Testing Digital Systems I

Lecture 11:
Test Generation for Sequential Circuits

Instructor: M. Tahoori

Sequential Circuits

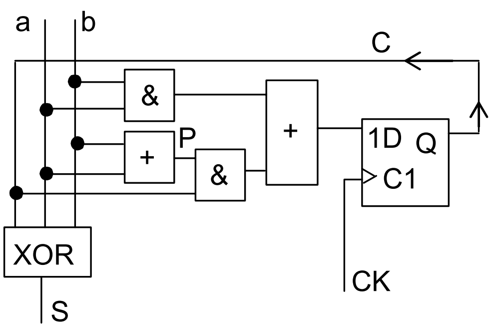
- Approach
  - Convert Finite State Machine to Corresponding Iterative Network
    - Multiple Time Frames (Iterative Cells) Needed for
      - Justification and Propagation
  - One Fault in Sequential Circuit
    - Many Faults in Corresponding Iterative Network
    - Use 9-valued signals
- Issues
  - Order of Justification and Propagation
  - Simulation Values
  - Test Point Insertion (Partial Scan)

Sequential ATPG

- Difficulties
  - Initialization of the bistables
  - Gated clocks
  - Circuits with multiple clock domains
  - Internally derived clocks, mixed data and clock signals
  - Asynchronous logic
  - Circuits with combinational feedback paths
  - Embedded counters
  - Embedded RAMs and ROMs

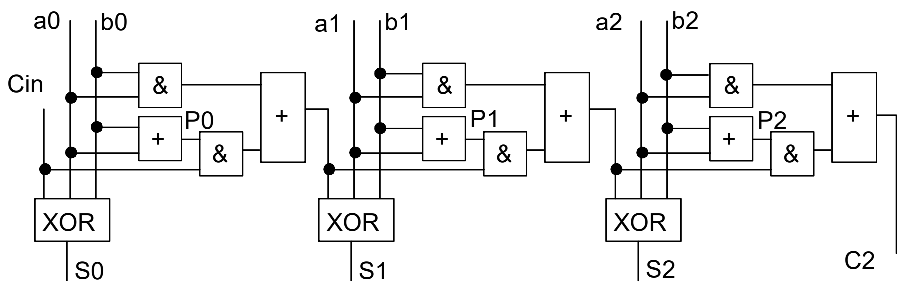
Finite State Machine

- Example : serial adder
  - $S_i = a_i \oplus b_i \oplus c_{i-1}$
  - $C_i = a_i\, b_i + c_{i-1}\,(a_i + b_i\,)$



Example

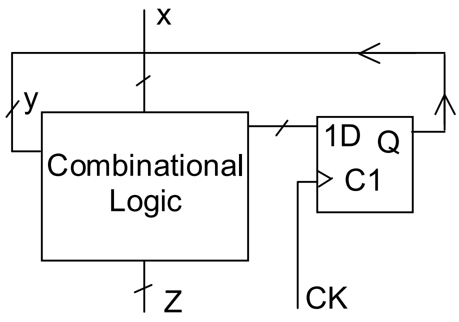
- Corresponding parallel binary adder circuit
  - Iterative network of the previous circuit



Copyright 2019, M. Tahoori   TDS I: Lecture 11   5

General Case

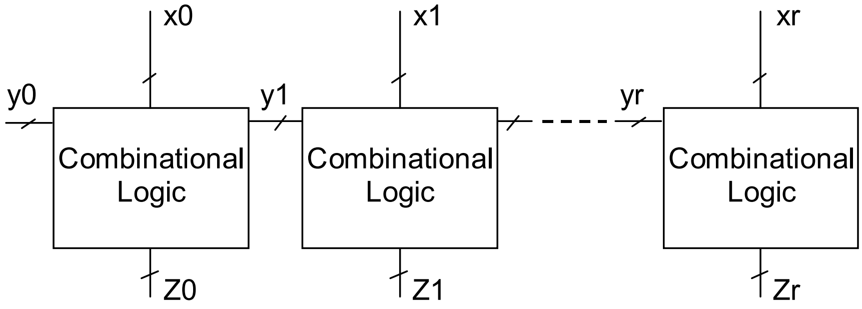
- Huffman model of sequential circuit
  - with edge-triggered D-flip-flops



