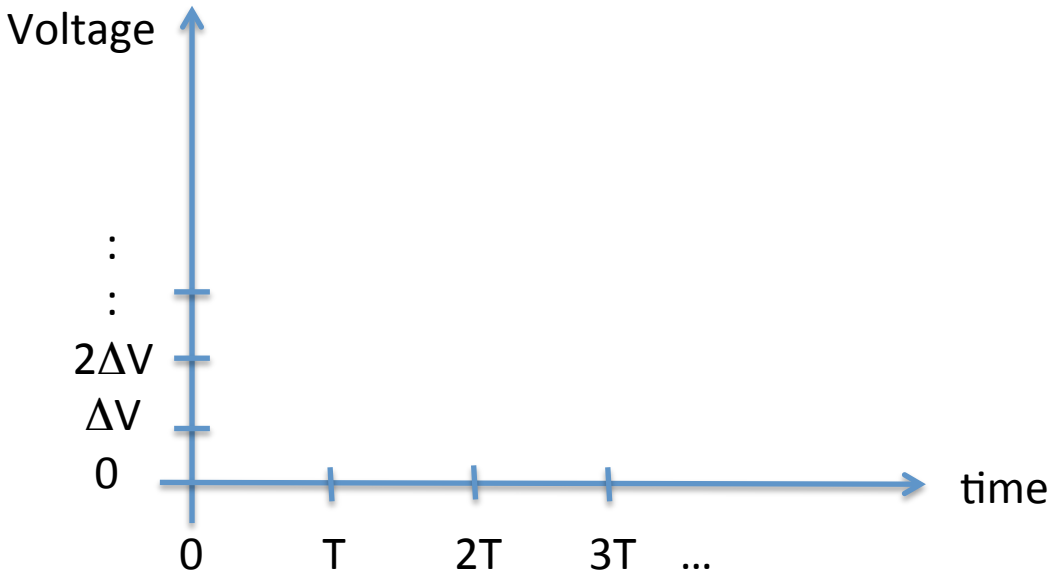
GONZAGA UNIVERSITY

Digital Systems: Quick Review

GONZAGA UNIVERSITY

Digital System

- Transform signals that can be abstracted as discrete in range and domain



GONZAGA UNIVERSITY

Analog & (~~vs.~~) Digital

- Analog Circuits Advantages
  - Require less devices
  - Better to deal with low signal amplitudes
  - Better to deal with high frequencies
- Digital Circuits Advantages
  - More "adaptable" (e.g. microprocessor)
  - Design can be done at a more abstract level
  - Better economic

GONZAGA UNIVERSITY

Types of Digital Circuits

- Combinational
  The value of the outputs at any time t depends only on the combination of the values applied at the inputs at time t (the system has no memory)
- Sequential
  The value of the outputs at any time t depends not only on the values applied at the inputs at time t but also on the past sequence of inputs that have been applied to it
  (system with memory)

C.L.: effect of gate delays

- The analysis of combinational circuits ignoring delays can predict only the steady-state behavior of a circuits. That is they predict a circuit's output as a function of its inputs under the assumption that the inputs have been stable for a long time, relative to the delays into the circuit's electronics.

- Because of circuit delays, the transient behavior of a combinational logic circuit may differ from what is predicted by a steady-state analysis.

- In particular a circuit's output may produce a short pulse (often called a glitch) at a time when steady state analysis predicts that the output should not change.

GONZAGA UNIVERSITY

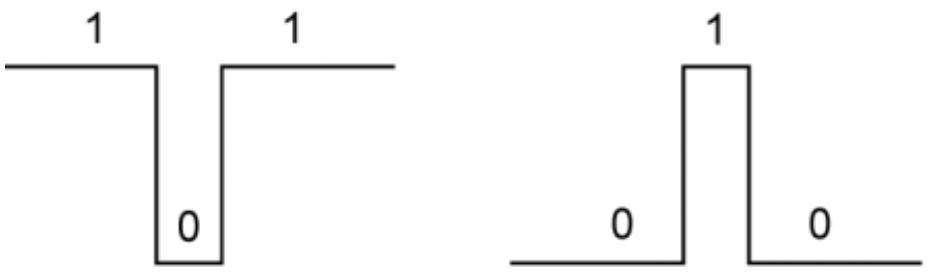
C.L.: Timing Hazards

- A glitch is an unwanted pulse at the output of a combinational logic network – a momentary change in an output that should not have changed.
- A circuit with the potential for a glitch is said to have a hazard.
- An hazard is something intrinsic about a circuit; a circuit with hazard may or may not have a glitch depending on input patterns and the electric characteristics of the circuit.

When do C.L. circuits have Hazards ?

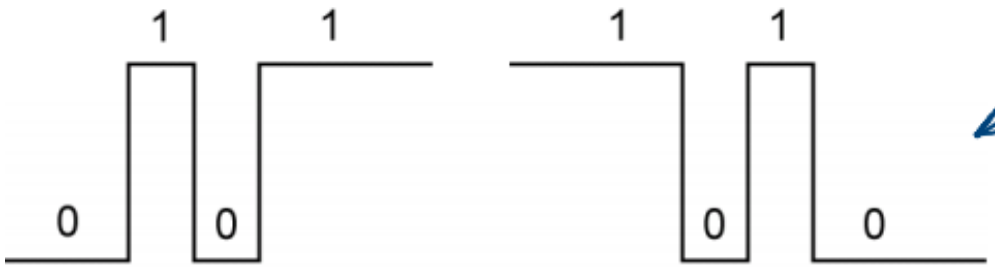
- Hazards are potential unwanted transients that occur in the output when different paths from input to output have different propagation delays

Types of Hazards (on an output)



(a) Static 1-hazard (b) Static 0-hazard

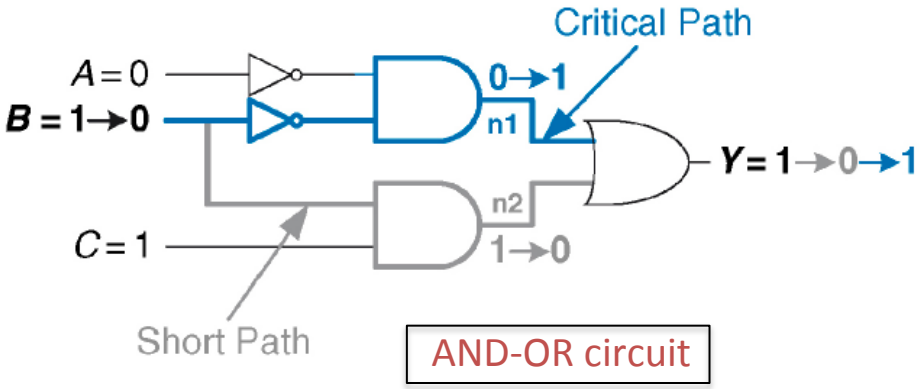
The output undergoes a momentary transition when it is expected to remain unchanged



