

Exam 3 Review

{ Arithmetic Op.
Look ahead Addition
Carry Save Addition
Floating Op.

Standard Cell

Combinational:

✱ Decoder / Encoder

✱ Mux / DeMux

Sequential

✱ Reg. / Counter

✱ Sequential Adder

Mult.

System Design

Standard Modules

Memory components

✱ Interconnect

✱ Handshaking Protocol.

{ Data Subsystem
Control Subsystem.

1. Introduction
2. Background
3. Methodology
4. Results
5. Discussion
6. Conclusion

1. Introduction
2. Background
3. Methodology
4. Results
5. Discussion
6. Conclusion

1. Introduction
2. Background
3. Methodology
4. Results
5. Discussion
6. Conclusion

CSE 140 Spring 2017: Final, Due 10PM 6/15/2017

This is an open book final. The exam contains **four problems**. Web searches are encouraged. If there is any uncertainty about the problems, make and state your assumptions. Each person is supposed to work by oneself. Thus, no discussion is allowed.

1. (Boolean Algebra) Prove the following Boolean theorem using Boolean laws only, i.e. no theorem is allowed for the proof. State the name of the law for each step of your proof.

$$x + 1 = 1 \tag{1}$$

2. (Switching Function) A set of switching variables a, b, c, d, e has the values defined by the following simultaneous switching equations.

$$a + b + c = 1 \tag{2}$$

$$b' + c' + d = 1 \tag{3}$$

$$c + d' + e = 1 \tag{4}$$

$$a + c + e' = 1 \tag{5}$$

$$a' + b' + d = 1 \tag{6}$$

Use a switching function $f(a, b, c, d, e)$ to cover the values of the variables as its On-Set F . In other words, a min-term m_i belongs to F if and only if the corresponding binary code $(a, b, c, d, e) = i$ satisfies the above equations.

2.1 Check which of the following min-terms belong to On-Set F .

$$m_0, m_5, m_{10}, m_{15}, m_{30}. \tag{7}$$

2.2 Express function $f(a, b, c, d, e)$ in a minimal sum of products form. Describe the number of product terms and the number of literals.

3. (Retiming) Check the digital correlator circuit in paper, C. Leiserson and J. Saxe, "Retiming Synchronous Circuitry," *Algorithmica*, pp. 6:5-35, 1991. (The paper is posted with the exam. You are encouraged to read the approaches and references of the paper, but it is not required.) Suppose that each comparator δ propagation delay is changed to 10 picosecond and each adder $+$ propagation delay is changed to 50 picosecond (changed from the numbers in the paper).

3.1 For the digital correlator circuit in figure 1 of the paper, what is the clock period of the circuit (the delay of a longest path of combinational rippling)?

3.2 For the circuit in figure 2 of the paper, what is the clock period of the circuit (the delay of a longest path of combinational rippling)?

3.3 For the same path between figure 1 and figure 2, that starts from and loops back to the Host, the loop traverses the same number of registers. Explain that the host

should experience the same responses from the two circuits in figures 1 and 2. State your explanation in no more than four sentences.

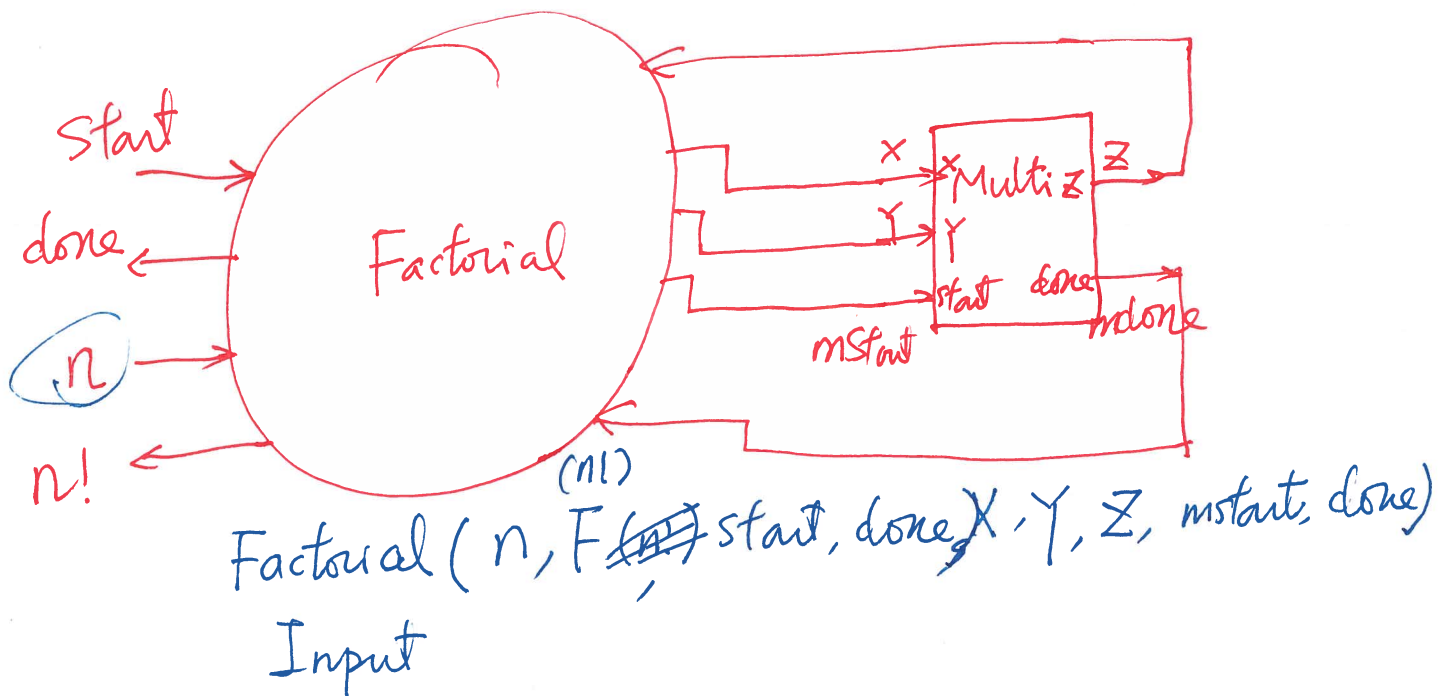
3.4 With aggressive retiming, what is the minimal clock period that can be achieved for the digital correlator in this paper? Hint: check figures 3 and 4 of the paper.