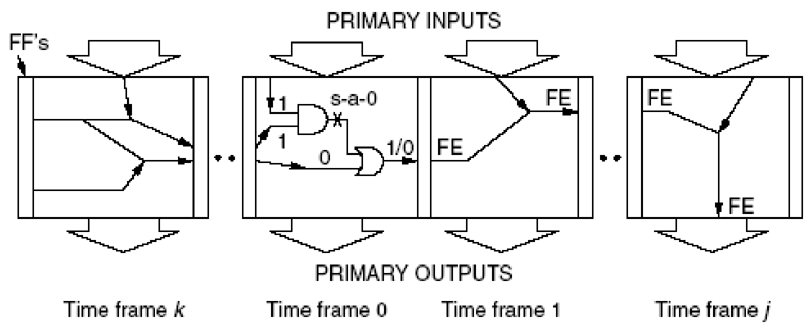
Copyright 2019, M. Tahoori   TDS I: Lecture 11   6

General Case

- Any sequential circuit with edge-triggered D-FF
  - can be directly converted into an iterative network



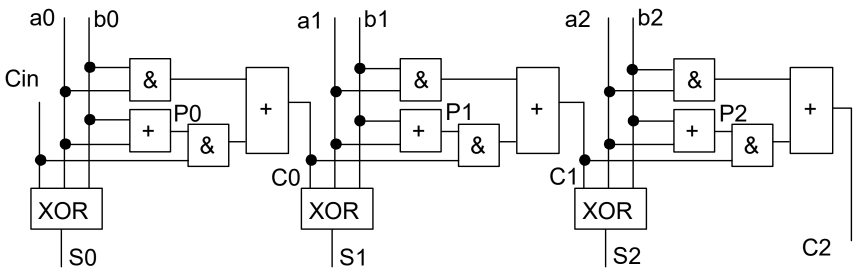
Iterative Logic Array Expansion



- To detect a fault, a sequence of vectors may be needed

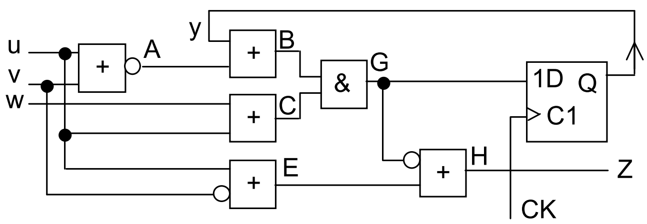
Example

- Test for P SA0
  - Provoke Fault on P1: a1 = 0, b1 = 1
  - Propagate Fault to S2:
    - C0 = 1
      - Need to consider last time frame: : a0 = 1, b0 = 1, Cin = X
    - a2 = 0, b2 = 0



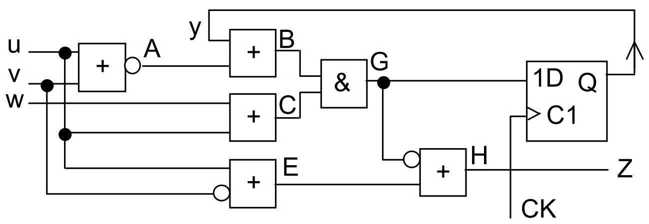
Example

- Test for u SA1
  - In time frame t
    - provoke fault: u = 0
    - propagate fault effect to Z (path E,H,Z): v = 1 and G = 1
      - Justify B = C = 1: y = 1 and w = 1
    - Requires G = 1 in the time frame t-1
      - for a don't care value of y (i.e., y = 0 or 1) in the time frame t - 1



Example (cont)

- Test for u SA1
  - In time frame t - 1
    - G = 1 , y = X
      - A = 1
      - U = D′ ⇒ A = 0 in the faulty circuit
  - Conflict !

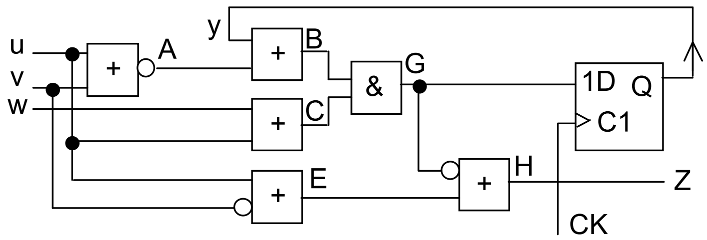


Example (cont)

- Problem with this example
  - Try to extend D-algorithm for sequential circuits
- Z is 1 in the presence of u/1 irrespective of
  - the logic values on other signal lines, and
  - the content of the flip-flop

Example (cont)

- Input sequence to set Z to 0 in the fault-free circuit
  - E = 0 and G = 1
    - B = 1 ⇒ y = 1
      - u = v = 0 and w = 1 in cycle t - 1
    - u = 0, v = 1 and w = 1 in cycle t



Nine-Valued Signals

- A fault can be detected even if in the presence of the fault a signal line in the faulty circuit has an unknown value (X)
  - While the corresponding signal line in the fault-free circuit has a known value (0 or 1) or vice-versa
- This information is not expressed by the logic values 0, 1, D and D′ introduced in the context of the D-algorithm

