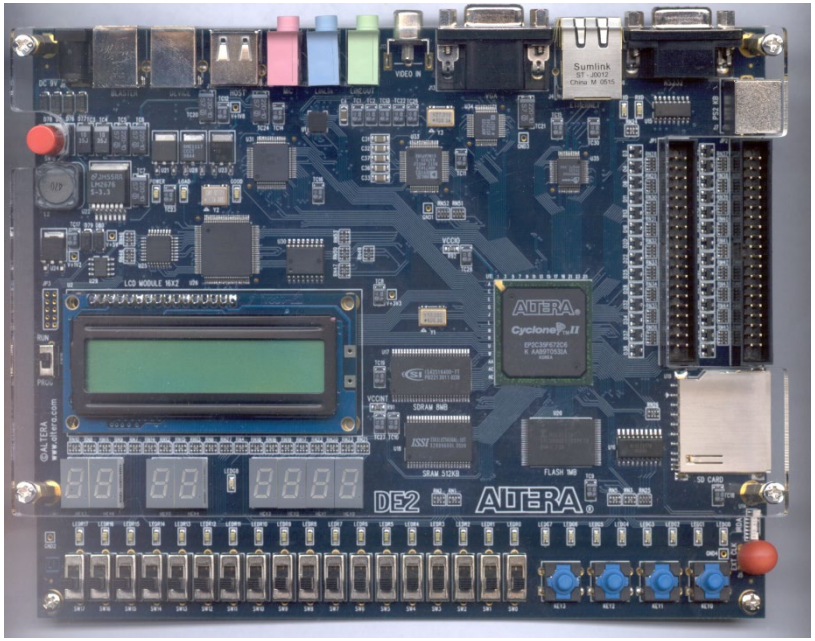
CPE 322
Digital Hardware Design Fundamentals

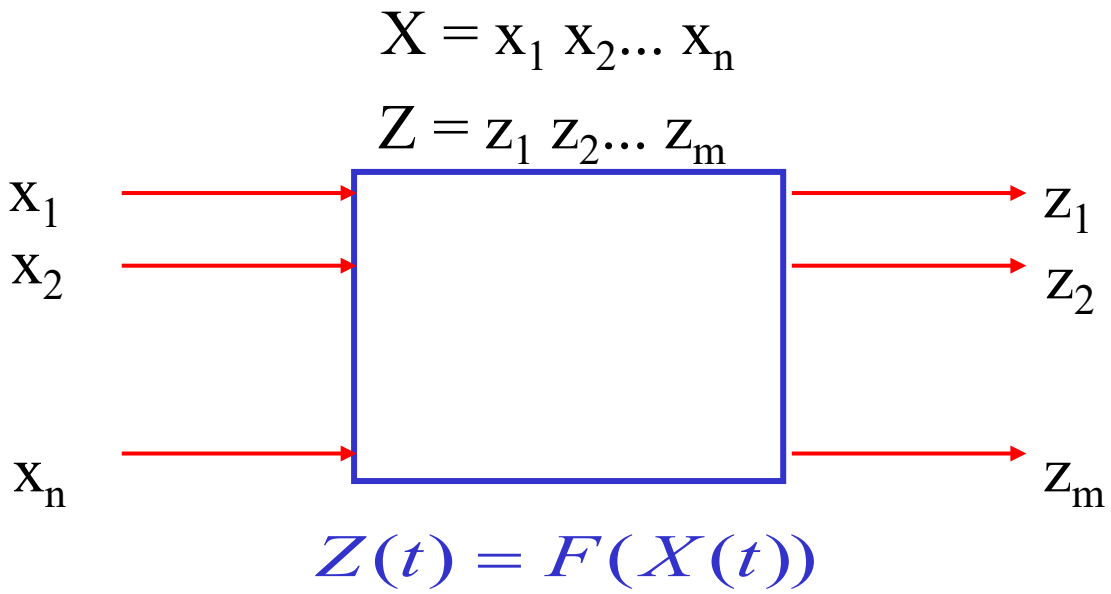
Electrical and Computer Engineering
University of Alabama in Huntsville

Hazard Detection and Prevention in Combinational Logic



Combinational Networks

- Has no memory (state)
  - Present is only a function of the current input, past history is not important



No memory (no feedback) but there often propagation delays in real-world networks!

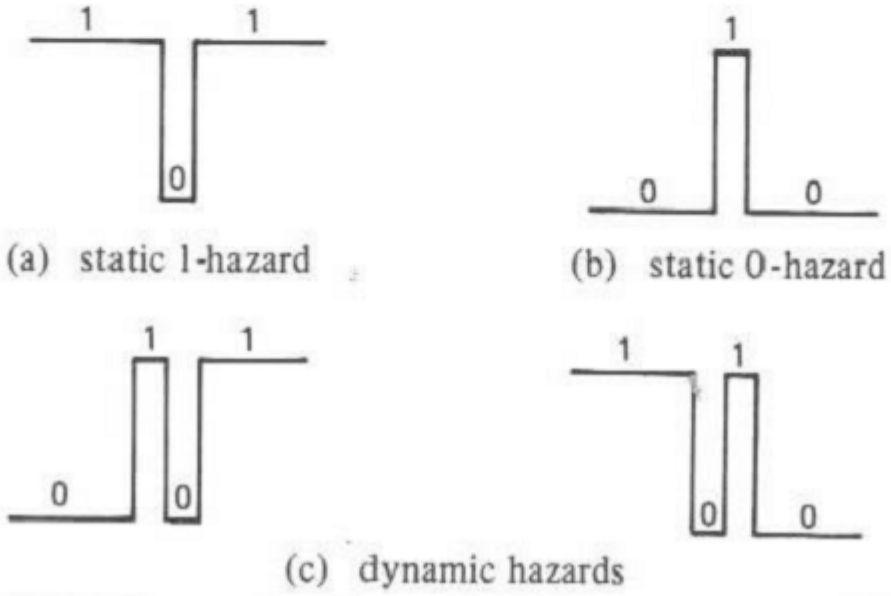
Hazards in Combinational Networks

- Occur when different paths from input to output have different propagation delays
- Static 1-hazard
  - a network output momentarily go to the 0 when it should remain a constant 1
- Static 0-hazard
  - a network output momentarily go to the 1 when it should remain a constant 0
- Dynamic hazard
  - if an output change three or more times, when the output is supposed to change from 0 to 1 (1 to 0)

Hazards in Combinational Networks

- **When an input to a combinational network changes, unwanted switching transients may appear at the network output.**
- **These transients occur because different paths through the network from input to output may have different propagation delays.**

Types of Hazards:



Hazards in Combinational Networks (CNs)

- We only consider hazards which occur when a single input variable changes

- Analysis begins by determining the *Transient Output Function*, F^t, which represents the behavior of the network under transient conditions.

