(Digital Circuit Design and Vulnerabilities)

Contents

Design Vulnerabilities (Combinational Circuit): Example

- Let us consider a 3-bit input encoder that assigns a 2-bit output code (as input for the next module) to each of the three different inputs as shown in the table below.

x	y	z	a	b
0	0	1	0	1
0	1	0	1	0
1	0	0	1	1

- The 3-bit input encoder is defined for only three of the given input combinations, i.e., it is incompletely specified.

- This gives the attacker an opportunity to utilize the unspecified states for backdoors and fault injection-based attacks.

Design Vulnerabilities (Combinational Circuit): Example

- Functional design of the encoder can be obtained as follows:

a	y'z'(00)	y'z(01)	yz(11)	yz'(10)
x'(0)	0	0	0	1
x(1)	1	0	0	0

b	y'z'(00)	y'z(01)	yz(11)	yz'(10)
x'(0)	0	1	0	0
x(1)	1	0	0	0

a= x'yz' + xy'z' and ***b= x'y'z + xy'z'***

x	y	z	a	b
0	0	1	0	1
0	1	0	1	0
1	0	0	1	1

- This design can optimize to reduce the implementation cost and improve the design performance.

Design Vulnerabilities (Combinational Circuit): Example

- Optimal design or simplified design for the encoder:

a	y'z'(00)	y'z(01)	yz(11)	yz'(10)
x'(0)	x	0	x	1
x(1)	1	x	x	x

b	y'z'(00)	y'z(01)	yz(11)	yz'(10)
x'(0)	x	1	x	0
x(1)	1	x	x	x

x	y	z	a	b
0	0	1	0	1
0	1	0	1	0
1	0	0	1	1

a = z' and **b= y'** (Best design but having security problems)

- **Problems:**
  - On input 000, outputs 11, (a **backdoor** to the case of input 100).
  - On input 011 or 111, output 00, (a **fault injection attack** to the next module).

Design Vulnerabilities (Combinational Circuit): Conclusion

- It is clear that, when the designer designs the system with don't care terms, the CAD tools which are used for synthesis may add unwanted outputs which cause security vulnerabilities.

- A naive way to design a secure circuit is to define outputs for all inputs, i.e., completely specify the system.

- However, this method may not be helpful since optimization cannot be done properly to reduce the area and power consumption. Designers must look for other possibilities to make the system secure.

Sequential System Design Process

- **Given:** system description
- **Goal:** logic diagram, Boolean function expression
- **Steps:**
  1. System specification
  2. State transition table construction
  3. State minimization and encoding
  4. Flip-flop selection
  5. Excitation/output table derivation
  6. Logic simplification
  7. Logic diagram drawing

Sequential System Design: State Encoding

- Each state of an FSM are represented with a unique pattern of high and low register output signals through a process called “**Encoding”.**
- To ensure efficient performance and resource usage it is vital to choose an Encoding scheme that best suits the design.

- There are many Encoding schemes available, the most important ones among them are Binary and One-Hot encoding.

- **Binary Encoding:** This scheme is very useful and efficient when there are even
- number of states. The minimum state bits required in binary coding can be found from the formula B = log2(S).

- Example of Binary encoding
  - State1 = ”00”
  - State2 = ”01”
  - State3 = ”10”
  - State4 = ”11”

Sequential System Design: State Encoding

- Another variation of binary encoding is called Gray encoding in which only one state bit changes at a time.
- Example of Gray encoding
  - State1 = ”00”
  - State2 = ”01”
  - State3 = ”11”
  - State4 = ”10”
- Gray encoding is useful when the outputs of the state bits are used synchronously.