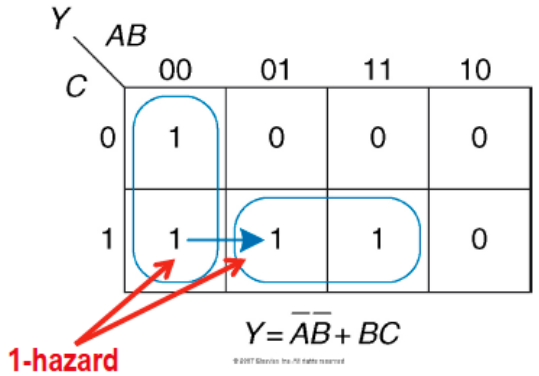
(c) Dynamic hazards

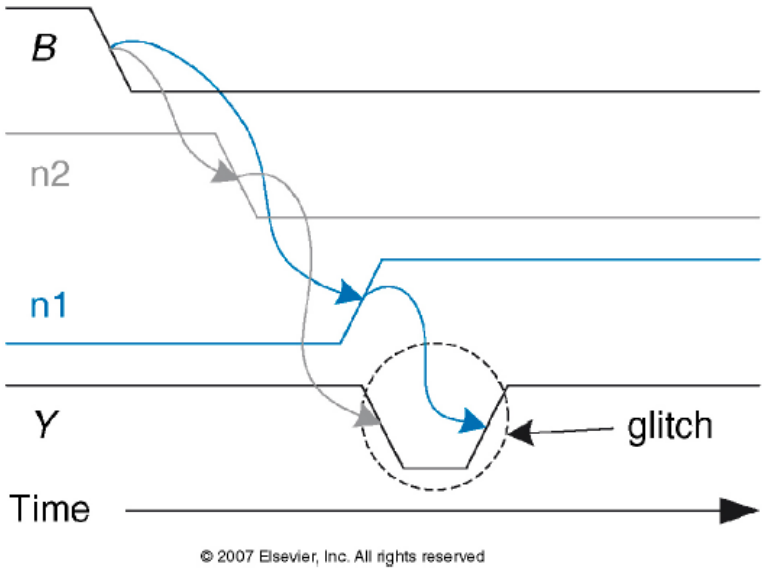
The output changes multiple times as the result of a single input transition

Detection of Static 1-Hazards



AND-OR circuit



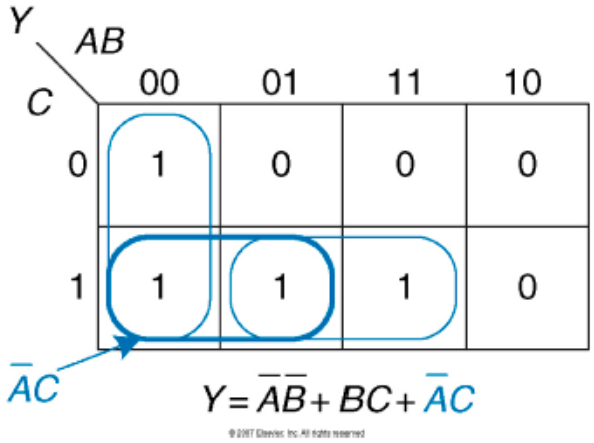


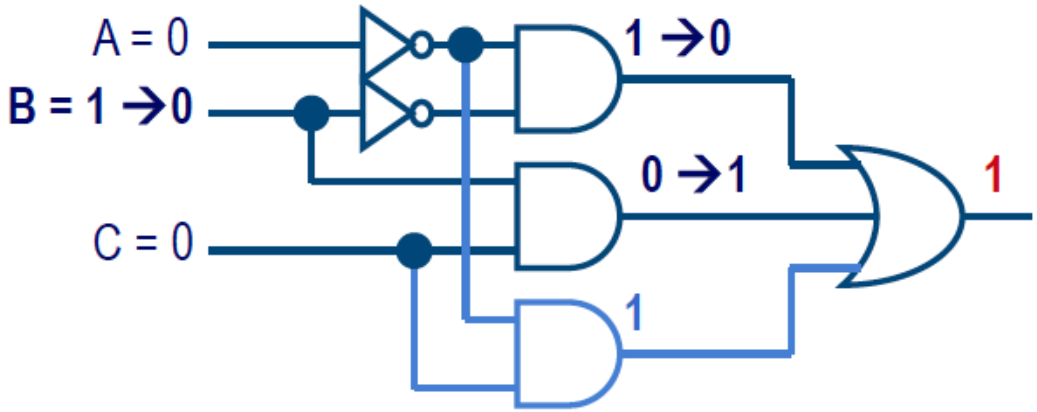
Basically because of gate delays for a moment when B changes is not true that $B+\overline{B}=1$

Temporary violation of complementary law.

Removing Static Hazards

- The fundamental strategy for eliminating an hazard is to add redundant prime implicants (extra prime implicants won't change F, but can cause F to be asserted independently of the change to the input that cause the hazard).