The *Transient Output Function*, F^t, is determined in the same way as ordinary (steady-state) output Functions except each variable and its complement are treated as independent variables because under transient conditions these variables may assume the same values.

Laws and Theorems of Boolean Algebra

Boolean manipulation of F^t involves the use of theorems that do not involve a variable and its complement on the same side of the expression.

Operations with 0 and 1:

$X + 0 = X$	(1-5)	$X \cdot 1 = X$	(1-5D)
$X + 1 = 1$	(1-6)	$X \cdot 0 = 0$	(1-6D)

Idempotent laws:

$X + X = X$	(1-7)	$X \cdot X = X$	(1-7D)

Involution law:

$(X')' = X$	(1-8)		

Laws of complementarity

$X + X' = 1$	(1-9)	$X \cdot X' = 0$	(1-9D)

Commutative laws:

$X + Y = Y + X$	(1-10)	$XY = YX$	(1-10D)

Associative laws:

$(X + Y) + Z = X + (Y + Z)$ $= X + Y + Z$	(1-11)	$(XY)Z = X(YZ) = XYZ$	(1-11D)

Distributive laws:

$X(Y + Z) = XY + XZ$	(1-12)	$X + YZ = (X + Y)(X + Z)$	(1-12D)

Laws and Theorems of Boolean Algebra

Simplification theorems:

$XY + XY' = X$	(1-13)	$(X + Y)(X + Y') = X$	(1-13D)
$X + XY = X$	(1-14)	$X(X + Y) = X$	(1-14D)
$(X + Y')Y = XY$	(1-15)	$XY' + Y = X + Y$	(1-15D)

DeMorgan's laws:

$(X + Y + Z + \cdots)' = X'Y'Z'\cdots$	(1-16)	$(XYZ\cdots)' = X' + Y' + Z' + \cdots$	(1-16D)
$[f(X_1, X_2, \ldots, X_n, 0, 1, +, \cdot)]' = f(X_1', X_2', \ldots, X_n', 1, 0, \cdot, +)$	(1-17)		

Duality:

$(X + Y + Z + \cdots)^D = XYZ\cdots$	(1-18)	$(XYZ\cdots)^D = X + Y + Z + \cdots$	(1-18D)
$[f(X_1, X_2, \ldots, X_n, 0, 1, +, \cdot)]^D = f(X_1, X_2, \ldots, X_n, 1, 0, \cdot, +)$	(1-19)		

Theorem for multiplying out and factoring:

$(X + Y)(X' + Z) = XZ + X'Y$	(1-20)	$XY + X'Z = (X + Z)(X' + Y)$	(1-20D)

Consensus theorem:

$XY + YZ + X'Z = XY + X'Z$	(1-21)	$(X + Y)(Y + Z)(X' + Z) = (X + Y)(X' + Z)$	(1-21D)

Boolean manipulation of F^t involves the use of theorems that do not involve a variable and its complement on the same side of the expression.

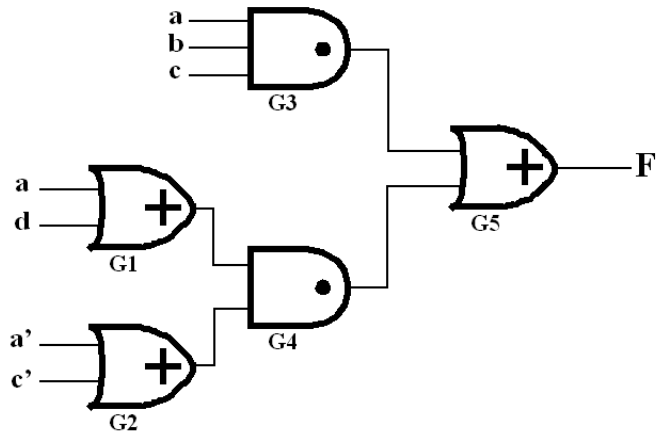
Detection of Static 1 Hazards

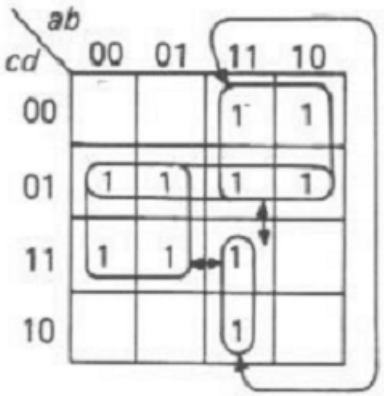
- Each product term of F^t is called a 1-term.
- The 1-terms of F^t are plotted on a Karnaugh map
- If two 1's in adjacent squares on the map of F^t are covered by the same 1-term, changing the input to the network between the corresponding two input states cannot cause a hazard.
- If two 1's in adjacent squares on the map are not covered by a single 1-term, a hazard is present.

Detection of Static 1 Hazards

$$F^t = abc + (a+d)(a' + c') = abc + aa' + ac' + a'd + c'd$$

$$X(Y + Z) = XY + XZ \quad (1\text{-}12)$$





- 1-Hazard exists for case where **a** changes (from 0 to 1 or 1 to 0) and **b=1**, **c=1**, **d=1**.
- 1-Hazard exists for case where **c** changes (from 0 to 1 or 1 to 0) and **a=1**, **b=1**, **d=1**.
- 1-Hazard exists for case where **c** changes (from 0 to 1 or 1 to 0) and **a=1**, **b=1**, **d=0**.

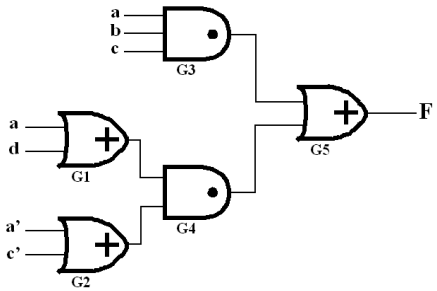
Detection of Static 0 Hazards

- Each sum term of F^t is called a 0-term.
- The 0-terms of F^t are plotted on a Karnaugh map
- If two 0's in adjacent squares on the map of F^t are covered by the same 0-term, changing the input to the network between the corresponding two input states cannot cause a hazard.
- If two 0's in adjacent squares on the map are not covered by a single 0-term, a hazard is present.