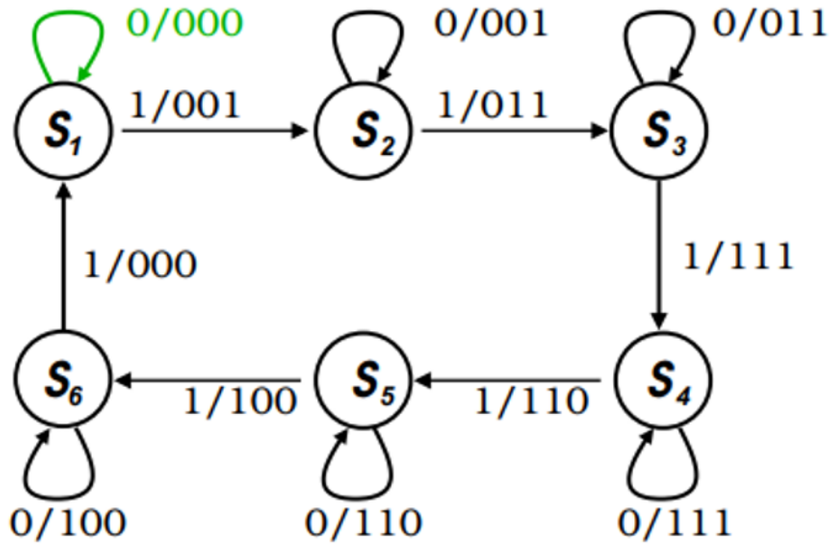
- **One-Hot coding:** It uses one register for each state, i.e., 4 registers for a 4-state state machine with only one state bit at a high logic level at one time.

- Example of One-Hot encoding
  - State1 = ”0001”
  - State2 = ”0010”
  - State3 = ”0100”
  - State4 = ”1000”

Sequential System Design: System Specification Example

- Design a circuit with one input x and three outputs A,B,C. An external source feeds x one bit per clock cycle, when x=0, the outputs remain no change; otherwise, they repeat the binary sequence: 0,1,3,7,6,4, one at a time.

State Transition Graph/ Table



current state			next state x=0			x=1		
A	B	C	A	B	C	A	B	C
0	0	0	0	0	0	0	0	1
0	0	1	0	0	1	0	1	1
0	1	1	0	1	1	1	1	1
1	1	1	1	1	1	1	1	0
1	1	0	1	1	0	1	0	0
1	0	0	1	0	0	0	0	0

Incompletely Specified Design

Sequential System Design: System Specification

- The sequential system design specification is given in the form of a State transition diagram with a brief description of it.

- **For the Fig. below the details is given as follows:** Design an FSM with four states and with one controlling input X and one output Y as shown in the State diagram.

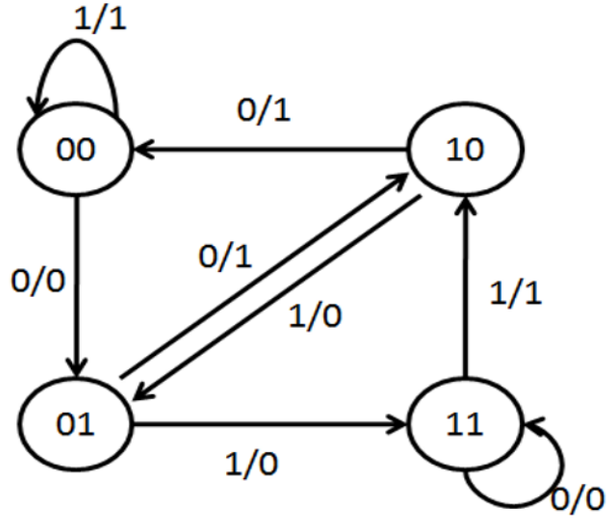


Fig. Required original FSM

Sequential System Design: State Transition Table Construction

- The state transition table is constructed by making a table for current and next states depending on the controlling input
- For the given FSM by seeing the diagram we can construct the state transition table as follows:-

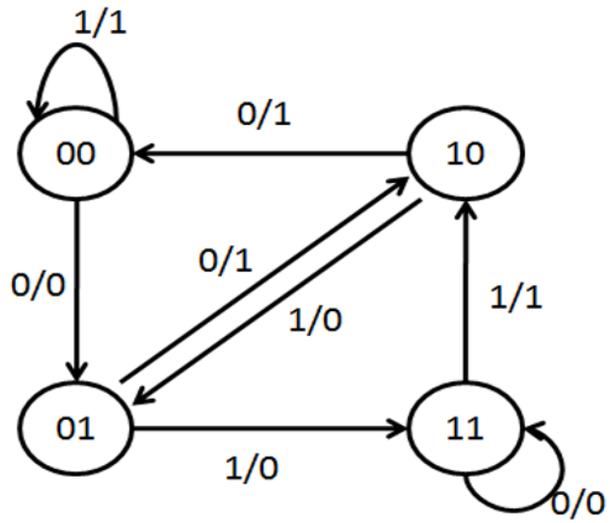


Fig. Required original FSM

Current State		In	Next State		Out
A	B	x	A	B	y
0	0	0	0	1	0
0	0	1	0	0	1
0	1	0	1	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	0	1	0
1	1	0	1	1	0
1	1	1	1	0	1

Completely Specified System Design

Sequential System Design: Flipflop selection

- Flipflop selection is mostly based on the designer. one can choose between either D or T flipflop depending on intuition which gives better performance and less area and power.