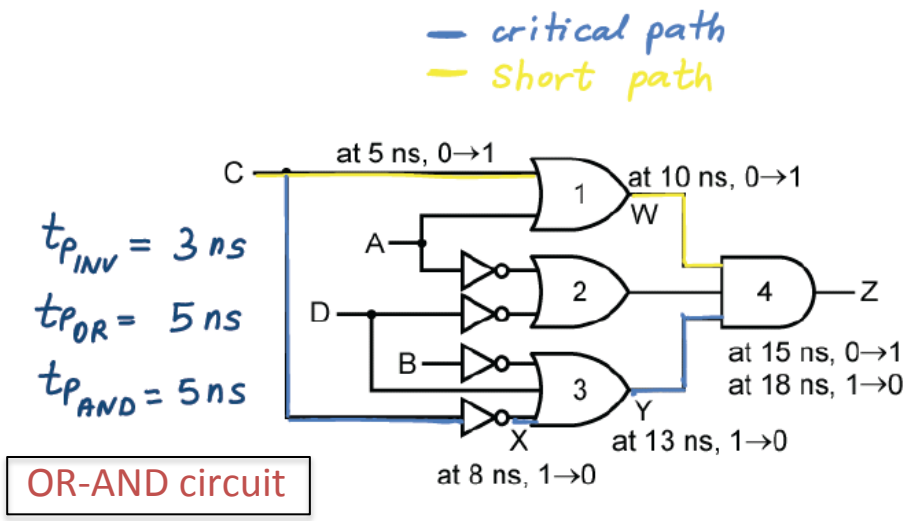


circuit with hazard removed

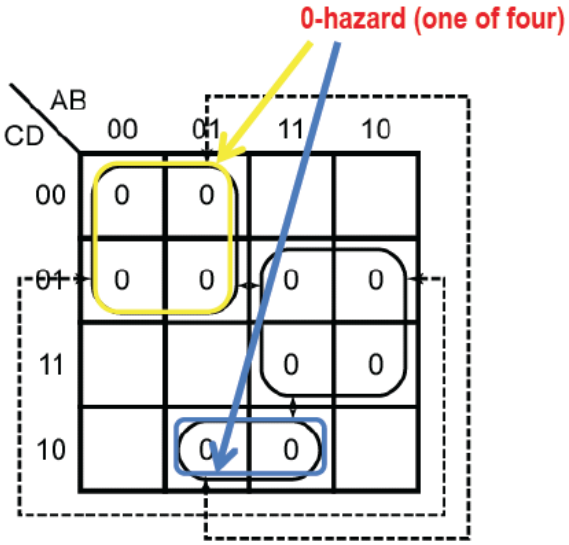
GONZAGA UNIVERSITY

Detection of static 0-Hazards

example: A=0, B=1, D=0, C=0→1



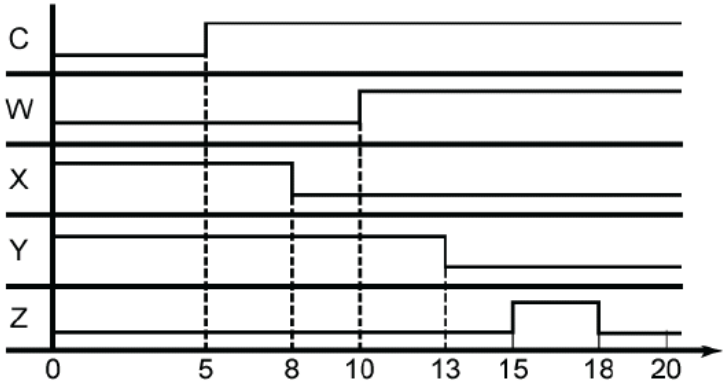
(a) Circuit with a static 0-hazard



(b) Karnaugh map for circuit of (a)

Basically because of gate delays for a moment when C changes is not true that $C \cdot \bar{C} = 0$

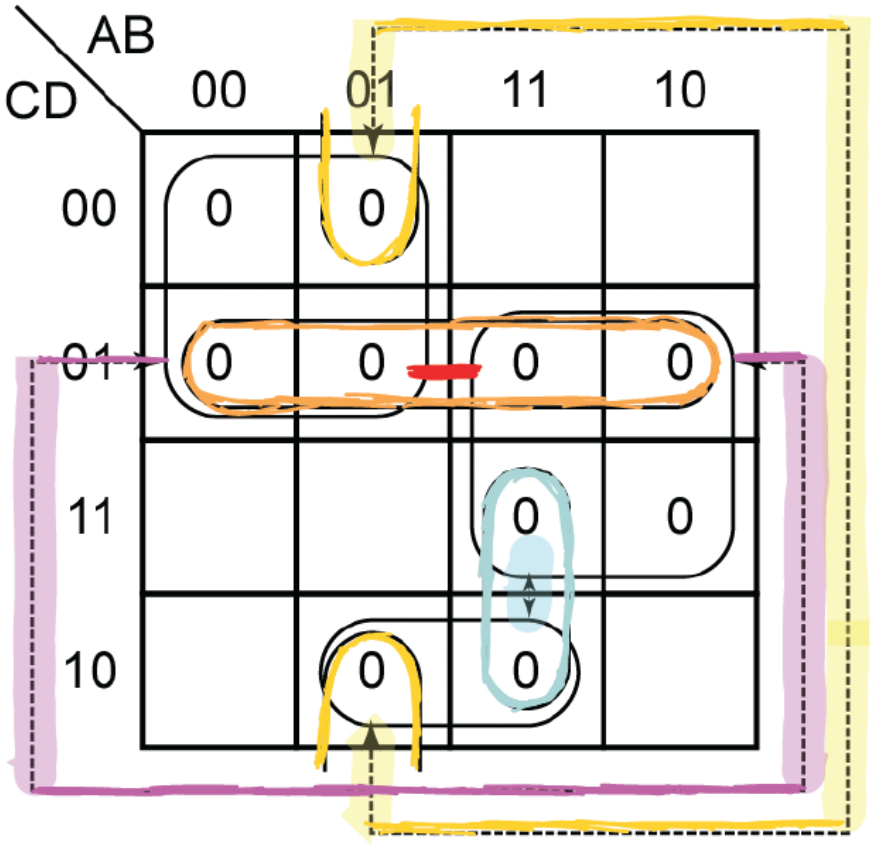
Temporary violation of complementary law



(c) Timing diagram illustrating 0-hazard of (a)

$F = (A+C) \cdot (\bar{A}+\bar{D}) \cdot (\bar{B}+\bar{C}+D)$

Removing Static Hazards



The circuit has 4 potential sources of hazards that must be removed

$$F = (A+C)\cdot(\bar{A}+\bar{D})\cdot(\bar{B}+\bar{C}+D)\cdot(C+\bar{D})\cdot(A+\bar{B}+D)\cdot(\bar{A}+\bar{B}+\bar{C})$$

In theory: we need to focus only on one type of hazard !

GONZAGA UNIVERSITY

- A **properly designed two level AND-OR circuit** has **no static 0-hazards**. A static 0-hazard would exist only if both a variable and its complement were connected to the same AND gate, which would be a nonsense ($A \cdot A' \cdot X=0$)