Detection of Static 0 Hazards

$F^t = abc + (a+d)(a' + c') = [(a+d)(a'+c')+a][(a+d)(a'+c')+bc]$

$X + YZ = (X+Y)(X+Z)$ (1-12D)



$[(a+d+a)(a'+c' +a)][((a+d)(a'+c')+b)((a+d)(a'+c')+c)]$

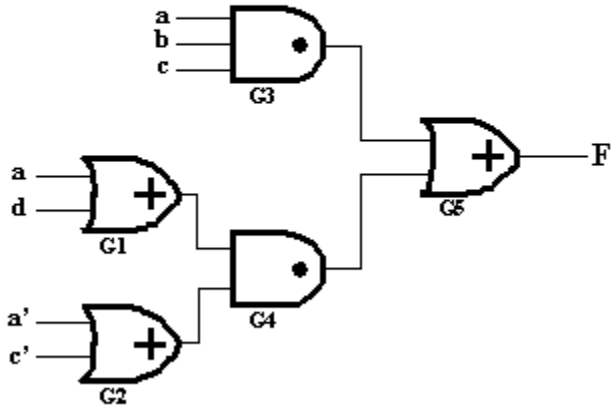
$X + YZ = (X+Y)(X+Z)$ (1-12D)

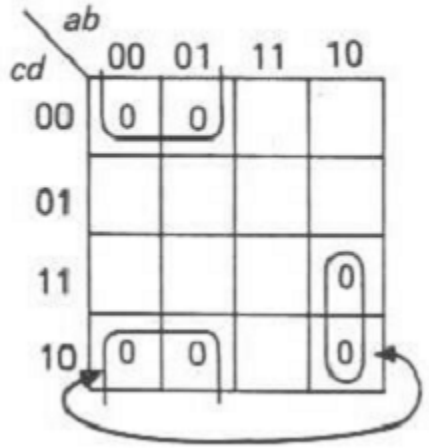
$(a+d)(a'+a+c')(a+d+b)(a'+c'+b)(a+c+d)(a'+c'+c)$

$X(X+Y) = X$ (1-14D)

Detection of Static 0 Hazards

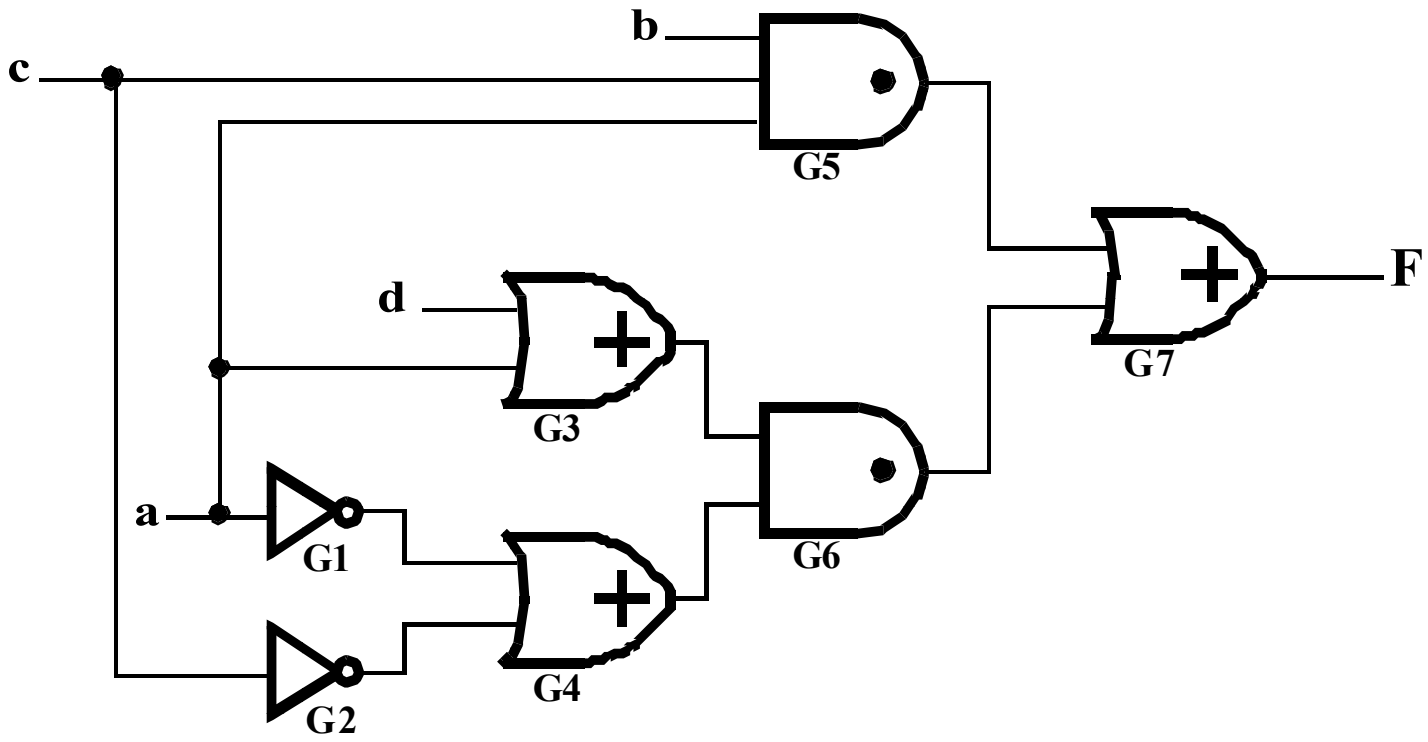
$$F^t = abc + (a+d)(a' + c') = (a+d)(a+a'+c')(b+a'+c')(c+a'+c')$$