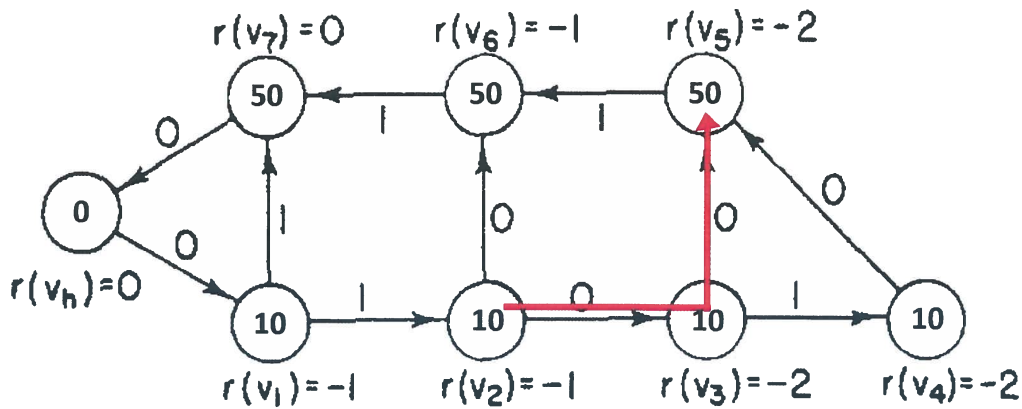
4. (System Designs) Follow the style of the routine *Multiple* in slide 5 of lecture 14 to describe your design. Suppose that module *Multiple*($X, Y, Z, start, done$) is used for the multiplication operation. Design a system that inputs a 3-bit binary number n and outputs its factorial $n!$.

4.1. Write the program. Explain the interface with module *Multiple*($X, Y, Z, start, done$).

4.2. Describe the data subsystem with a schematic diagram. You can treat module *Multiple*($X, Y, Z, start, done$) as a black box.

4.3. Draw the state diagram of the control subsystem and describe the control signals using a truth table.





Minimal Clock Period = 2 Comparator Propagation Delay + 1 Adder Propagation Delay = $(2 * 10) + 50 = 20 + 50 = 70$ ps.

Rubrics:

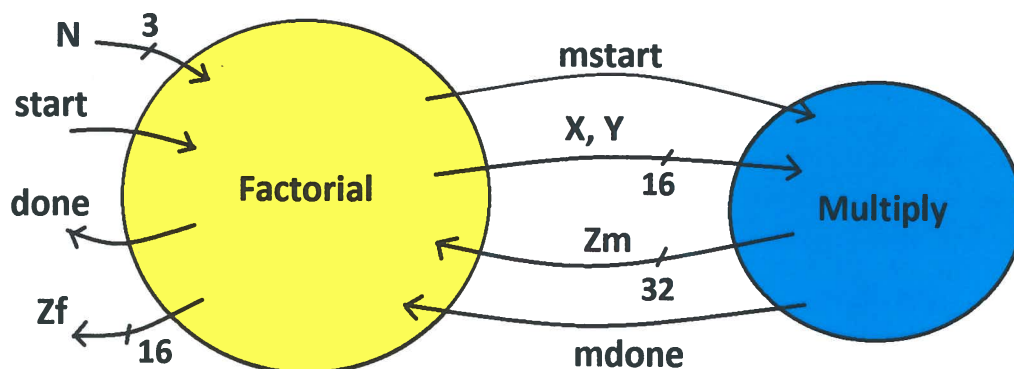
3 points, if correct answer. No partial credit.