- For the given FSM we are using T-flipflop. T stands for toggle; the output is the same as previous input for T = 0 and output is complemented of previous input for T = 1.

T Flip flop Truth Table

Q(t)	Q(t+1)	T
0	0	0
0	1	1
1	0	1
1	1	0

Sequential System Design: Output table Derivation

- The excitation table is constructed by setting the flipflops in appropriate modes such that we get the next state from the current state depending on the controlling input.

- For the given FSM, the excitation is as follows:

Current State		In	Next State		Flipflop inputs		Out
A	B	x	A	B	TA	TB	y
0	0	0	0	1	0	1	0
0	0	1	0	0	0	0	1
0	1	0	1	0	1	1	1
0	1	1	1	1	1	0	0
1	0	0	0	0	1	0	1
1	0	1	0	1	1	1	0
1	1	0	1	1	0	0	0
1	1	1	1	0	1	1	1

Sequential System Design: Logic simplification

- The simplified logic cab obtained for TA, TB and Y using K-Map as follows:

Current State		In	Next State		Flipflop inputs		Out
A	B	x	A	B	TA	TB	y
0	0	0	0	1	0	1	0
0	0	1	0	0	0	0	1
0	1	0	1	0	1	1	1
0	1	1	1	1	1	0	0
1	0	0	0	0	1	0	1
1	0	1	0	1	1	1	0
1	1	0	1	1	0	0	0
1	1	1	1	0	1	1	1

TA	A'B'(00)	A'B(01)	AB(11)	AB'(10)
X'(0)	0	1	0	1
X(1)	0	1	0	1

TB	A'B'(00)	A'B(01)	AB(11)	AB'(10)
X'(0)	1	1	0	0
X(1)	0	0	1	1

Y	A'B'(00)	A'B(01)	AB(11)	AB'(10)
X'(0)	0	1	0	0
X(1)	1	0	1	0

$TA = A'B + AB'$ $TB = A'X' + AX$ $Y = A'B'X + A'BX' + ABX$

$TA = A \oplus B$ $TB = (A \oplus X)'$ $Y = A \oplus B \oplus X$

Sequential System Design: Logic Diagram

- From logic simplification we obtained the logic expressions as TA = A ⊕ B, TB = (A ⊕ X) and Y = A ⊕ B ⊕ X.