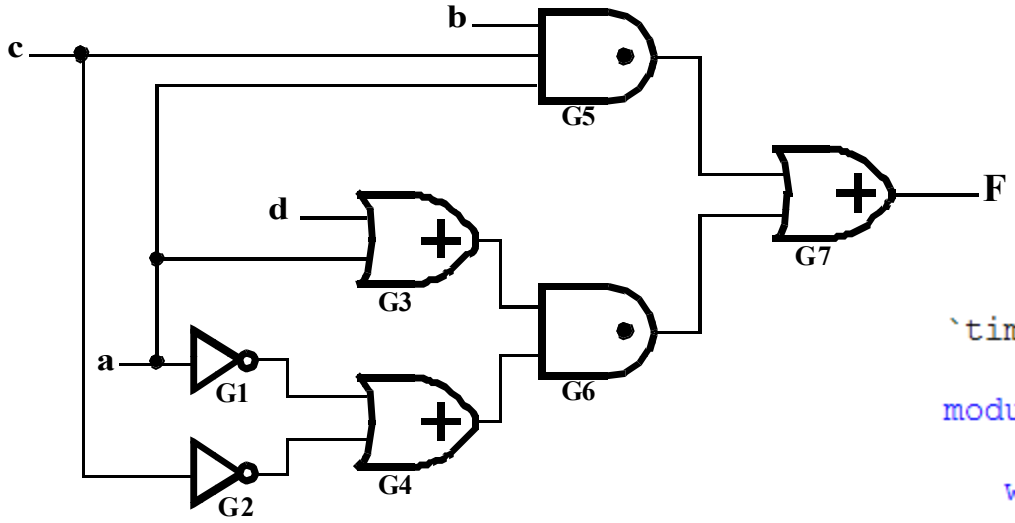
- 0-Hazard exists for case where **a** changes (from 0 to 1 or 1 to 0) and **b=0**, **c=1**, **d=0**.

Using Static Timing to Observe the Effect of Hazards



Assuming 1 ns rise/fall times for <u>each</u> gate (AND, OR, and NOT gates)

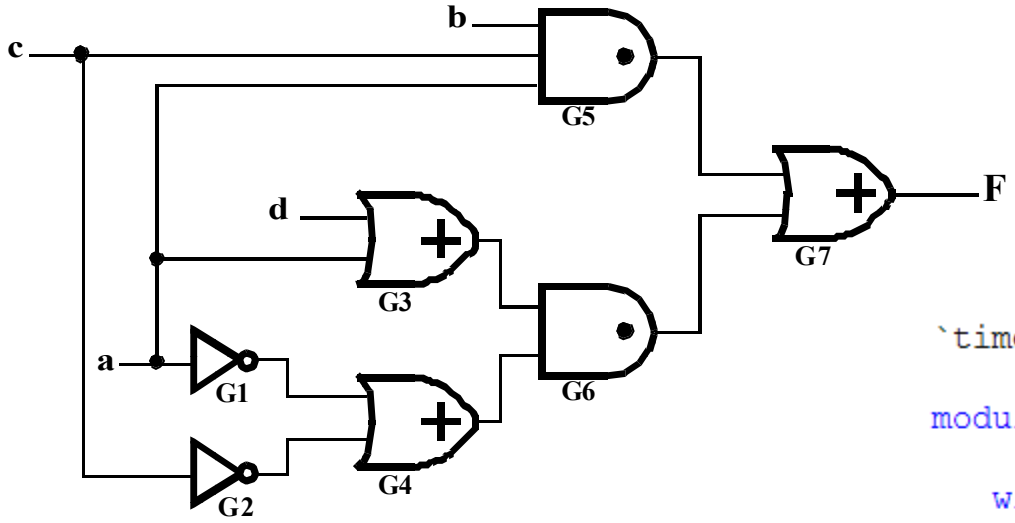
Using Static Timing to Observe the Effect of Hazards



Assuming 1 ns rise/fall times for <u>each</u> gate (AND, OR, and NOT gates)

```
`timescale 1ns/100ps

module example (input a,b,c,d, output F);

   wire a_n,c_n,n1,n2,n3,n4;

   not #(1) G1(a_n,a);
   not #(1) G2(c_n,c);

   or  #(1) G3(n1,d,a);
   or  #(1) G4(n2,a_n,c_n);

   and  #(1) G5(n3,b,c,a);
   and  #(1) G6(n4,n1,n2);
   or   #(1) G7(F,n3,n4);

endmodule
```

Using Static Timing to Observe the Effect of Hazards

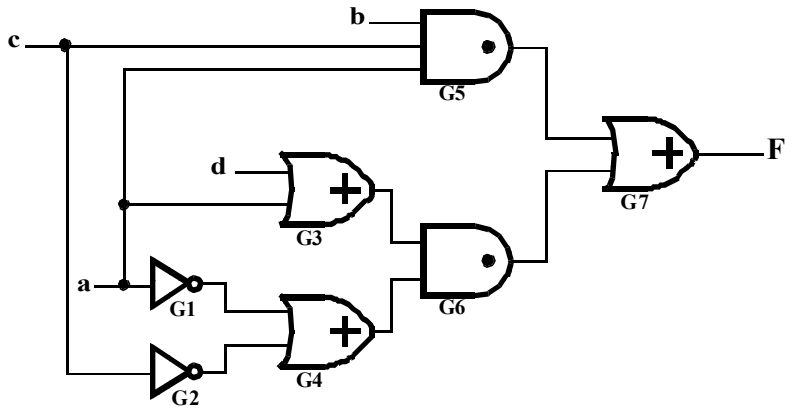


Assuming 1 ns rise/fall times for <u>each</u> gate (AND, OR, and NOT gates)

```
`timescale 1ns/100ps

module example (input a,b,c,d, output F);

   wire a_n,c_n,n1,n2,n3,n4;

   assign #1 a_n = ~a;
   assign #1 c_n = ~c;

   assign #1 n1 = d | a;
   assign #1 n2 = a_n | c_n;

   assign #1 n3 = b & c & a;
   assign #1 n4 = n1 & n2;
   assign #1 F = n3 | n4;

endmodule
```