Nine-Valued Signals

Symbol	Symbol	Fault-free circuit value	Faulty Circuit Value
<0, 0>	0	0	0
<1, 1>	1	1	1
<1, 0>	D	1	0
<0, 1>	D′	0	1
<X, X>	X	0 or 1	0 or 1
<0, X>	G0	0	0 or 1
<1, X>	G1	1	0 or 1
<X, 0>	F0	0 or 1	0
<X, 1>	F1	0 or 1	1

Using Nine-valued Signals

- Propagate assigned values
- Assign values to propagate D or $\overline{D}$
- Assign values to provoke D or $\overline{D}$ at stuck fault gate output
  - Primitive D-cube of the fault
- Line Justification

Propagating Assigned Values

NOT	
0	1
1	0
D	D′
D′	D
X	X
G0	G1
G1	G0
F0	F1
F1	F0

AND	0	1	D	D′	X	G0	G1	F0	F1
0	0	0	0	0	0	0	0	0	0
1	0	1	D	D′	X	G0	G1	F0	F1
D	0	D	D	0	F0	0	D	F0	F0
D′	0	D′	0	D′	G0	G0	G0	0	D′
X	0	X	F0	G0	X	G0	X	F0	X
G0	0	G0	0	G0	G0	G0	G0	0	G0
G1	0	G1	D	G0	X	G0	G1	F0	X
F0	0	F0	F0	0	F0	0	F0	F0	F0
F1	0	F1	F0	D′	X	G0	X	F0	F1

Propagation D-Cubes

- For propagating the fault effect through an OR gate with D input, apply <X, 0> to the other inputs of the OR gate.
- For propagating the fault effect through an OR gate with D′ input, apply <0, X> to the other inputs of the OR gate.
- For propagating the fault effect through an AND gate with D input, apply <1, X> to the other inputs of the AND gate.
- For propagating the fault effect through an AND gate with D′ input, apply <X, 1> to the other inputs of the AND gate.

Assigning Values To Provoke D Or $\overline{D}$

- At stuck fault gate output
- Primitive D-cube of the fault

AND Gate with output SA0

a	b	Z
<1,X>	<1,X>	D

AND Gate with output SA1

a	b	Z
<0,X>	<X,X>	$\overline{D}$
<X,X>	<0,X>	$\overline{D}$

AND Gate with input a SA0

a	b	Z
<1,X>	<1,X>	D

AND Gate with input a SA1

a	b	Z
<0,X>	<X,1>	$\overline{D}$

Line Justification

AND Gate with output <X,1>

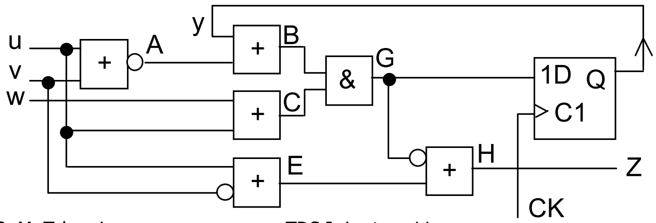
a	b	Z
<X,1>	<X,1>	<X,1>

AND Gate with output <0,X>

a	b	Z
<X,X>	<0,X>	<0,X>
<0,X>	<X,X>	<0,X>

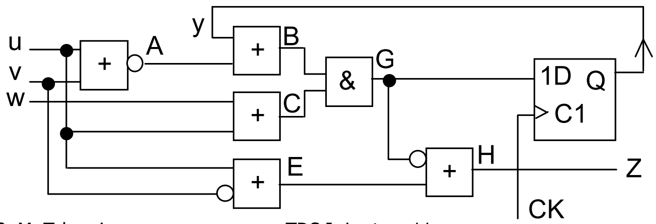
Example: u/1

- Provoke the fault
  - Apply <0, 1> = D′ on signal line u at time frame t
- Propagate fault effect (along path through E and H to Z)
  - v and G must be <1, X> = G1
  - v is a primary input ⇒ v = <1, 1> at time t
    - A = <0, 0>
  - justify B = C = <1, X> (= G1)
    - justifying C = <1, X> ⇒ w = C= <1,1> at time frame t
    - justifying B = <1, X> ⇒ justify y = <1, X>
      - justify G = <1, X> at time frame t-1



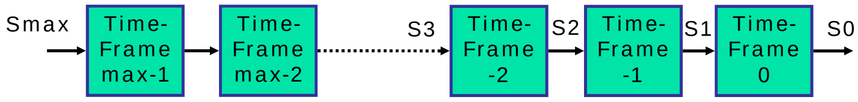
Example: u/1 (cont)