- The logic diagram that will be delivered by you should look like this:-

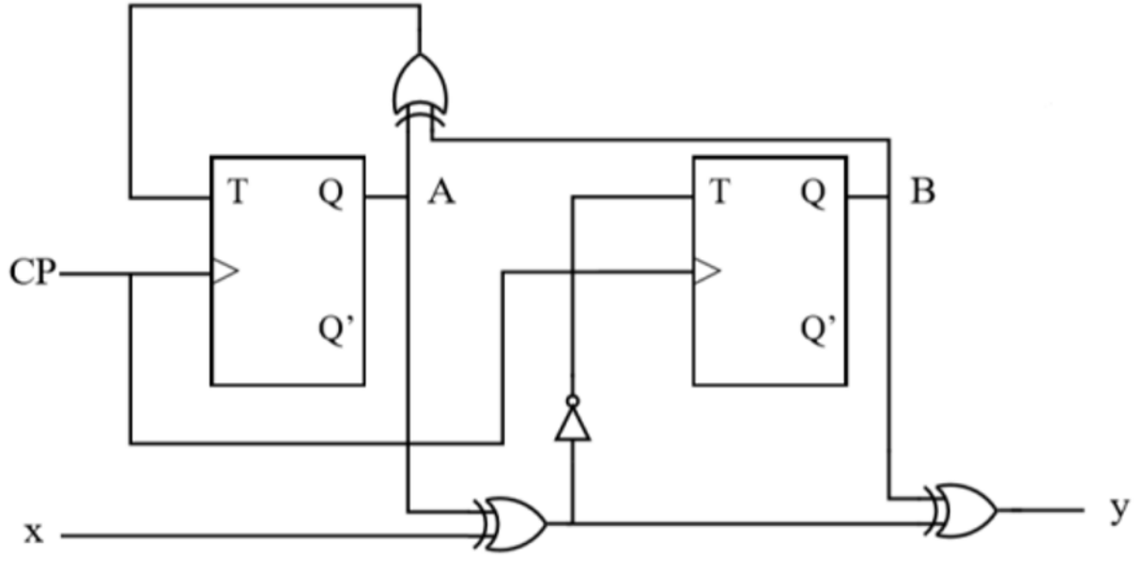


Fig. Delivered architecture

Sequential System Design: Example

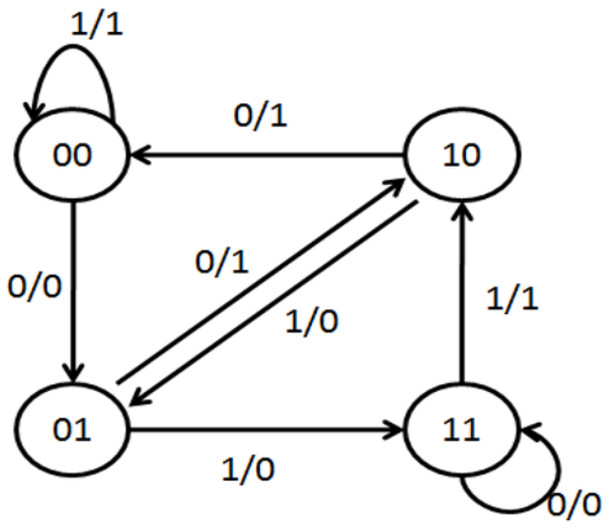


Fig. Required original FSM

T Flip flop Truth Table

Q(t)	Q(t+1)	T
0	0	0
0	1	1
1	0	1
1	1	0

Flip-flop input functions:

$TA = A \oplus B$

$TB = (A \oplus x)'$

Output:

$y = A \oplus B \oplus x$

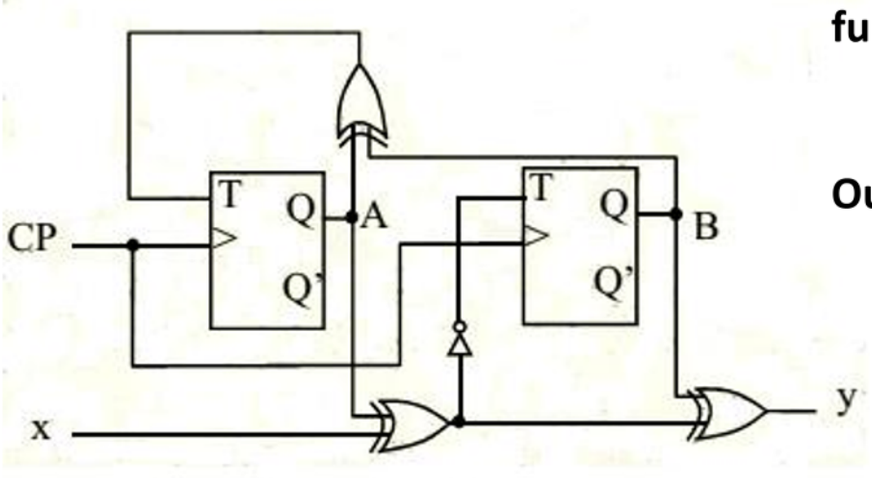


Fig. Delivered architecture

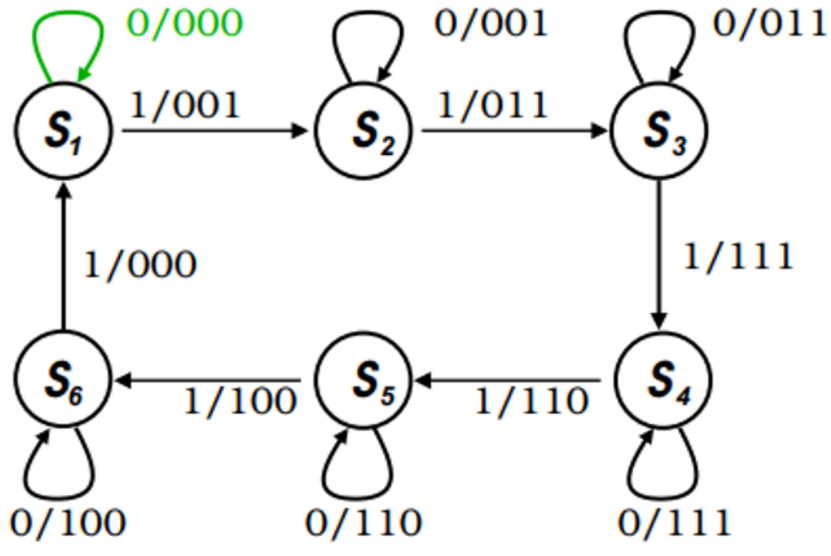
Current State		In	Next State		Flip-flop inputs		Out
A	B	x	A	B	TA	TB	y
0	0	0	0	1	0	1	0
0	0	1	0	0	0	0	1
0	1	0	1	0	1	1	1
0	1	1	1	1	1	0	0
1	0	0	0	0	1	0	1
1	0	1	0	1	1	1	0
1	1	0	1	1	0	0	0
1	1	1	1	0	0	1	1

Completely Specified System Design

Practice Problem

- Design a circuit with one input x and three outputs A,B,C. An external source feeds x one bit per clock cycle, when x=0, the outputs remain no change; otherwise, they repeat the binary sequence: 0,1,3,7,6,4, one at a time.

State Transition Graph/ Table



current state			next state x=0			x=1		
A	B	C	A	B	C	A	B	C
0	0	0	0	0	0	0	0	1
0	0	1	0	0	1	0	1	1
0	1	1	0	1	1	1	1	1
1	1	1	1	1	1	1	1	0
1	1	0	1	1	0	1	0	0
1	0	0	1	0	0	0	0	0

Incompletely Specified

Self-Correcting Sequential System Design: Example

- Design a system that generates repeated binary sequence of 0,1,3,5,7. (Incompletely Specified)

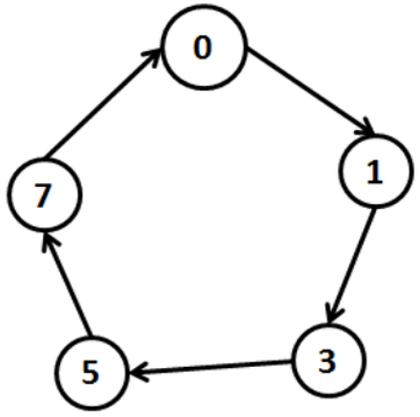


Fig. Required FSM and Transition Table

State Transition Table

Current State				Next State		
	A	B	C	A	B	C
0	0	0	0	0	0	1
1	0	0	1	0	1	1
3	0	1	1	1	0	1
5	1	0	1	1	1	1
7	1	1	1	0	0	0

- The given system is incompletely specified it means some of the states in the FSM are not defined.
- The given FSM is defined for only 0,1,3,5,7. States 2, 4 and 6 are not specified, they can be used for fault injection-based attacks.
- The faults include incorrect transitions, backdoors and hardware trojan insertion.

Self-Correcting Sequential System Design: Example (Cont..)

- Design a system that generates repeated binary sequence of 0,1,3,5,7. (Incompletely Specified)

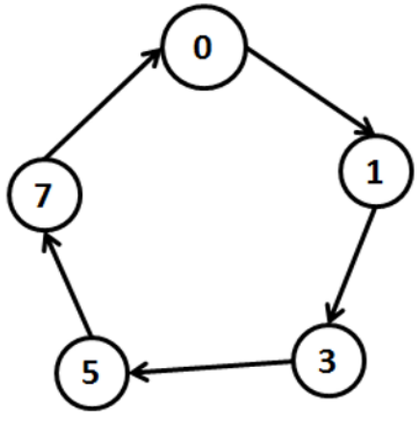


Fig. Required FSM and Transition Table

State Transition Table

Current State				Next State		
	A	B	C	A	B	C
0	0	0	0	0	0	1
1	0	0	1	0	1	1
3	0	1	1	1	0	1
5	1	0	1	1	1	1
7	1	1	1	0	0	0

- The given system is incompletely specified it means some of the states in the FSM are not defined.
- The given FSM is defined for only 0,1,3,5,7. States 2, 4 and 6 are not specified, they can be used for fault injection-based attacks.
- The faults include incorrect transitions, backdoors and hardware trojan insertion.

