(\bar{x_2}+\bar{x_6})(\bar{x_3}+\bar{x_4})$$

$$= (\bar{x_1}\bar{x_2}+\bar{x_1}\bar{x_6}+\bar{x_2}\bar{x_4}\bar{x_5}\bar{x_6}+\bar{x_4}\bar{x_5}\bar{x_6})(\bar{x_3}+\bar{x_4})$$

$$= \bar{x_1}\bar{x_2}\bar{x_3} + \bar{x_1}\bar{x_2}\bar{x_4} + \bar{x_1}\bar{x_3}\bar{x_6} + \bar{x_1}\bar{x_4}\bar{x_6}$$
$$+ \bar{x_3}\bar{x_4}\bar{x_5}\bar{x_6} + \bar{x_4}\bar{x_5}\bar{x_6}$$

Thus, we have 5 maximal compatible classes as follows:

$\bar{x_1}\bar{x_2}\bar{x_3}$: $C1$: (S_4, S_5, S_6) ⟸ (S_1, S_2), (S_3, S_6)

$\bar{x_1}\bar{x_2}\bar{x_4}$: $C2$: (S_3, S_5, S_6) ⟸ $(S_1, S_2), (S_4, S_6), (S_2, S_5), (S_4, S_5)$

$\bar{x_1}\bar{x_3}\bar{x_6}$: $C3$: (S_2, S_4, S_6) ⟸ (S_2, S_3), (S_1, S_3)

$\bar{x_1}\bar{x_4}\bar{x_6}$: $C4$: (S_2, S_3, S_5) ⟸ (S_1, S_3), (S_1, S_2), (S_4, S_6)

$\bar{x_4}\bar{x_5}\bar{x_6}$: $C5$: (S_1, S_2, S_3)

(iii) Reduce the state table into the minimum number of states and show the reduced state table.

If we consider the maximal compatible classes to find a minimum cover, we will get either $C1\ C2\ C3\ C5$ or $C1\ C2\ C4\ C5$ i.e. a minimum of four reduced states.

However, the state table can be reduced into $\underline{3}$ states by using the following cover:

(S_1, S_2, S_3), (S_4, S_5), (S_6)

The reduced state table is:

Present state	Next state, output			
	00	01	11	10
S123	S123, 0	S123, 1	S45, 1	S45, 1
S45	S123, —	S123, 1	S123, 0	S6, —
S6	S6, 1	S1, —	S2, —	S5, —

Q.3. Consider the incompletely-specified FSM that has 4 states, two inputs and two outputs, represented by the following state table:

Product	Input	Present State	Next State	Output
P1	10	S1	S2	11
P2	00	S2	S2	11
P3	01	S2	S2	00
P4	00	S3	S2	00
P5	10	S2	S1	11
P6	10	S3	S1	11
P7	00	S1	S1	--
P8	01	S3	S0	00
P9	11	S1	S1	10
P10	11	S3	S3	01
P11	11	S0	S0	11

(i) Assuming the following constraints, S0 = S1 OR S3, and that the code of S2 is covered by all other state codes, the state table can be reduced into the table shown below. Using implicant merging, covering and disjunctive relations show step by step how you can obtain the reduced state stable given below.

Input	Present State	Next State	Output
–0	S1, S2	S2	11
10	S2, S3	S1	11
00	S1	S1	--
01	S3	S0	00
11	S0, S1	S1	10
11	S0, S3	S3	01

- P5 and P6 can be merged using implicant merging into the following row :

  lo   S2, S3   SI   ll

- P1 can be changed to :

  – 0   S1   S2   ll

  if we impose the covering constraint S1 ⊇ S2

- P2 can be changed to :

  – 0   S2   S2   ll

  if we impose the covering constraint S1 ⊇ S2

- These two rows can now be merged to :

  – 0   S1, S2   S2   ll

- $P3$, can be changed to:

		S2	00
0-	S2		

- $P4$ can be changed to:

		S2	00
0-	S3		

if we impose the constraint that $S0 \geq S2$

- Now $P3$ and $P4$ can be merged into:

		S2	00
0-	S2,S3		

- If we impose the disjunctive constraint that $S0 = S1$ or $S3$, then $P11$ can be removed and $P9$ and $P10$ changed to:

$P9$	11	S0,S1	S1	10
$P10$	11	S0,S3	S3	01

- Finally, the following row can be eliminated if it is assumed that the code of $S2$ is covered by all other codes i.e. the code of $S2$ is the all-0 code:

0-	S2,S3	S2	00

(ii) Show the encoding constraint matrix and compute all the seed dichotomies of the encoding constraint matrix. Then, eliminate seed dichotomies that violate the given covering and disjunctive constraints.