Driving the input to Detect Hazards

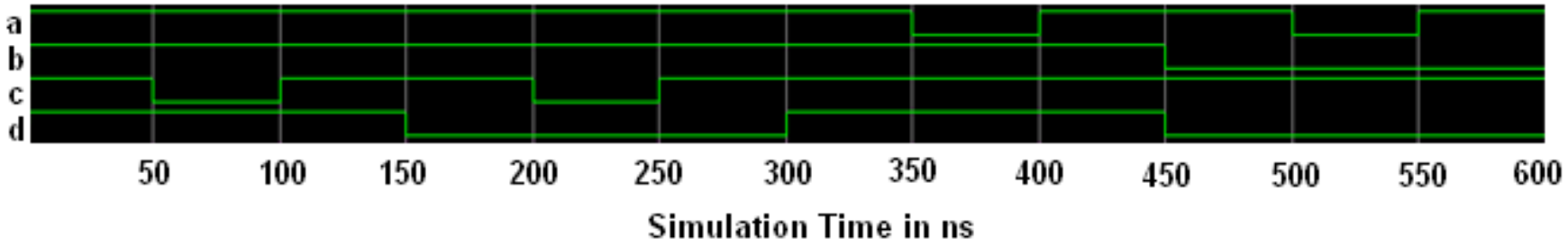
- 1-Hazard exists for case where **c** changes (from 0 to 1 or 1 to 0) and **a=1**, **b=1**, **d=1**.
- 1-Hazard exists for case where **c** changes (from 0 to 1 or 1 to 0) and **a=1**, **b=1**, **d=0**.
- 1-Hazard exists for case where **a** changes (from 0 to 1 or 1 to 0) and **b=1**, **c=1**, **d=1**.



Assuming 1 ns rise/fall times for each gate
(AND, OR, and NOT gates)

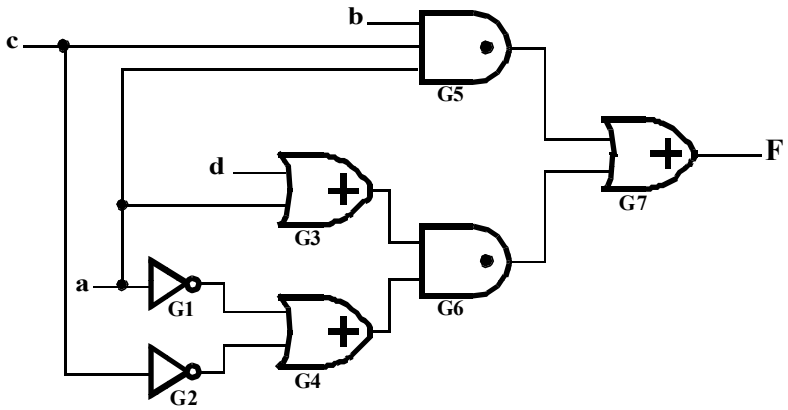
- 0-Hazard exists for case where **a** changes (from 0 to 1 or 1 to 0) and **b=0**, **c=1**, **d=0**.

force a 1 0 ns, 0 350 ns, 1 400 ns, 0 500 ns, 1 550 ns
force b 1 0 ns, 0 450 ns
force c 1 0 ns, 0 50 ns, 1 100 ns, 0 200 ns, 1 250 ns
force d 1 0 ns, 0 150 ns,1 300 ns, 0 450 ns

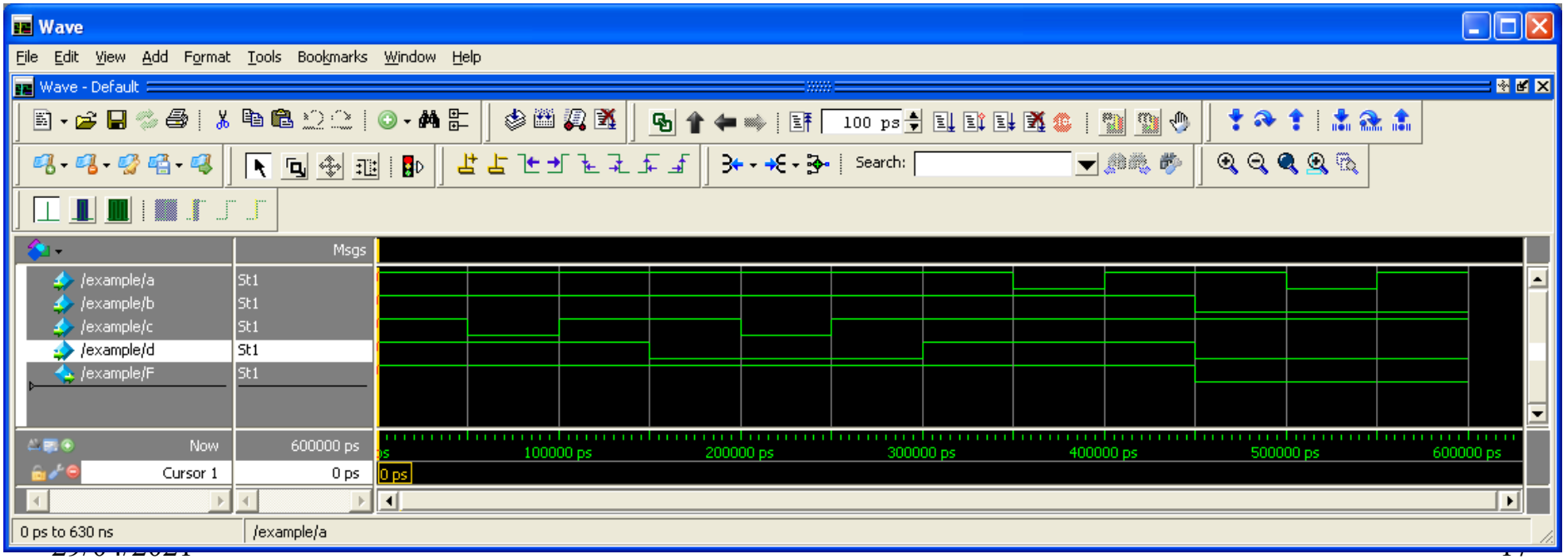


RTL Simulation (zero delay)

```
`timescale 1ns/100ps
a=1; b=1; c=1; d=1;
#50 c=0; #50 c=1;
#50 d=0; #50 c=0; #50 c=1; #50 d=1;
#50 a=0; #50 a=1;
#50 b=0; d=0;
#50 a=0; #50 a=1;
run 600 ns
```