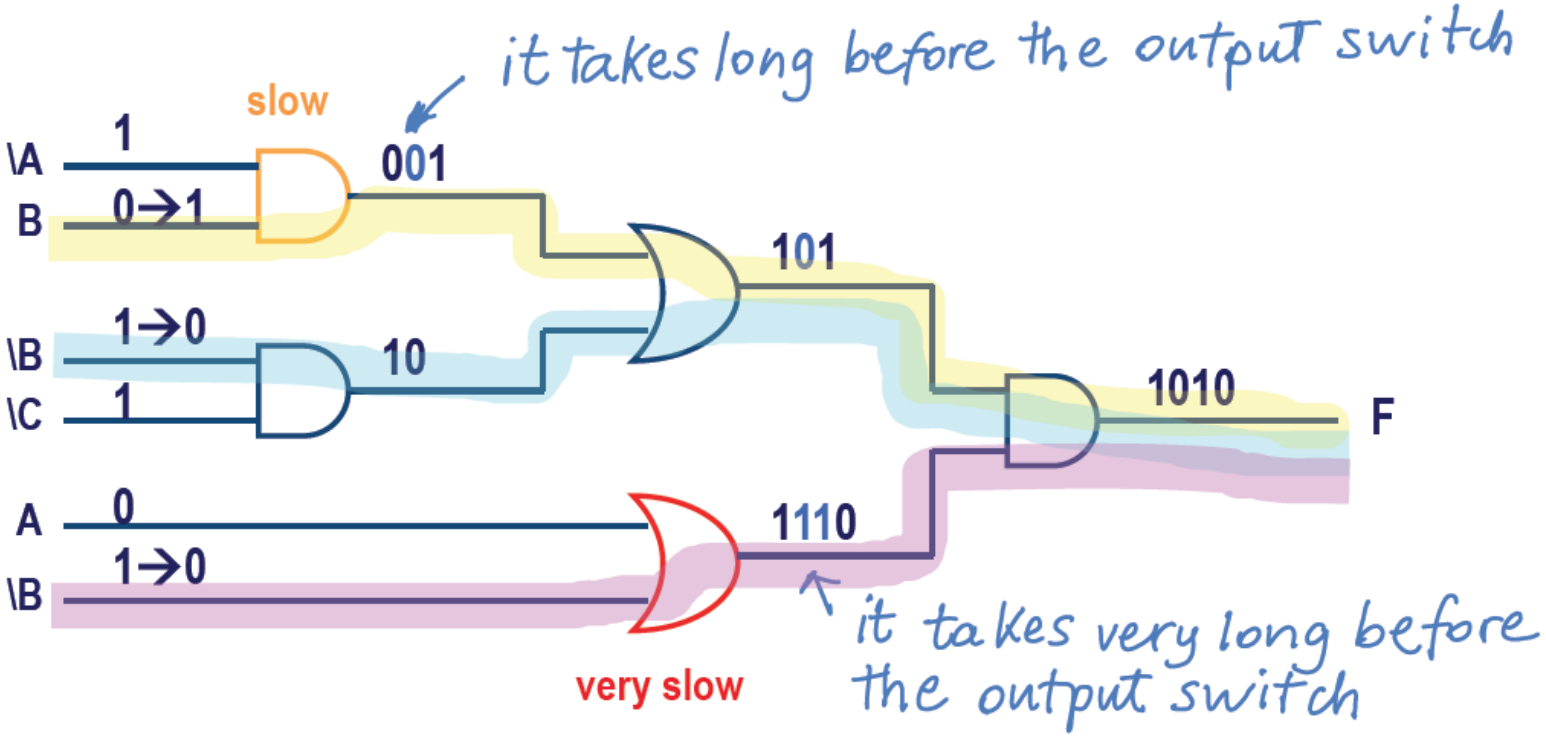
- A **properly designed two level OR-AND circuit** has **no static 1-hazards**. A static 1-hazard would exist only if both a variable and its complement were connected to the same OR gate, which would be a nonsense ($A+A'+X=1$)

GONZAGA UNIVERSITY

Dynamic Hazards

- If there are 3 or more paths from an input or its complement to the output the circuit has the potential for a dynamic hazard.
- Three or more paths from an input or its complement to the output can exist only in a multi-level networks. This means that dynamic hazards do not occur in a properly designed two level AND-OR or OR-AND network.
- Analysis and elimination of dynamic hazards is a rather complicated process.
- If you need a hazard free network, it is best to use a 2-level network and use the techniques shown earlier to eliminate the static hazards.

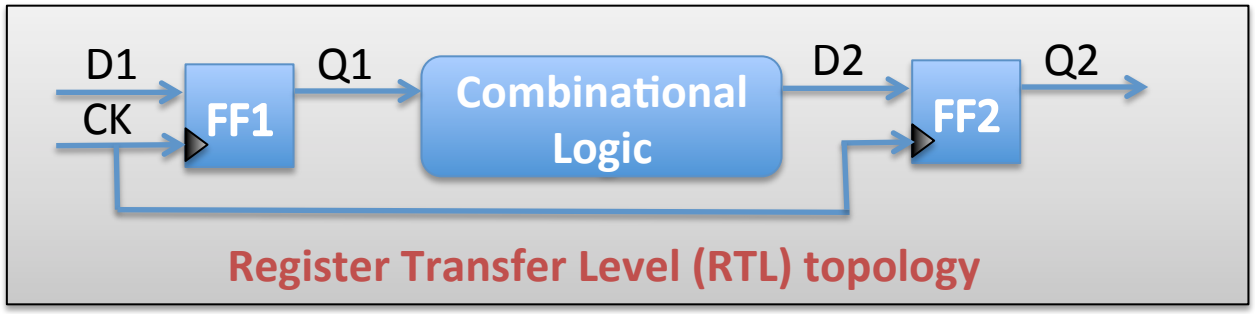
Dynamic Hazard Example



All gates are ideal (very fast) except a slow AND gate and a very slow OR gate

Hazard-Free Design

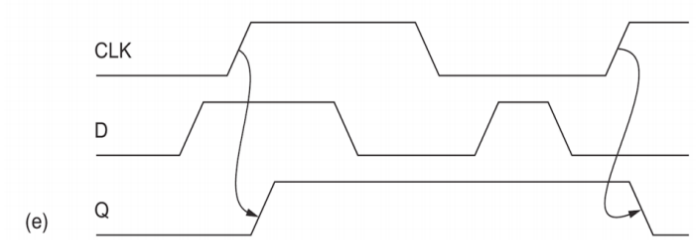
- Best way to deal with hazards:
  structure the design so that you do not have to worry about them !!!
- A well-designed, synchronous digital system is structured so that hazard analysis is not needed

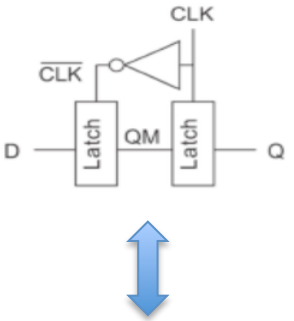


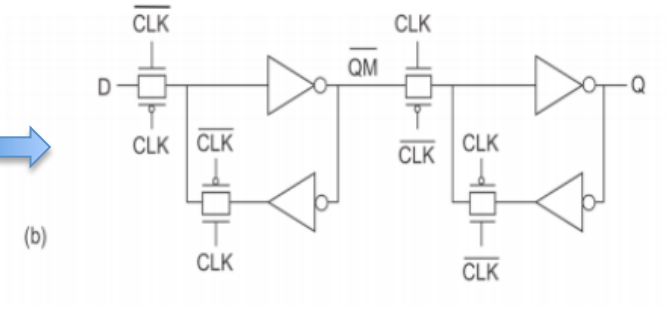
- In a synchronous system, all the inputs to a combinational circuit are changed at a particular time, and the outputs are not looked at until they have time to settle to a steady-state value.