Self-Correcting Sequential System Design: Example (Cont..)

- The output excitation table, obtained logic expressions and diagram can be seen below:

Current State				Next State			Flip-flop Inputs		
	A	B	C	A	B	C	TA	TB	TC
0	0	0	0	0	0	1	0	0	1
1	0	0	1	0	1	1	0	1	0
3	0	1	1	1	0	1	1	1	0
5	1	0	1	1	1	1	0	1	0
7	1	1	1	0	0	0	1	1	1
2	0	1	0	-	-	-	-	-	-
4	1	0	0	-	-	-	-	-	-
6	1	1	0	-	-	-	-	-	-

- TA = B
- TB = C
- TC = C'+AB

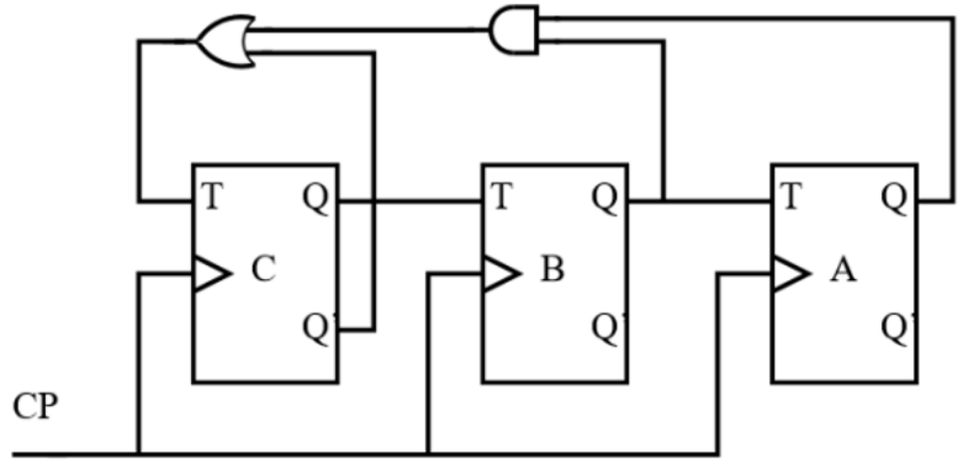


Fig. Delivered architecture

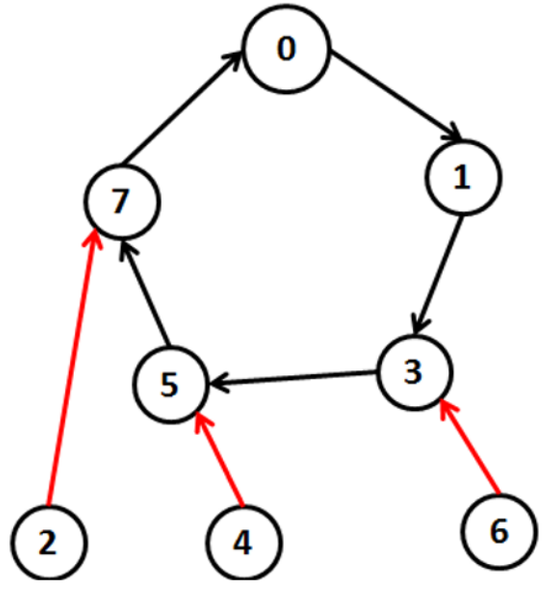
- Since obtained design is deterministic, we identify the value undefined states using above expressions/ architecture.

Self-Correcting Sequential System Design: Example (Cont..)

- From the logic expression it can be seen that from state 2 we can go to state 7, for 4-5 and for 6-3, i.e, the design is self correcting.