Encoding constraint Matrix:

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$
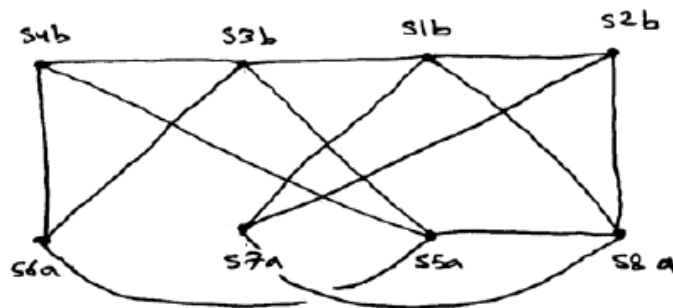
Seed Dichotomies:

$S1a$: $(S1, S2)$, $(S0)$ ✗
$S1b$: $(S0)$, $(S1, S2)$

$S3a$: $(S2, S3)$, $(S0)$ ✗
$S3b$: $(S0)$, $(S2, S3)$

$S5a$: $(S0, S1)$, $(S2)$
$S5b$: $(S2)$, $(S0, S1)$ ✗

$S7a$: $(S0, S3)$, $(S1)$
$S7b$: $(S1)$, $(S0, S3)$ ✗

$S2a$: $(S1, S2)$, $(S3)$ ✗
$S2b$: $(S3)$, $(S1, S2)$

$S4a$: $(S2, S3)$, $(S1)$ ✗
$S4b$: $(S1)$, $(S2, S3)$

$S6a$: $(S0, S1)$, $(S3)$
$S6b$: $(S3)$, $(S0, S1)$ ✗

$S8a$: $(S0, S3)$, $(S2)$
$S8b$: $(S2)$, $(S0, S3)$ ✗

(iii) Compute all the prime dichotomies and eliminate those that violate the disjunctive constraints.

Prime Dichotomies:

P1 : (S1b, S2b, S7a, S8a) : (S0,S3), (S1,S2)

P2 : (S3b, S4b, S5a, S6a) : (S0,S1), (S2,S3)

P3 : (S5a, S8a) : (S0, S1, S3), (S2)

P4 : (S1b, S3b) : (S0), (S1,S2,S3)

Note that P4 violates the disjunctive constraint and has to be discarded. Also, P3 violates the disjunctive constraint if we assume 2-bits encoding.

(iv) Find a state encoding satisfying the given constraints. Verify that your encoding satisfies all the constraints.

A minimum cover is P1 and P2 which results in the following state encoding

State	code	
S0	1	1
S1	0	1
S2	0	0
S3	1	0

Note that the disjunctive constraint $S_0 = S_1$ OR S_3 is satisfied since $11 = 01$ OR 10.

Also, the code of $S_2 = 00$, which is covered by all other codes.

Also, $S_1 \cdot S_2 = 0-$, which does not intersect with other codes.

$S_2 \cdot S_3 = -0$, which does not intersect with other codes.

$S_0 \cdot S_1 = -1$, which does not intersect with other codes.

$S_0 \cdot S_3 = 1-$, which does not intersect with other codes.

(v) Using K-MAP, obtain the equations for the output and flip-flops. Compare your solution to the solution obtained by running the SIS command **stg_to_network** using the state codes obtained in (iii).

$$D_1 = I_0 F_1$$

$$D_0 = \overline{I_1} F_0 + I_0 F_0 + \overline{I_1} I_0 F_1 + I_1 \overline{I_0} \overline{F_0}$$

$$O_1 = I_1 \overline{I_0} + \overline{I_0} \overline{F_1} + F_0$$

$$O_0 = \overline{I_0} \overline{F_1} + I_1 F_1$$

\# of literals = 21 literals

The circuit resulting based on the stg_to_network command is as follows:

```
sis> read_blif hw4q3_081.blif
sis> stg_to_network
sis> print
    [0] = IN_0 IN_1 LatchOut_v2 + IN_0' IN_1 LatchOut_v2
    [1] = IN_0 IN_1' LatchOut_v3' + IN_0' IN_1 LatchOut_v2 + IN_0' LatchOut_v3+ IN_1
LatchOut_v3
    {OUT_0} = IN_0 IN_1' LatchOut_v3' + IN_1 LatchOut_v3 + IN_1' LatchOut_v2'
    {OUT_1} = IN_0 IN_1 LatchOut_v2 + IN_0 IN_1' LatchOut_v3' + IN_1' LatchOut_v2'
sis> print_stats
hw4q3_081    pi= 2  po= 2  nodes= 4    latches= 2
lits(sop)= 31  #states(STG)=  4
```