4. (System Designs) Follow the style of the routine *Multiply* in slide 5 of lecture 14. Suppose that module *Multiply*(*X*, *Y*, *Z*, *start*, *done*) is used for the multiplication operation. Design a system that inputs a 3-bit binary number *n* and outputs its factorial *n!*.

4.1. Write the program. Explain the interface with module *Multiply*(*X*, *Y*, *Z*, *start*, *done*).

Solution:

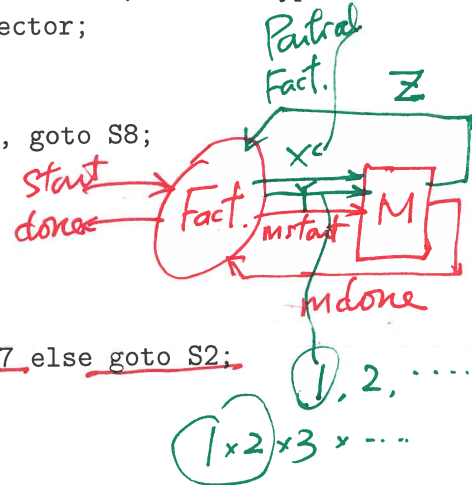
Interface:



Program: (n!)

```
Factorial(N, Zf, start, done, X, Y, Zm, mstart, mdone)
{
  Input: N[2:0] type bit-vector, start type boolean, Zm[31:0] type bit-vector,
  mdone type boolean;
  Local-Object: F[15:0] type bit-vector, i[15:0] type bit-vector;
  Output: Zf[15:0] type bit-vector, done type boolean, mstart type boolean,
  X[15:0] type bit-vector, Y[15:0] type bit-vector;
```

```
S0: if start' goto S0 || done = 1;
S1: F = 1 || i = 1 || done = 0 || if N == 0, goto S8;
S2: mstart = 0 || if mdone', goto S2;
S3: if i == 1, goto S5
S4: F = Zm[15:0]
S5: if i >= N, goto S8;
S6: i = i + 1 || X <= F;
S7: Y <= i || mstart = 1 || if mdone, goto S7 else goto S2;
S8: Zf = F || done = 1 || goto S0;
}
```



The module Multiply is executed every time we want to perform a multiplication when computing the factorial. F and i are connected to X and Y inputs of Multiply. mstart is connected to the input start of Multiply. The output Z of Multiply is connected to Zm. The output done of Multiply is connected to mdone. Initially mstart is set to 0, and we wait until mdone becomes 1. Then X and Y, are updated with the new values of F and i, and mstart is set to 1. We then wait for mdone to become 0, when this happens we know that Multiply has started execution. We then set mstart to 0 and wait for mdone to be 1. When this happens we know that Multiply has finished execution and that the multiplication result has been reflected in the input Zm, which we load to F. We then continue our program to compute the factorial.

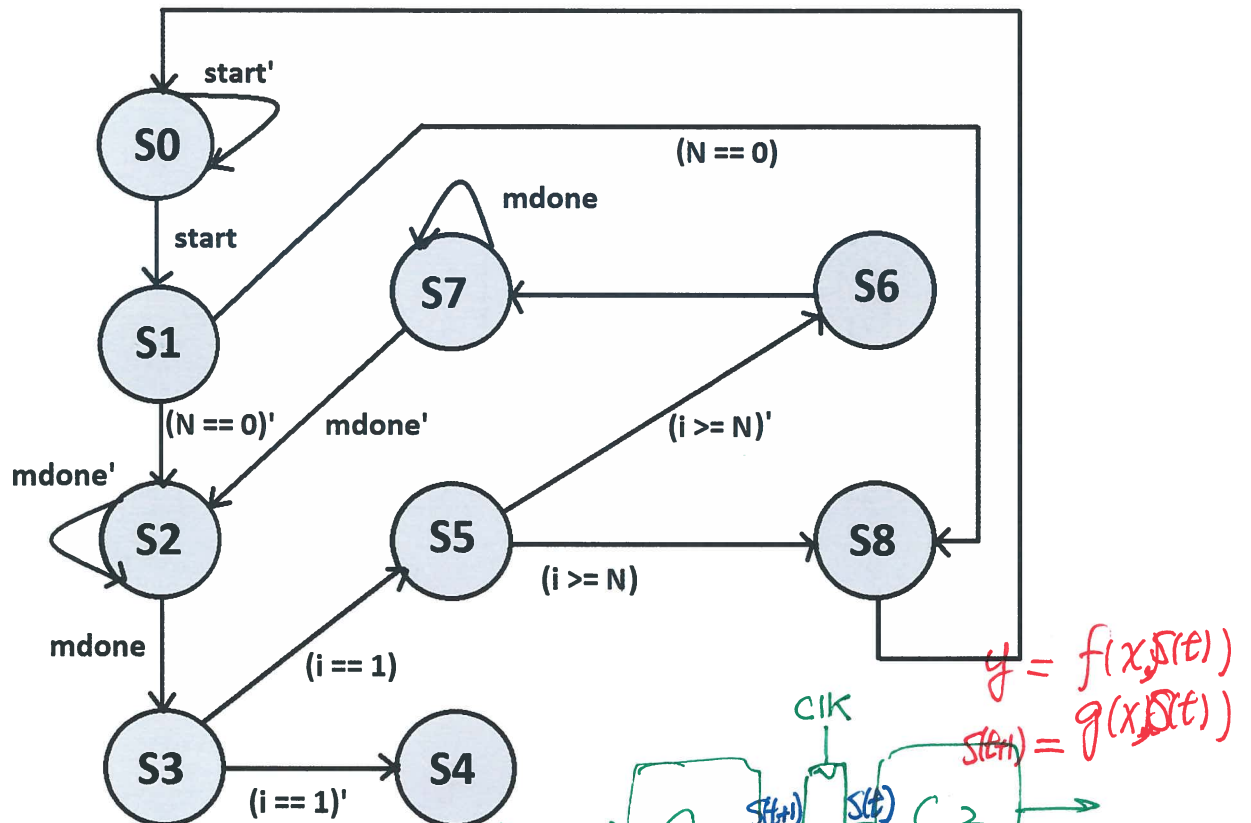
Rubrics:

- 6 points**, if the program is correct with explanation for the interface.
- Subtract 1 point, if the interface explanation is missing.
- Subtract 1 point, if there is a logical mistake in factorial calculation.
- Subtract 0.5 points, if the check for mdone == 1 is missing before calling Multiply.
- Subtract 0.5 points, if mdone == 0 is not checked and mstart = 0 is not set.
- Subtract 0.5 points, if inputs, local objects and outputs are missing or incorrect.
- Use the assumptions mentioned by students to evaluate the above conditions.
- 0 points for no attempt.

4.2. Describe the data subsystem with a schematic diagram. You can treat module *Multiply*(X, Y, Z, start, done) as a black box.

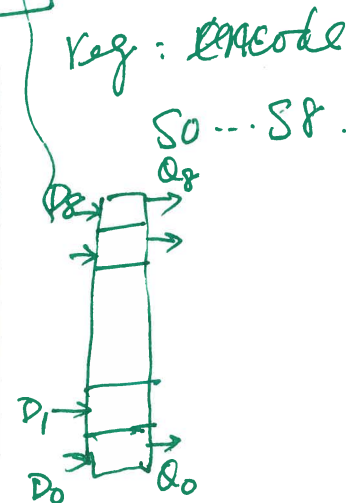
Solution:

State Diagram:

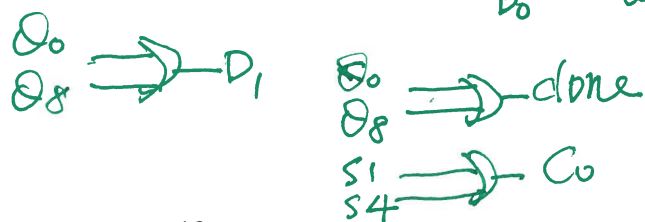


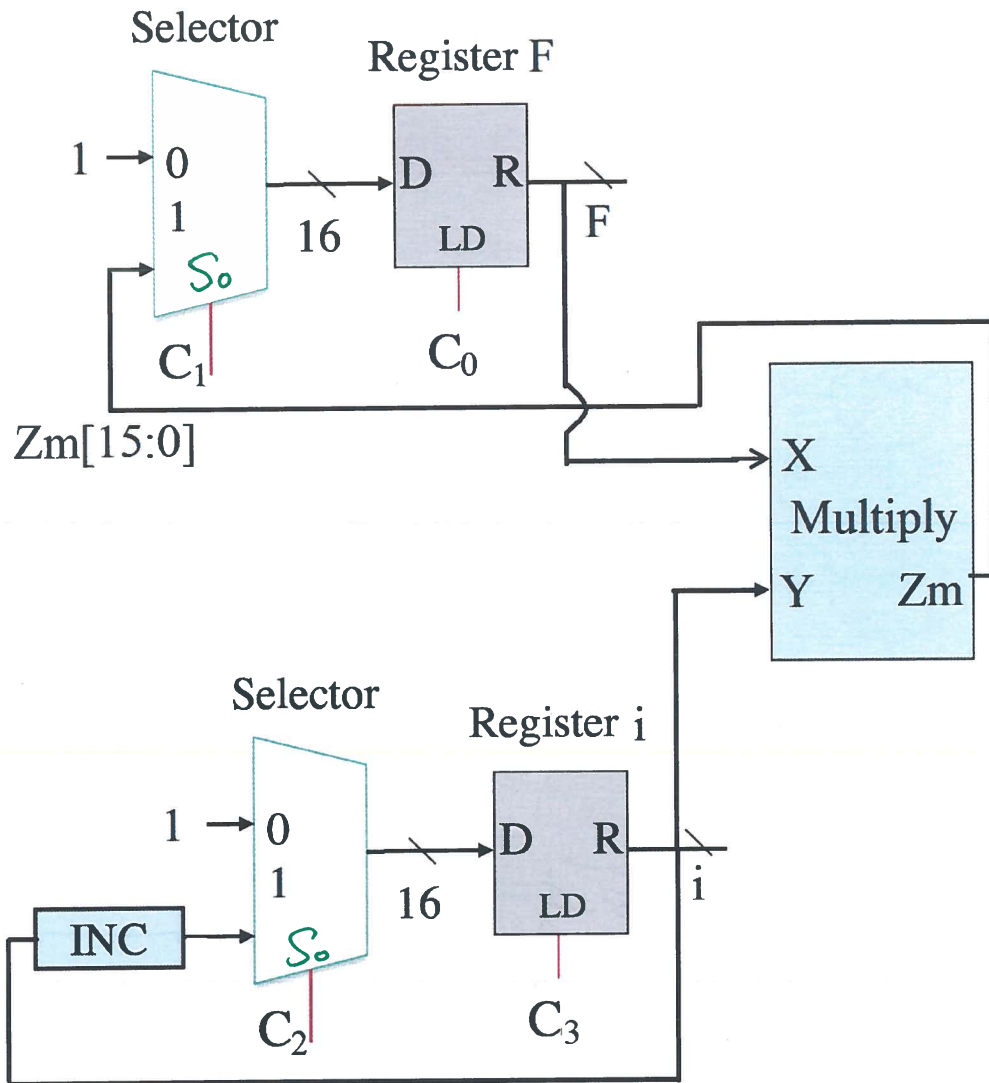
Control Signals:

S	C ₀	C ₁	C ₂	C ₃	done	mstart
S0	0	X	X	0	1	X
S1	1	0	0	1	0	X
S2	0	X	X	0	0	0
S3	0	X	X	0	0	0
S4	1	1	X	0	0	0
S5	0	X	X	0	0	0
S6	0	X	1	1	0	0
S7	0	X	X	0	0	1
S8	0	X	X	0	1	0



Rubrics:





Rubrics:

5 points, if the schematic diagram for the data subsystem is correct.

Subtract 1 point for every missing/unnecessary register or selector or control signal or operator like INC or wire.

Subtract 1 point if there is a missing label on selectors, registers or Multiply.

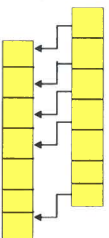
Subtract 2 points if missing the Multiply module.

Please refer to the program of the student when checking above.

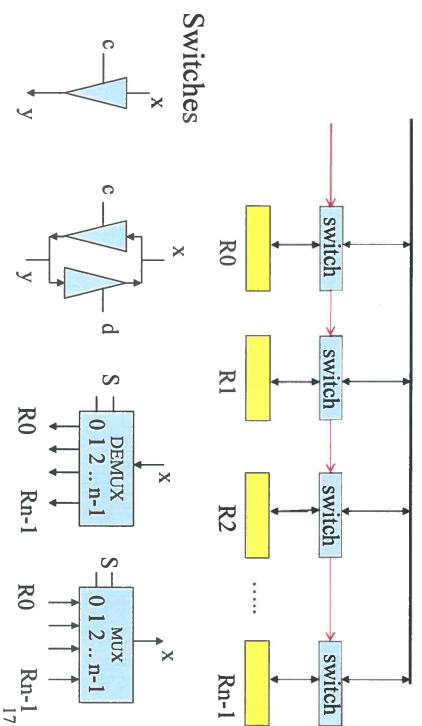
4.3. Draw the state diagram of the control subsystem and describe the control signals using a truth table.

Solution:

2. Band of Wires (BUS)

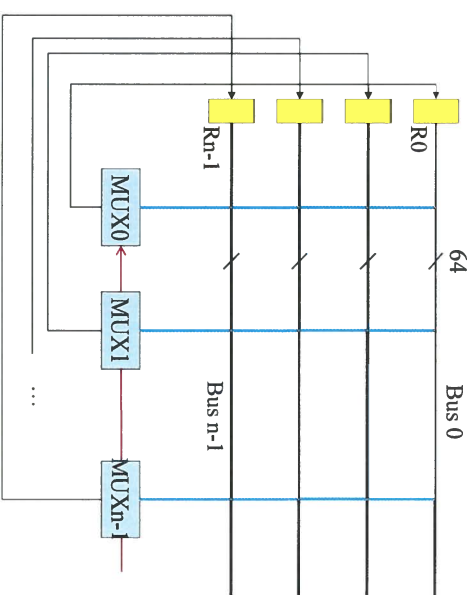


3. Shared Bus

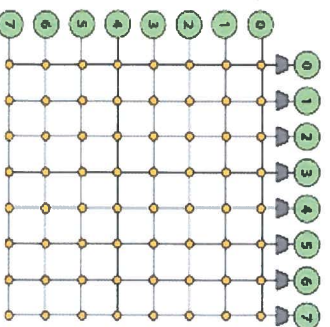


4. Crossbar (Multiple buses running horizontally)

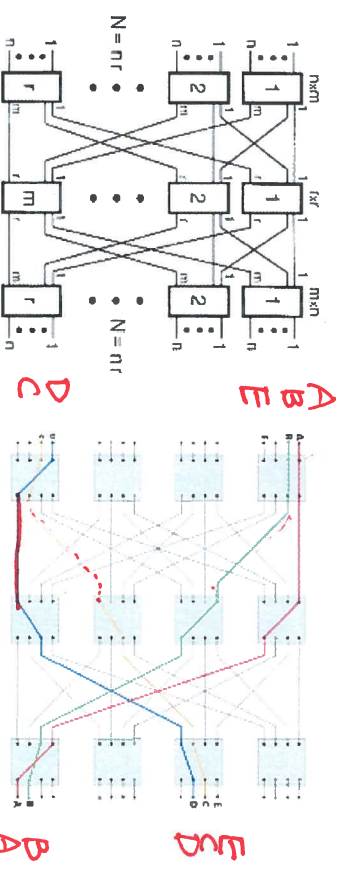
m simultaneous transfers are possible, but more expensive.



Crossbar Switches: VLSI Design

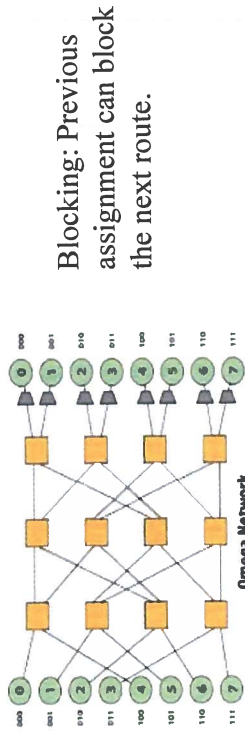
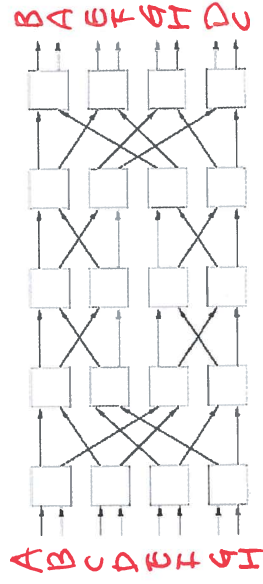


Multistage Network: Clos Network



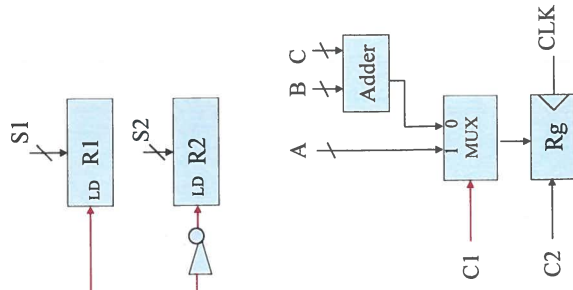
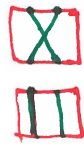
blocking the previous rearrange ports

Multistage Networks: Benes and Omega Networks



Blocking: Previous assignment can block the next route.

21



Ex. If C then $R1 \leftarrow S1$
Else $R2 \leftarrow S2$
Endif;

If C1 then $X \leftarrow A$
Else $X \leftarrow B + C$
Endif
If C2 then $Rg \leftarrow X$
Endif

23

III. Specification: Program

1. Objects (Registers, Outputs of combinational logic)
2. Operation (Logic, Add, Multiplication, DSP, and etc.)
3. Assignment
4. Sequencing

Example:

Signal S1, S2, R[15:0]: FFs, Registers, wires
 $Z \leftarrow A + B$: Registers, Adder, Interconnect
 $R1 \leftarrow R2$: Registers and Interconnect
 Begin, End: Control
 if () then (), ENDIF: Control

22

System Designs

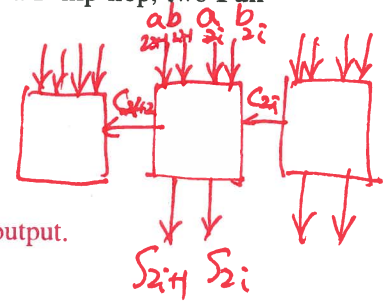
- Introduction
– Methodology and Framework
- Components
- Specification
- Implementation

24

III. A sequential adder inputs (a_{2i}, b_{2i}) and (a_{2i+1}, b_{2i+1}) , the $2i$ 'th and $(2i+1)$ 'th bits of two binary numbers in each clock cycle for $i = 0$ to $\frac{n}{2} - 1$ and outputs the sum s_{2i}, s_{2i+1} , where n is the number of bits and is assumed to be even. Implement the adder with a D flip-flop, two Full Adders and a minimal network of AND, OR, NOT gates. (15 points)

1. State table (8 points)

- -1: two-bit serial adders without carry out $2i+2$.
- -2: incomplete two-bit serial adders with carries
- -4: two-bit serial adders without carry in or write carry in as output.
- -6: only state table for one-bit serial adder.