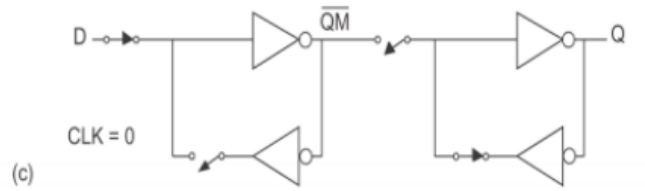
Timing Issues in S.L. circuits

- Set-up Time
- Hold Time
- Clock skew









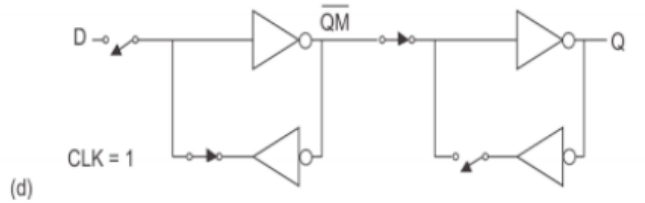
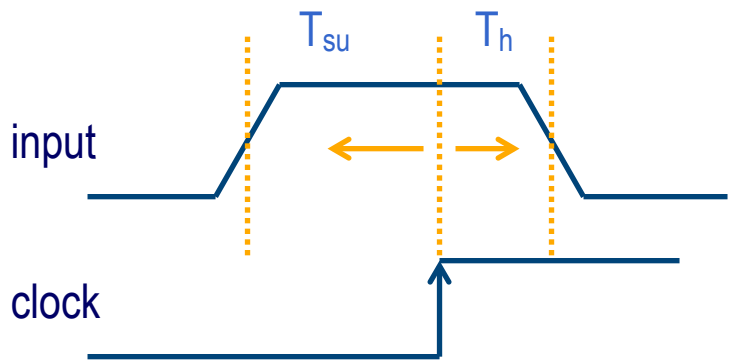


FIG 1.31 CMOS positive-edge-triggered D flip-flop

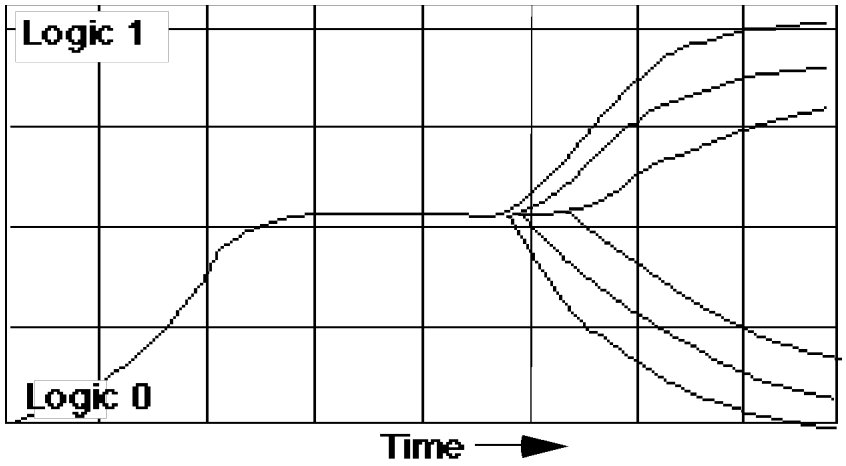
Setup/Hold Time



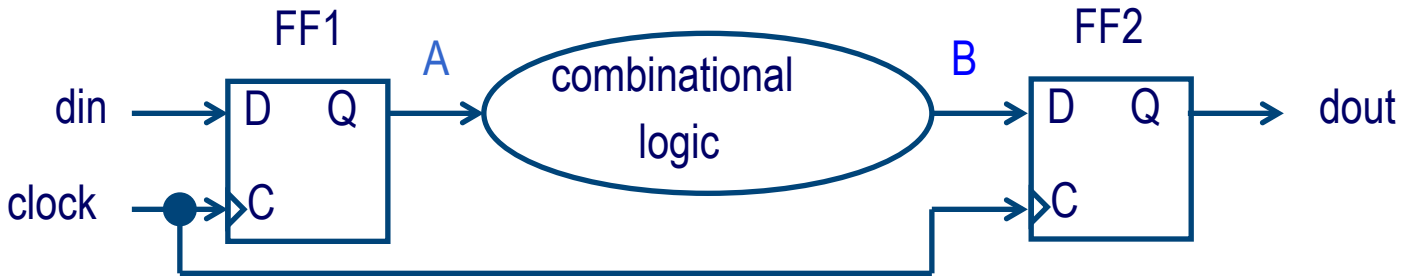
- An input to a flip flop can be validly recognized only if:
  - it is stable before the clocking event for a minimum time interval T_{setup} and
  - it is stable after the clocking event for a minimum time interval T_{hold}

Setup/Hold Time Violation

- It is dangerous to allow input signals to change very close to the sampling event (that is the active clock edge)
- If setup or hold time constraints are not satisfied, the input maybe interpreted as a 1 or a 0 or some unrecognizable value between 0 and 1 (metastable value)