Note that the number of literals is more than what we have obtained using K-map because the objective here is to minimize the number of products and not the number of literals. To optimize the number of literals, we need to do single output optimization using the command stg_to_network –e 2 which produces the following circuit:

```
sis> read_blif hw4q3_081.blif
sis> stg_to_network -e 2
sis> print
    [0] = IN_1 LatchOut_v2
    [1] = IN_0 IN_1' LatchOut_v3' + IN_0' IN_1 LatchOut_v2 + IN_0' LatchOut_v3
+ IN_1 LatchOut_v3
    {OUT_0} = IN_0 IN_1' + IN_1' LatchOut_v2' + LatchOut_v3
    {OUT_1} = IN_0 LatchOut_v2 + IN_1' LatchOut_v2'
sis> print_stats
hw4q2_081    pi= 2  po= 2  nodes= 4    latches= 2
lits(sop)= 21  #states(STG)=  4
```
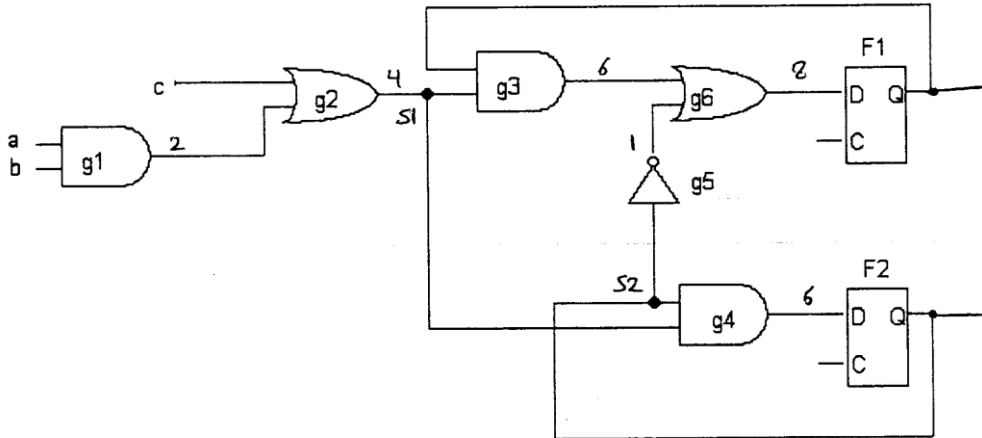
(vi) Perform state assignment using the program nova by running the SIS command *state_assign nova*. Compare the obtained solution to your solution in (v) in terms of number of literals.

The result produced is as follows which is worse that the solution obtained in (v).

```
sis> read_blif hw4q3_081.blif
sis> state_assign nova
Running nova, written by Tiziano Villa,  UC Berkeley
sis> print
    {OUT_0} = IN_0 IN_1' LatchOut_v2' + IN_1 LatchOut_v2 LatchOut_v3 + IN_1' LatchOut_v2 +
IN_1' LatchOut_v3' + LatchOut_v2 LatchOut_v3'
    {OUT_1} = IN_0 LatchOut_v2' + IN_1' LatchOut_v2 + IN_1' LatchOut_v3' + LatchOut_v2
LatchOut_v3'
    v4.0 = IN_0 IN_1' LatchOut_v2' + IN_0' IN_1 LatchOut_v3 + IN_0' LatchOut_v2+ IN_1
LatchOut_v2 LatchOut_v3 + LatchOut_v2 LatchOut_v3'
    v4.1 = IN_0 LatchOut_v2' + IN_0' LatchOut_v2 + IN_1 LatchOut_v2 LatchOut_v3
sis> print_stats
hw4q3_081    pi= 2  po= 2  nodes= 4    latches= 2
lits(sop)= 40  #states(STG)=  4
```

The state assignment generated by nova is: {S0 = 10, S1 = 11, S2 = 00, S3 = 01}.