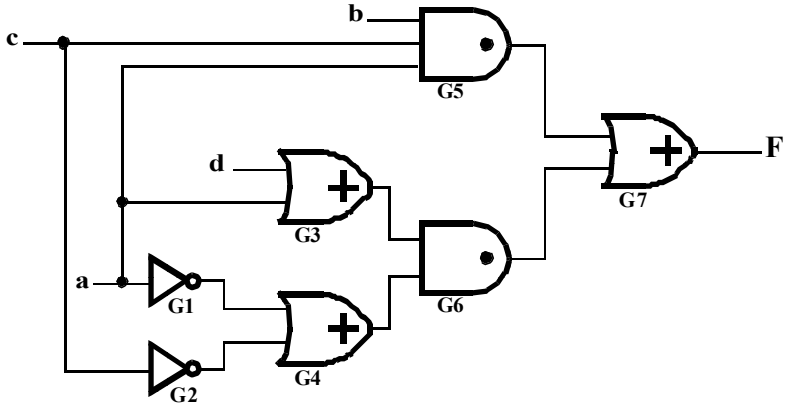


Assuming 1 ns rise/fall times for each gate
(AND, OR, and NOT gates)

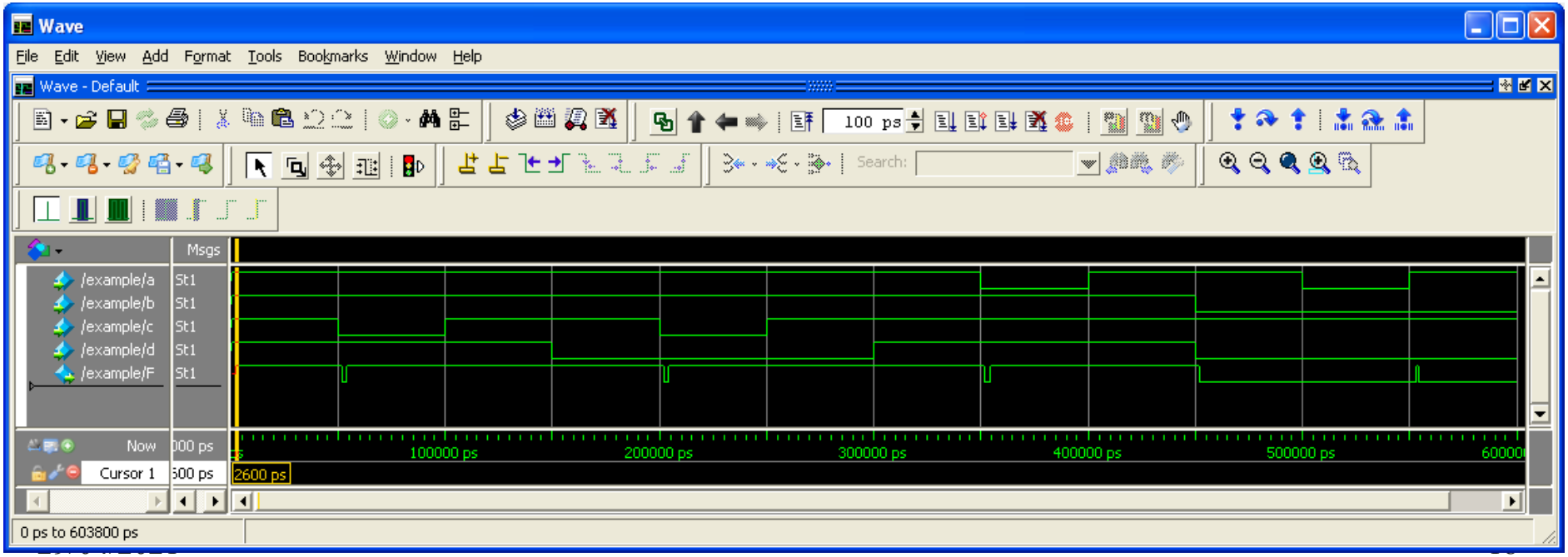


Static Timing Simulation (1ns Delay each gate)

```
`timescale 1ns/100ps
a=1; b=1; c=1; d=1;
#50 c=0; #50 c=1;
#50 d=0; #50 c=0; #50 c=1; #50 d=1;
#50 a=0; #50 a=1;
#50 b=0; d=0;
#50 a=0; #50 a=1;
run 600 ns
```



Assuming 1 ns rise/fall times for each gate
(AND, OR, and NOT gates)



Static Timing Simulation (1ns Delay each gate)

- In this case we were able to see the effect of all four hazards!
  - Not always the case – hazards may cause a glitch but whether or not one occurs depends upon the actual timing values.
- There is automated help in ModelSim™ to allow for hazard checking.

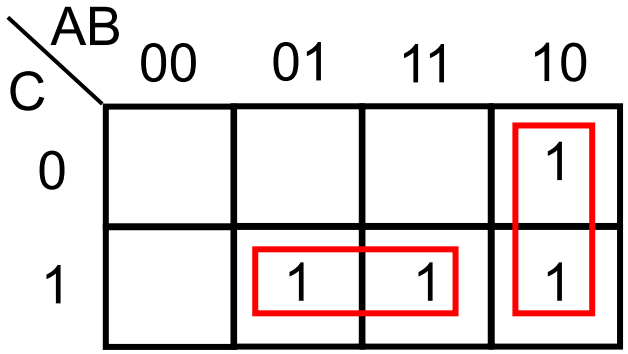
Design of Hazard Free Networks

To design a network that is free of static and dynamic hazards, the following procedure may be used:

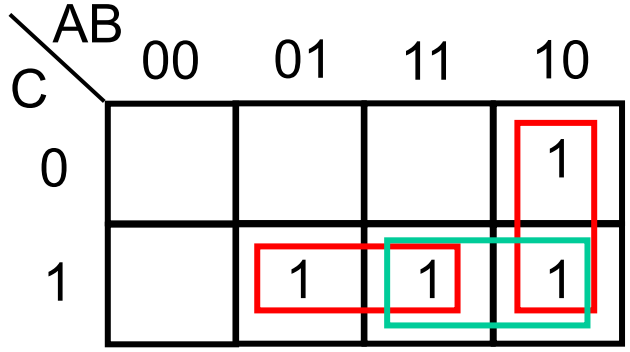
1. Find a sum-of-products expression (F^t) for the output in which every pair of adjacent 1s is covered by a 1-term. (The sum of all prime implicants will always satisfy this condition.) A two-level AND-OR network based on this F^t will be free of 1-, 0-, and dynamic hazards.
2. If a different form of network is desired, manipulate F^t to the desired form by simple factoring, DeMorgan's laws, etc. Treat each x_i and x_i' as independent variables to prevent introduction of hazards.

Alternatively, you can start with a product-of-sums expression in which every pair of adjacent 0s is covered by a 0-term.