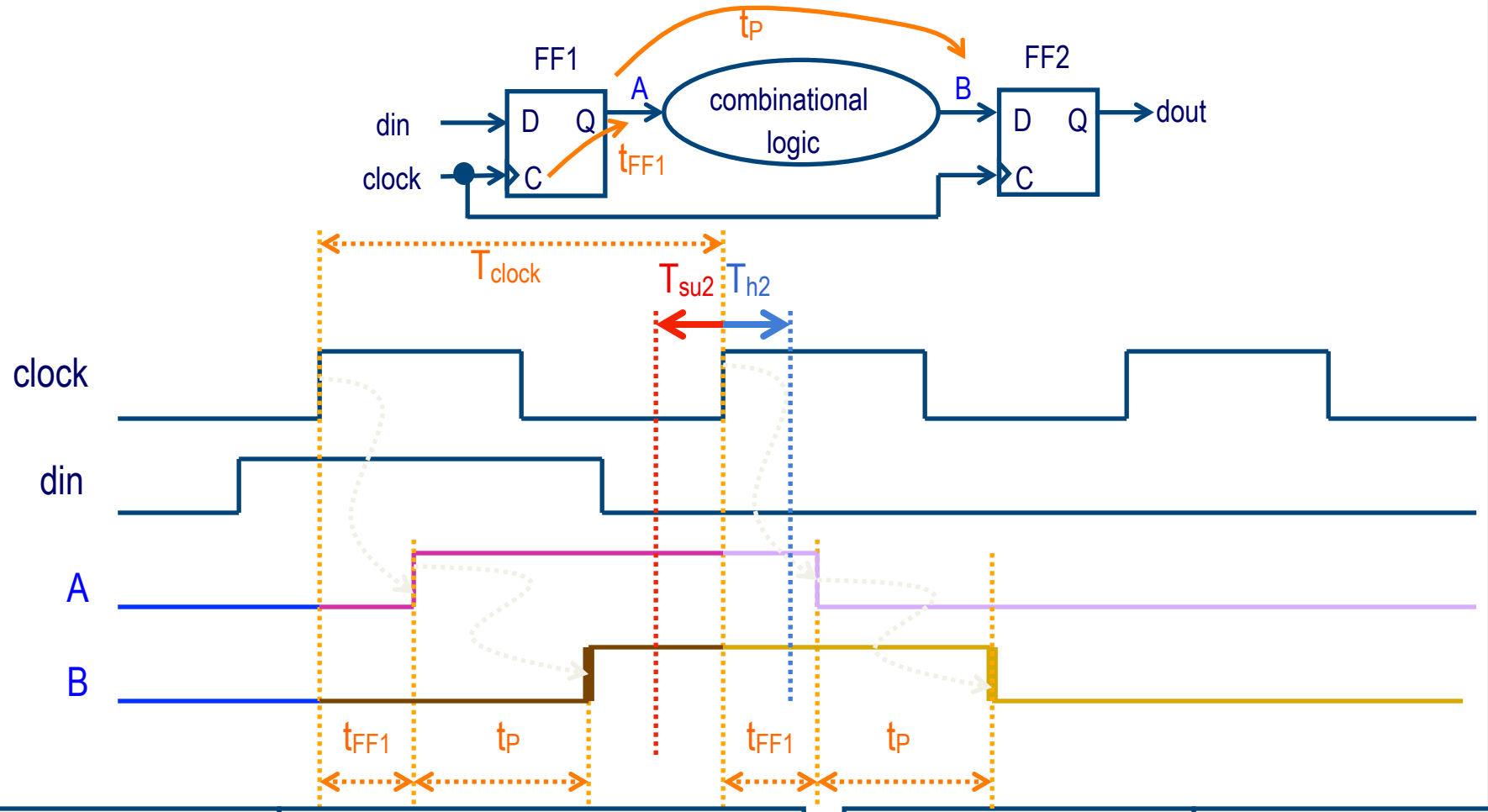
Timing Constraints in S.L. circuits



Let's assume din is applied in a way that satisfies setup and hold time for FF1, and let's examine what happen to FF2

GONZAGA UNIVERSITY

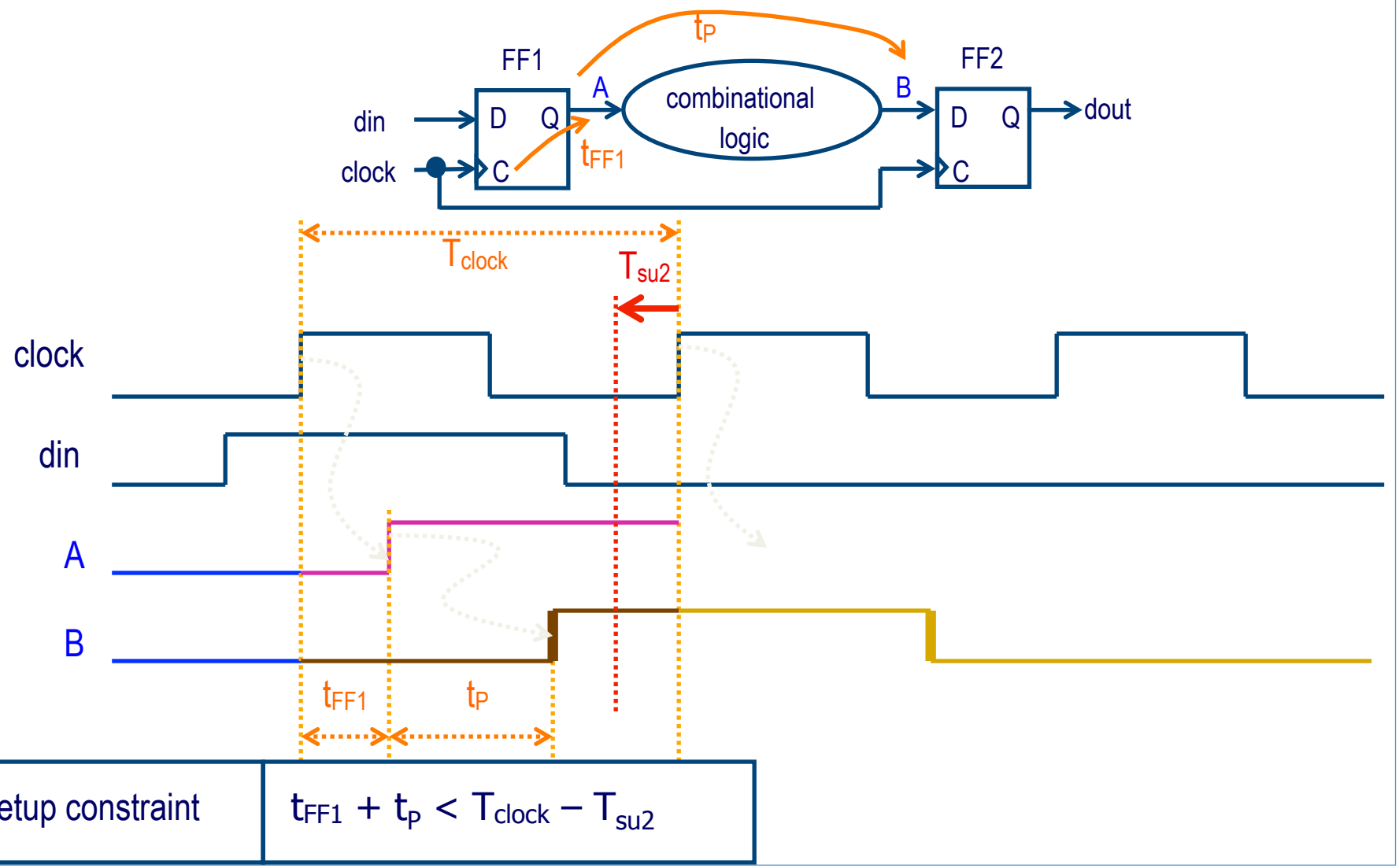
Timing constraints in sequential circuits



Setup constraint	$t_{FF1} + t_P < T_{clock} - T_{su2}$	$t_{FF1} + t_P > T_{h2}$	Hold constraint