Q.4. Consider the following circuit with inputs {a, b, c} and outputs {F1, F2}. Assume that the delay of an Inverter is 1 unit delay, the delay of a 2-input AND gate is 2 unit delays, and the delay of a 2-input OR gate is 2 unit delays. Consider the circuit given below:

(i) Determine the critical path of this circuit and the maximum propagation delay.
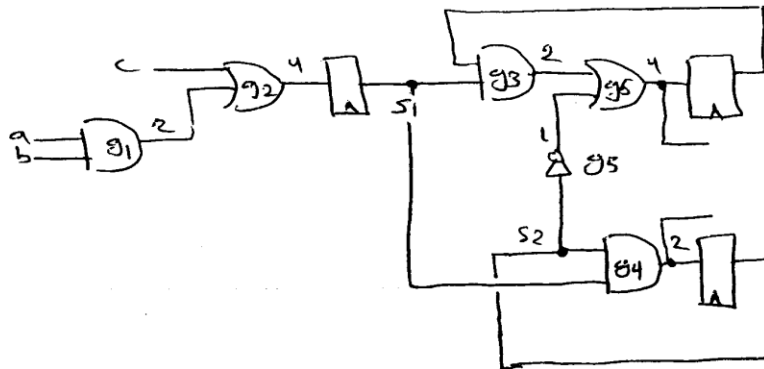
The critical path is the path {g1, g2, g3, g6} which is 8 delay units

(ii) Using only the **Retiming** transformation, reduce the critical path of this circuit with the minimum number of flip-flops possible.

The critical delay can be reduced to 4 with an extra flip-flop as follows:

- Retime g6 by +1
- Retime g4 by +1
- Retime g3 by +1
- Retime g5 by +1
- Retime S1 by +1
- Retime S2 by +1

The resulting circuit after retiming is :

It is clear that the critical path of the retimed circuit is 4 delay units.

(iii) Read the library **q4.lib** using the command ***read_library q4.lib***. Then, map your design to the library using the command ***map –s***. Then, retime the circuit using the command ***retime -i***. Compare the maximum arrival time before and after retiming. Compare the obtained solution to the solution you obtained in (ii).

The result of applying the retiming transformation is given below:

final cycle delay   = 4.00
final number of registers = 4
final logic cost   = 11.00
final register cost   = 64.00
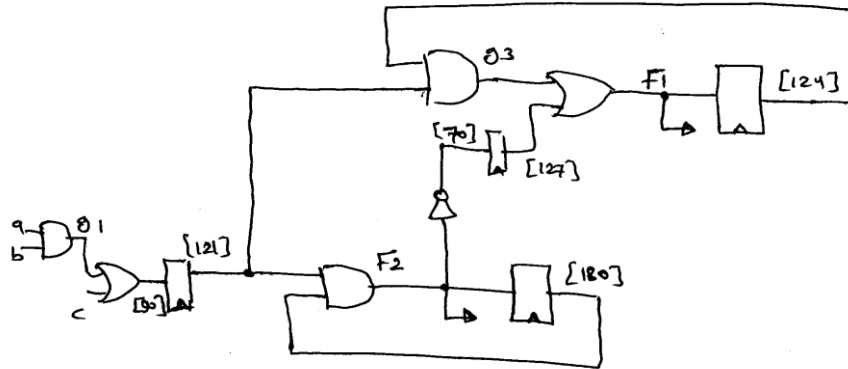
RETIME: Final cycle time achieved = 4.00

The resulting circuit is:
.model hw4q4_081.blif
.inputs a b c
.outputs f1 f2
.default_input_arrival 0.00 0.00
.default_output_required 0.00 0.00
.default_input_drive 0.00 0.00
.default_output_load 1.00
.default_max_input_load 999.00
.mlatch dff_re D=[90] Q=[121] NIL 3
.mlatch dff_re D=f1 Q=[124] NIL 3

.mlatch dff_re D=[70] Q=[127] NIL 3
.mlatch dff_re D=f2 Q=[130] NIL 3
.gate and2 1A=a 1B=b O2=g1
.gate or2 1A=c 1B=g1 O1=[90]
.gate and2 1A=[121] 1B=[124] O2=g3
.gate inv 1A=f2 O=[70]
.gate or2 1A=g3 1B=[127] O1=f1
.gate and2 1A=[121] 1B=[130] O2=f2
.end

Note that the number of registers obtained by SIS is different due to our description of the circuit having the fanout connecting to G5 not coming directly from F2. This is why we were able to minimize the number of registers more than what the tool could have done. If we have connected F2 to a buffer representing S2 and then let S2 fanout, SIS will obtain the same solution we have obtained.