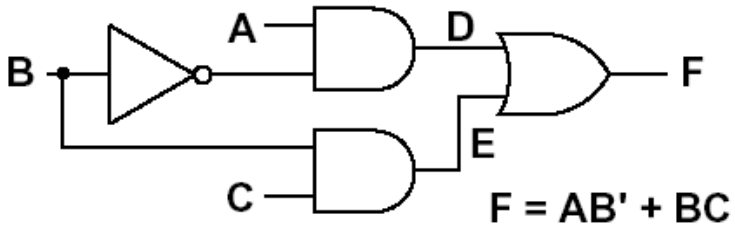
Hazards in Combinational Circuits

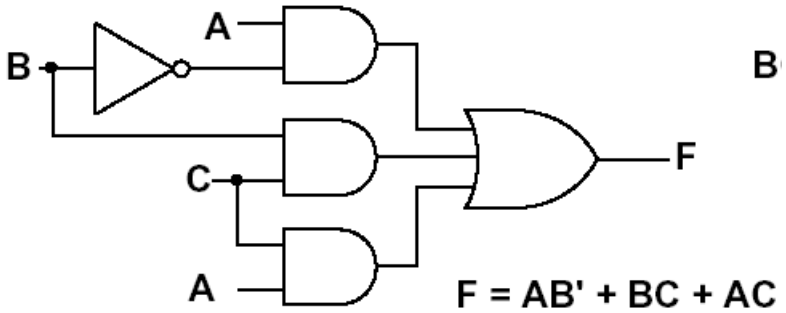


$f = AB' + BC$



$f = AB' + BC + AC$





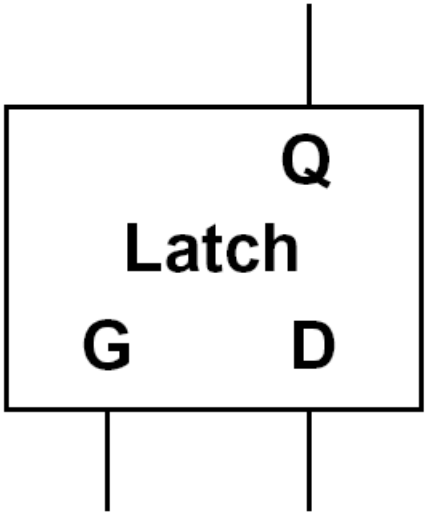
To avoid hazards:
every par of adjacent 1s should be covered by a 1-term

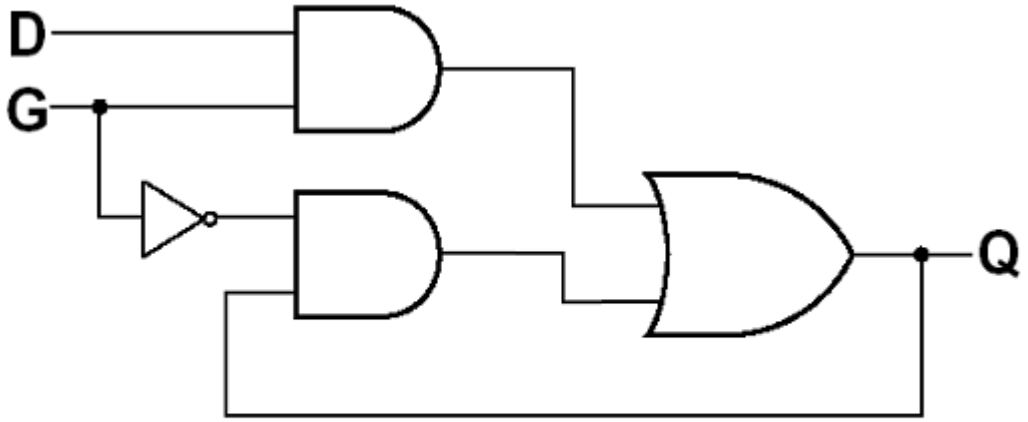
Hazards in Combinational Circuits

Why do we care about hazards?

- Combinational networks
  - don't care – the network will function correctly
- Synchronous sequential networks
  - don't care - the input signals must be stable within setup and hold time of flip-flops
- Asynchronous sequential networks
  - hazards can cause the network to enter an incorrect state
  - circuitry that generates the next-state variables must be hazard-free
- Dynamic Power consumption is proportional to the number of transitions