GONZAGA UNIVERSITY

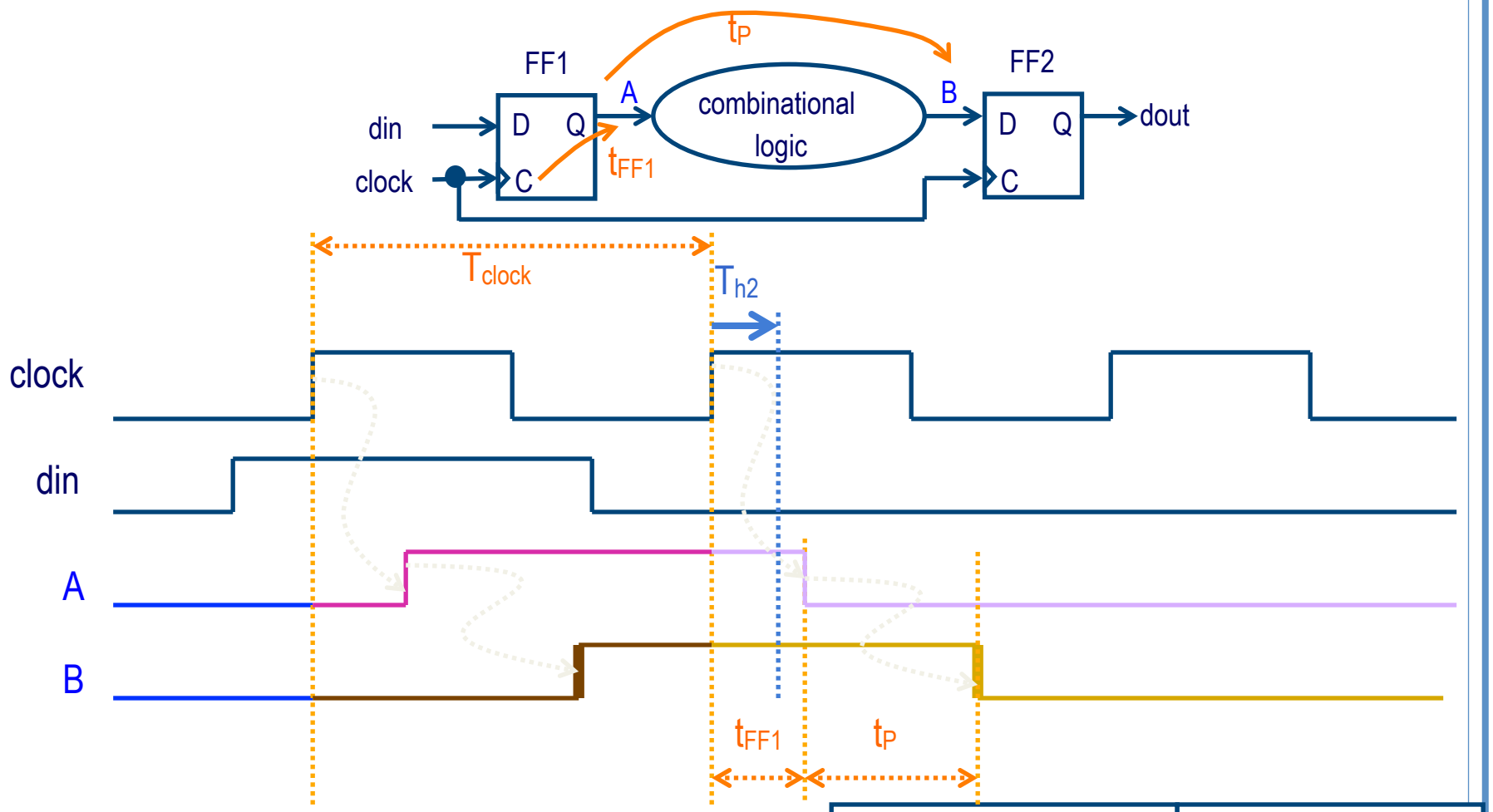
Timing constraints in sequential circuits



Setup constraint	$t_{FF1} + t_P < T_{clock} - T_{su2}$

GONZAGA UNIVERSITY

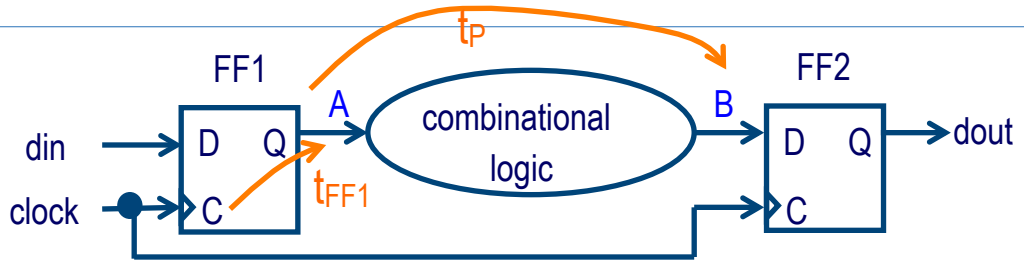
Timing constraints in sequential circuits



$t_{FF1} + t_P > T_{h2}$	Hold constraint

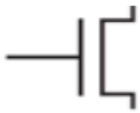
GONZAGA UNIVERSITY

Max/Min delays



Setup constraint	$t_{FF1} + t_P < T_{clock} - T_{su2}$	$t_{FF1} + t_P > T_{h2}$	Hold constraint

- Unfortunately, delays through gates are not constant. Delays change with:
  - Supply Voltage, Temperature, and Manufacturing Process
- **Setup constraint** is more difficult to satisfy when delays are **max** ($V_{DD}\downarrow$, T $\uparrow$, P$\uparrow$)
- **Hold constraint** is more difficult to satisfy when delays are **min** ($V_{DD}\uparrow$, T $\downarrow$, P $\downarrow$)