- justifying G = <1, X> (= G1) in time frame t-1
  - justify B = C = <1, X> at time frame t-1
    - justify C = <1, X> by setting w = 1 in time frame t-1
    - justifying <1, X> on B ⇒ set A = <1, X> and y = <X, X>
      - A = <1, X> ⇒ u = <0, X> and v = <0, X> in time frame t-1
        - u = v = 0 at time frame t-1
- Test for u/1
  - u = 0, v = 0 and w = 1 in time frame t-1
  - u = 0, v = 1 and w = 1 in time frame t



Complexity of ATPG

- Synchronous circuit
  - All flip-flops controlled by clocks; PI and PO synchronized with clock:
    - Cycle-free circuit – No feedback among flip-flops
      - Test generation for a fault needs no more than *dseq* + 1 time-frames
      - *dseq* is the sequential depth.
    - Cyclic circuit – Contains feedback among flip-flops:
      - May need 9^{Nff} time-frames
        - *Nff* is the number of flip-flops.
- Asynchronous circuit – Higher complexity!



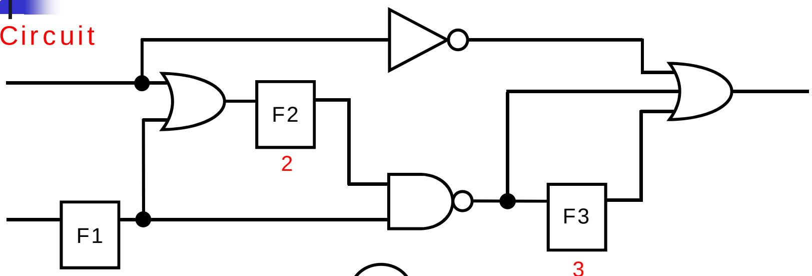
max = Number of distinct vectors with 9-valued elements = 9^{Nff}

Cycle-Free Circuits

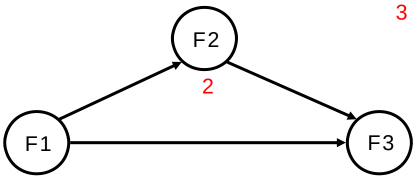
- Characterized by
  - Absence of cycles among flip-flops and
  - a sequential depth, dseq.
- dseq is the maximum number of flip-flops on any path between PI and PO.
- Both good and faulty circuits are initializable.
- Test sequence length for a fault
  - is bounded by dseq + 1.

Cycle-Free Example

Circuit



s - graph



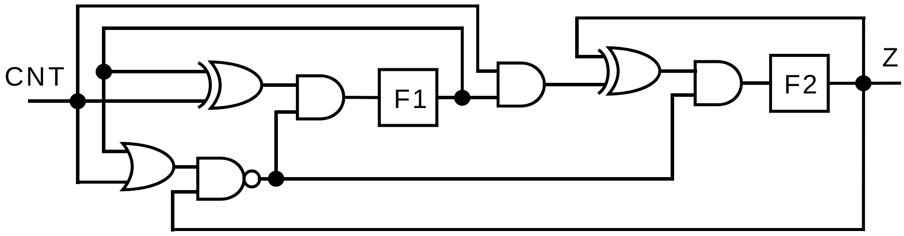
dseq = 3

All faults are testable

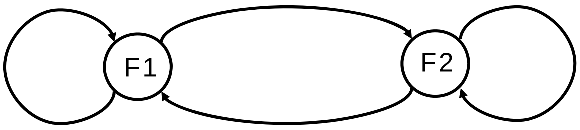
Copyright 2019, M. Tahoori TDS I: Lecture 11 25

Cyclic Circuit Example

Modulo-3 counter



s - graph



Copyright 2019, M. Tahoori TDS I: Lecture 11 26

Modulo-3 Counter

- Cyclic structure
  - Sequential depth is undefined.
- Circuit is not initializable.
  - No tests can be generated for any stuck-at fault.
- After expanding the circuit to $9^{Nff} = 81$, or fewer, time-frames ATPG program calls any given target fault untestable.
- Circuit can only be functionally tested by multiple observations.
- Functional tests, when simulated, give no fault coverage.

Summary

- Combinational ATPG algorithms are extended:
  - Time-frame expansion unrolls time as combinational array
  - Nine-valued logic system
  - Justification via backward time
- Cycle-free circuits:
  - Require at most dseq time-frames
    - dseq is the maximum number of flip-flops on any path between PI and PO
  - Always initializable
- Cyclic circuits:
  - May need 9^{Nff} time-frames
    - Nff is the number of flip-flops
  - Circuit must be initializable
  - Partial scan can make circuit cycle-free
- Asynchronous circuits:
  - High complexity
  - Low coverage and unreliable tests