Transparent D Latch with Hazard

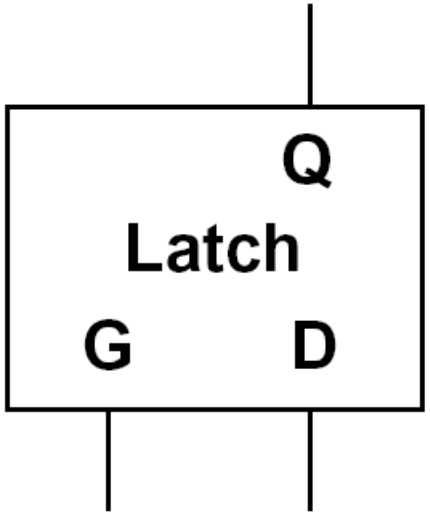


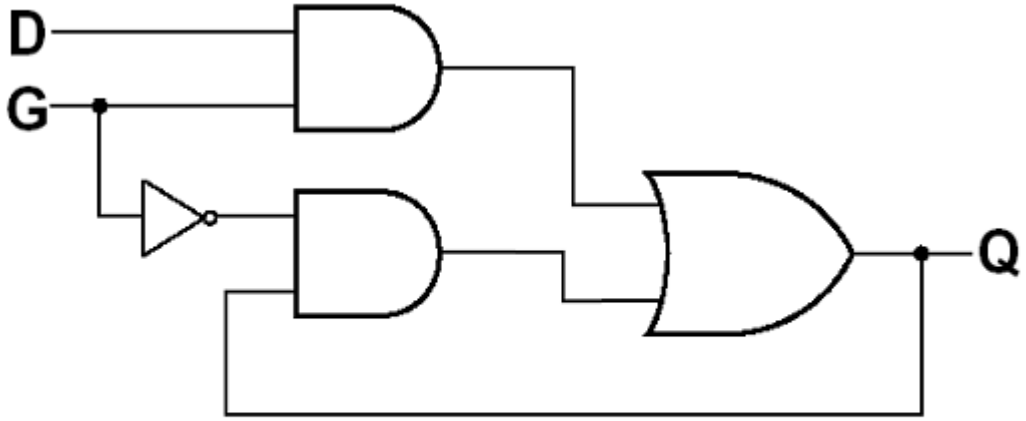


$Q^+ = DG + G'Q$

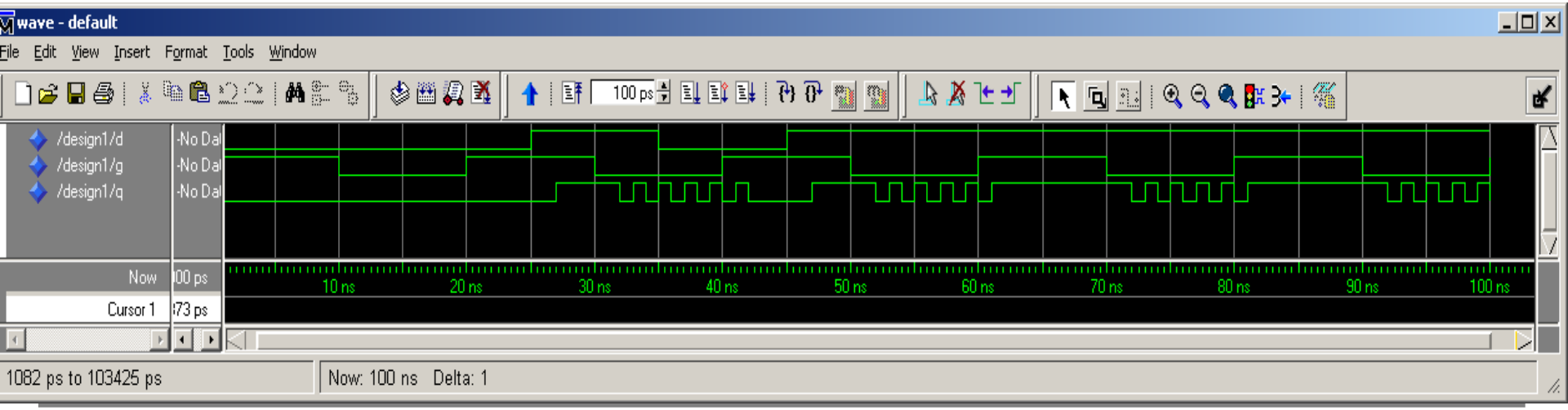
DG \ Q	0	1
00		
01		
11		
10		

Transparent D Latch with Hazard (1 ns Delay all gates)

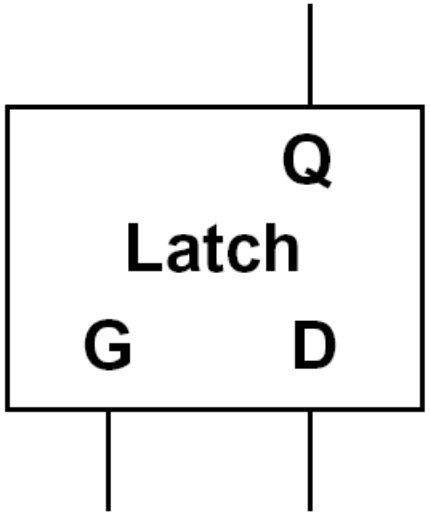


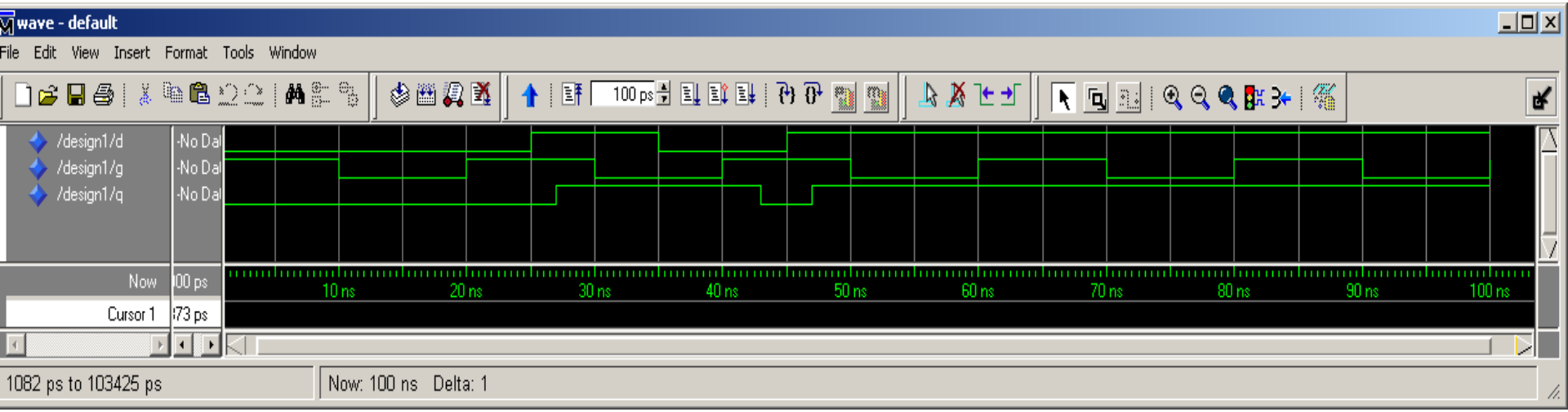


$Q^+ = DG + G'Q$



Transparent D Latch (hazard removed) (1 ns Delay all gates)





Hazards In a FPGA-based Design

- Hazards can appear in any combinational logic section of designs that are implemented in reconfigurable logic devices such as your Altera Cyclone II device on the DE-2 board
- In such devices all combinational logic is implemented using an interconnection of 1, 2, and 4 input ROM type lookup tables.
- The wiring delay between lookup table and the uneven propagation delay of the lookup tables themselves can result in different paths for signals taken from input to output that have different propagation delays (just as in the case of multi-gate-level logic).

Hazards In a FPGA-based Design

- Analysis of hazards in programmable logic is difficult and beyond the scope of this class.
- Hazards are not a problem, though, if there is time to reach a steady state before the final value is stored in a flip-flop (synchronous design techniques).
- Hazards though can be very serious issue if we use combinational control logic to gate the clock!
  - (we will discuss this a bit later)