2. Diagram (7 points)

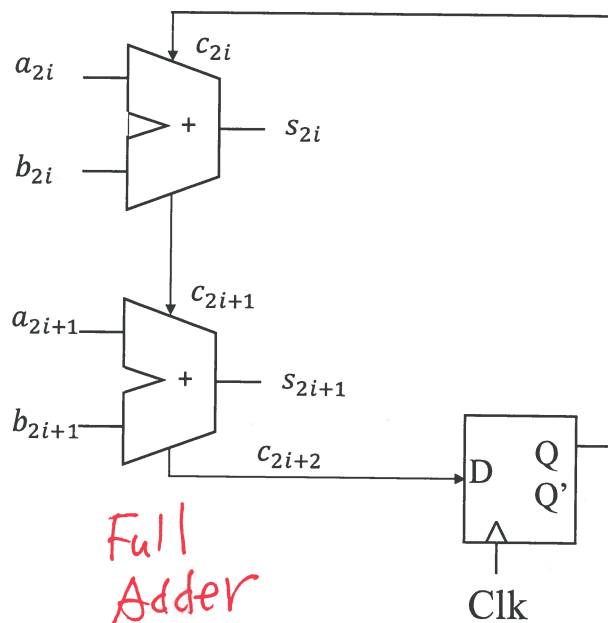
- -2: each mistake.
- -4: wrong diagram with one D F-F and two full adders.

III.1. Write the state table to describe the next states and outputs as functions of present state and inputs. (8 points)

Answer: Input $a_{2i+1:2i}$ and $b_{2i+1:2i}$, output sum $s_{2i+1:2i}$, the carry in c_{2i} to digit $2i$ and carry out c_{2i+2} from digit $(2i+1)$ are connected by the D F-F,

$$D = c_{2i+2}, \quad Q = c_{2i}.$$

III.2. Draw the schematic diagram. (7 points)



$a_{2i+1:2i}$	$b_{2i+1:2i}$	c_{2i}	c_{2i+1}	c_{2i+2}	$s_{2i+1:2i}$
00	00	0	0	0	00
00	00	1	0	0	01
00	01	0	0	0	01
00	01	1	1	0	10
00	10	0	0	0	10
00	10	1	0	0	11
00	11	0	0	0	11
00	11	1	1	1	00
01	00	0	0	0	01
01	00	1	1	0	10
01	01	0	1	0	10
01	01	1	1	0	11
01	10	0	0	0	11
01	10	1	1	1	00
01	11	0	1	1	00
01	11	1	1	1	01
10	00	0	0	0	10
10	00	1	0	0	11
10	01	0	0	0	11
10	01	1	1	1	00
10	10	0	0	1	00
10	10	1	0	1	01
10	11	0	0	1	01
10	11	1	1	1	10
11	00	0	0	0	11
11	00	1	1	1	00
11	01	0	1	1	00
11	01	1	1	1	01
11	10	0	0	1	01
11	10	1	1	1	10
11	11	0	1	1	10
11	11	1	1	1	11

IV. Counter: Given a modulo 8 counter, construct a counter that counts in a repeated sequence (1,2,5,6,7,0) with minimal networks of AND, OR gates. Assume that NOT gates are free. (15 points)

1. Truth table (8 points)

- -1: for each wrong/missing entry.