Current State				Next State			Flip-flop Inputs		
	A	B	C	A	B	C	TA	TB	TC
0	0	0	0	0	0	1	0	0	1
1	0	0	1	0	1	1	0	1	0
3	0	1	1	1	0	1	1	1	0
5	1	0	1	1	1	1	0	1	0
7	1	1	1	0	0	0	1	1	1
2	0	1	0	1	1	1	1	0	1
4	1	0	0	1	0	1	0	0	1
6	1	1	0	0	1	1	1	0	1

- TA = B
- TB = C
- TC = C'+AB



Design is self-correcting

Self-Correcting Sequential System Design: Example

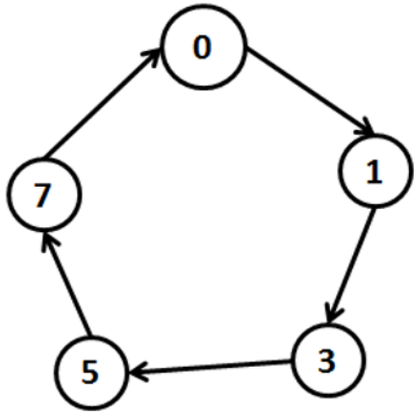
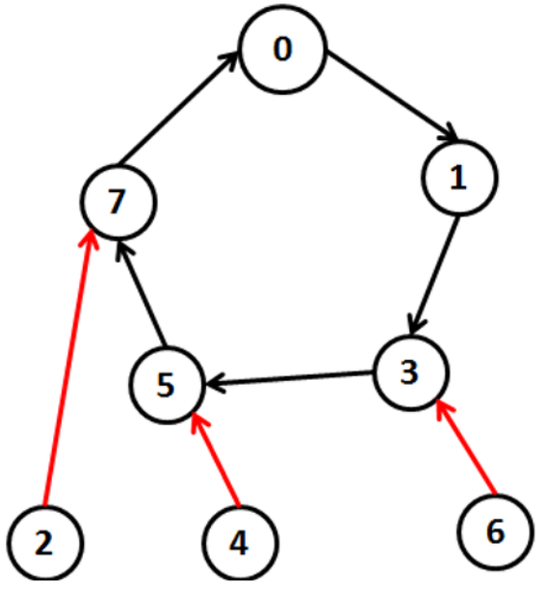


Fig. Original design

	A	B	C	A	B	C
0	0	0	0	0	0	1
1	0	0	1	0	1	1
3	0	1	1	1	0	1
5	1	0	1	1	1	1
7	1	1	1	0	0	0

- TA = B
- TB = C
- TC = C'+AB



Design is self-correcting

	A	B	C	A	B	C	TA	TB	TC
0	0	0	0	0	0	1	0	0	1
1	0	0	1	0	1	1	0	1	0
3	0	1	1	1	0	1	1	1	0
5	1	0	1	1	1	1	0	1	0
7	1	1	1	0	0	0	1	1	1
2	0	1	0	1	1	1	1	0	1
4	1	0	0	1	0	1	0	0	1
6	1	1	0	0	1	1	1	0	1

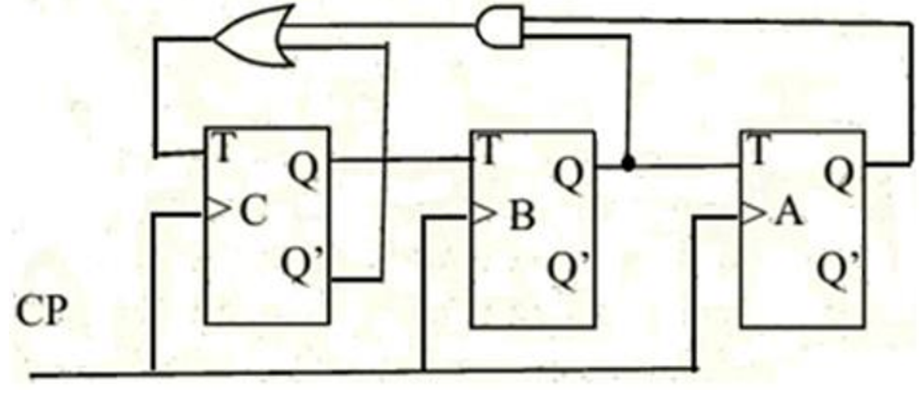


Fig. Delivered design and transition table

Summary of The Class

- Combinational Circuit Design
  - Design Vulnerabilities in Combinational Circuit
- Sequential System Design Process
  - State Encoding
  - Sequential circuit Design
- Self-Correcting Sequential System Design

Thank You