2. Diagram (7 points)

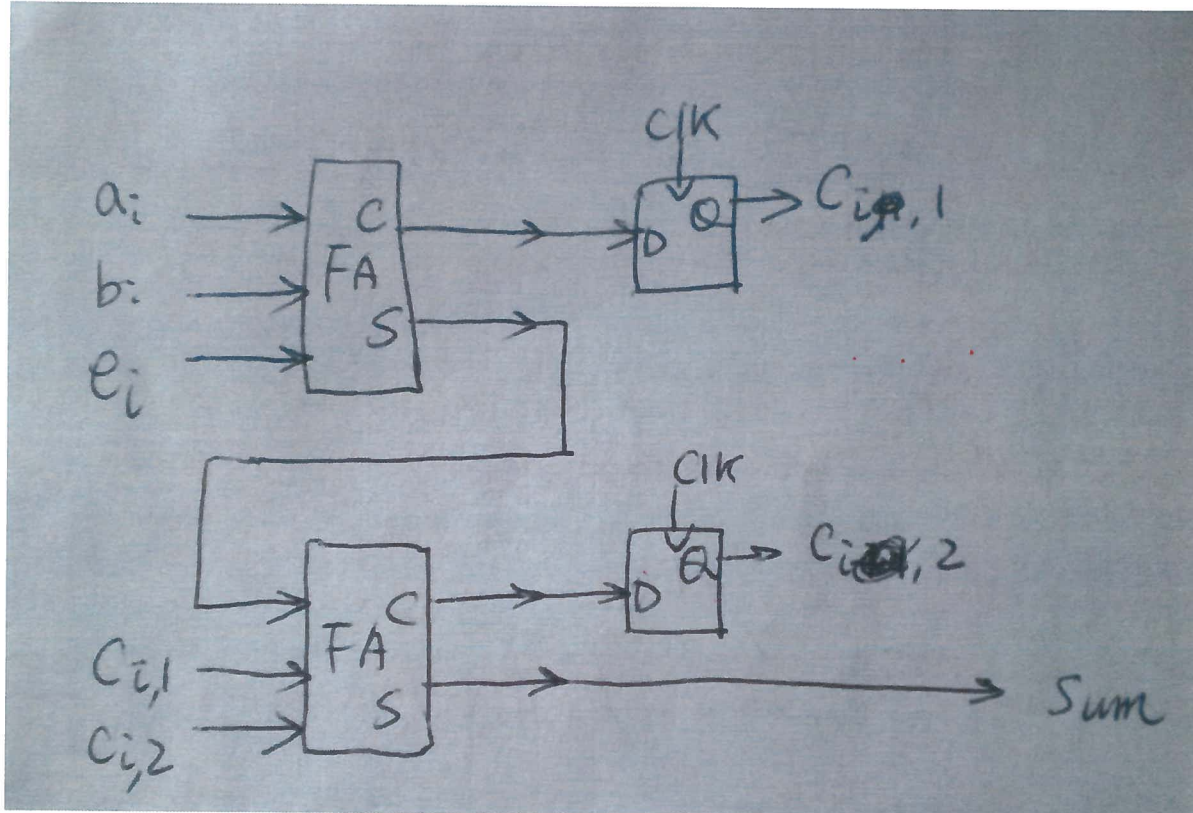
- -1: missing labels (CLK, CNT, LD, D and Q).
- -1: not using minimal networks for LD and D.
- -2: wrong LD or D.
- -1: wrong CNT.

II. Serial Adders:

II.1. A sequential adder inputs a_i , b_i , the i 'th bit of two binary numbers in each clock cycle for $i=0$ to $n-1$ and outputs the sum s_i . Implement the adder with a T flip-flop, and a minimal AND-OR-NOT network (if the network is needed). Write the state table to describe the next states and outputs. Draw the schematic diagram.

Class Notes

II.2. A sequential three-at-a-time adder inputs a_i , b_i , e_i , the i 'th bit of three binary numbers in each clock cycle for $i=0$ to $n-1$ and produces the sum s_i . Implement the adder with a minimal number of Full Adders and two D flip-flops. Other logic components are not needed, nor available. Draw the logic diagram



General Statement: In this exam, we assume that every input comes with its complement.

I. Adders/Subtractors: A Half Subtractor inputs two bits x and subtrahend y , and outputs the difference d and borrow-out $bout$, while a Full Subtractor inputs three bits x , subtrahend y and borrow-in bin , and outputs the difference d and borrow-out $bout$.

I.1. Write the truth tables of the Half Subtractor and the Full Subtractor. Describe those logic using minimal sum of products expressions.

HS

x	y	d	$bout$
0	0	0	0
1	0	1	1
2	1	1	0
3	1	0	0

$d = x \oplus y$
 $bout = xy'$

FS

x	y	bin	d	$bout$
0	0	0	0	0
1	0	0	1	1
2	0	1	1	1
3	0	1	0	1
4	1	0	1	0
5	1	0	0	0
6	1	1	0	0
7	1	1	1	1

$d = x \oplus y \oplus bin$
 $bout = x'bin + x'y + ybin$

I.2. Implement a Full Subtractor with only one Half Subtractor and a minimal two level NOR gates. Draw the logic diagram.

① Use HS for $x \oplus y$, rewrite truth table

x	y	bin	d_{HS}	b_{HS}	d	$bout$
0	0	0	0	0	0	0
1	0	0	1	0	1	1
2	0	1	1	1	1	1
3	0	1	1	0	0	1
4	1	0	1	0	1	0
5	1	0	0	0	0	0
6	1	1	0	0	0	0
7	1	1	0	1	1	1

② Derive $d, bout$ as functions of bin, d_{HS}, b_{HS}

③ From K-map. $d = d_{HS} \oplus bin$
 $bout = b_{HS} + bin d_{HS}$