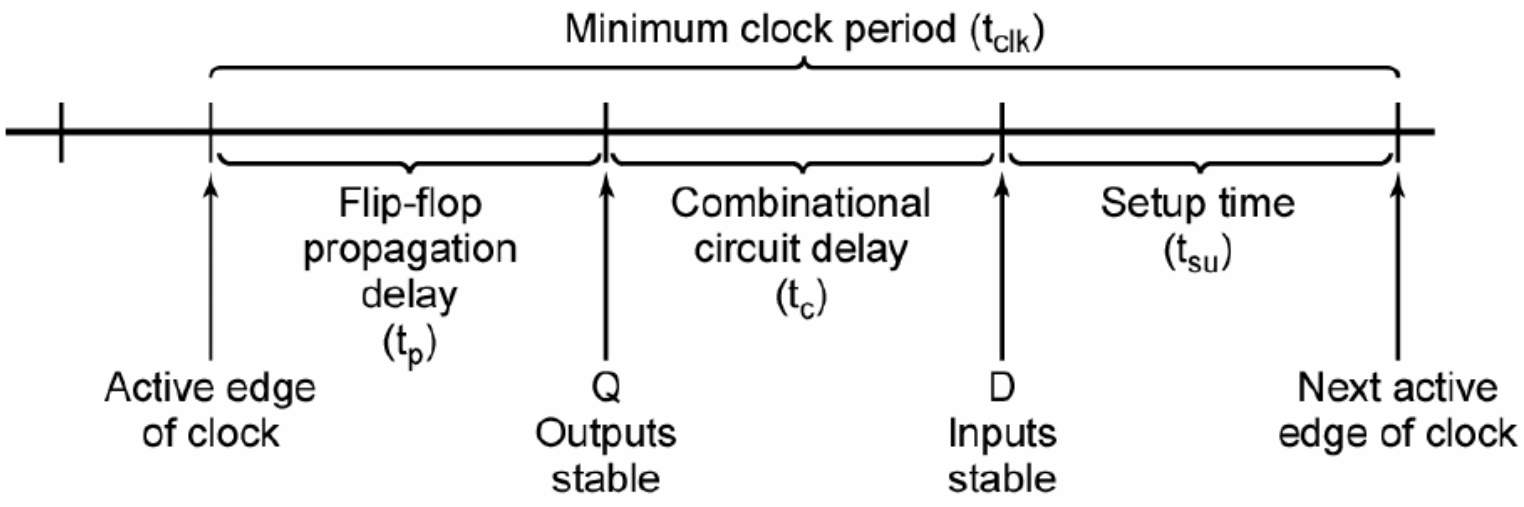
$$Ids = \beta(Vgs - VT)^2$$

$V_{DD}\downarrow \rightarrow V_{gs}\downarrow \rightarrow I_{ds}\downarrow$ $\rightarrow$ it takes longer to charge C_L

T$\uparrow \rightarrow \mu\downarrow \rightarrow \beta\downarrow \rightarrow I_{ds}\downarrow \rightarrow$ it takes longer to charge C_L

Minimum Clock period for a Sequential circuit

max frequency at which the circuit can operate



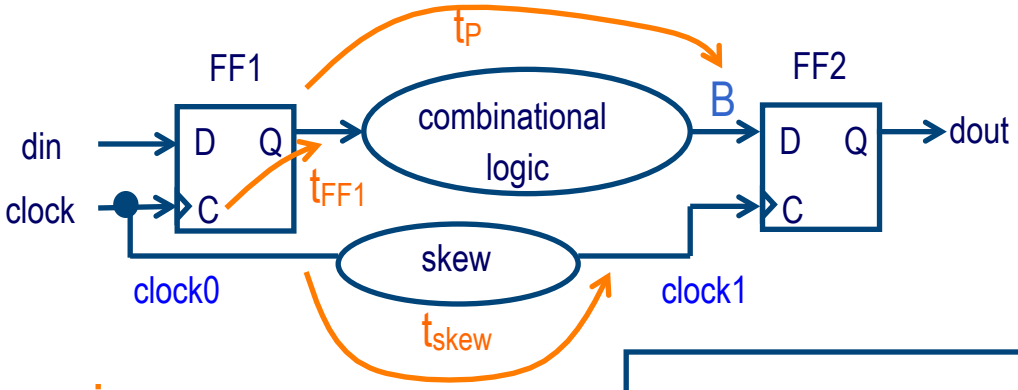
Minimum Clock Period for a Sequential Circuit

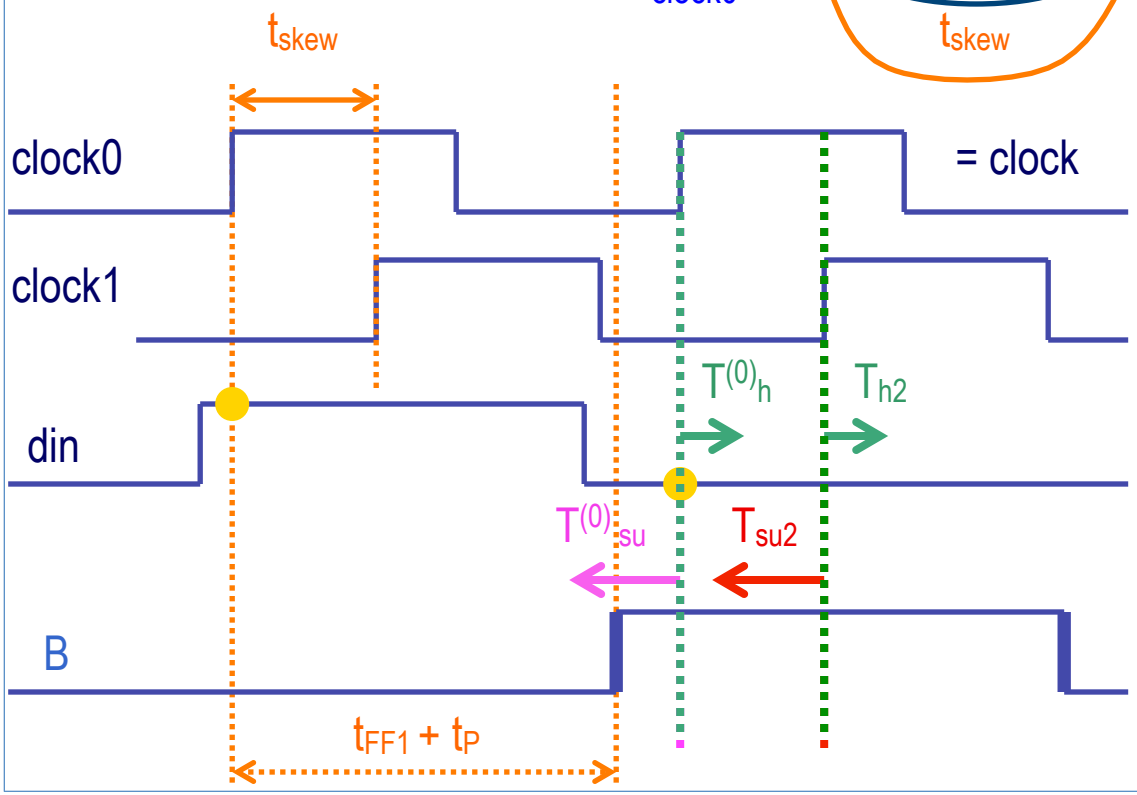
make sure to compute it with max delays

GONZAGA UNIVERSITY

Clock skew

In this example if clock0=clock1 (no skew) setup at FF2 is violated





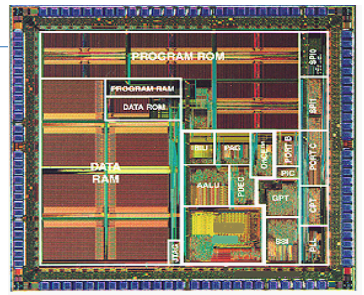
Positive skew makes easier to satisfy setup constraint:

$t_{FF1}+t_P < T_{clock}-T_{su2}+t_{skew}$

Positive skew makes more difficult to satisfy hold constraint:

$t_{FF1} + t_P > T_{h2}+t_{skew}$

Design Abstraction Levels



RTL SW

System Level

RTL

Register Transfer Level (HDL)

Gate Level

Transistor Level

Abstraction

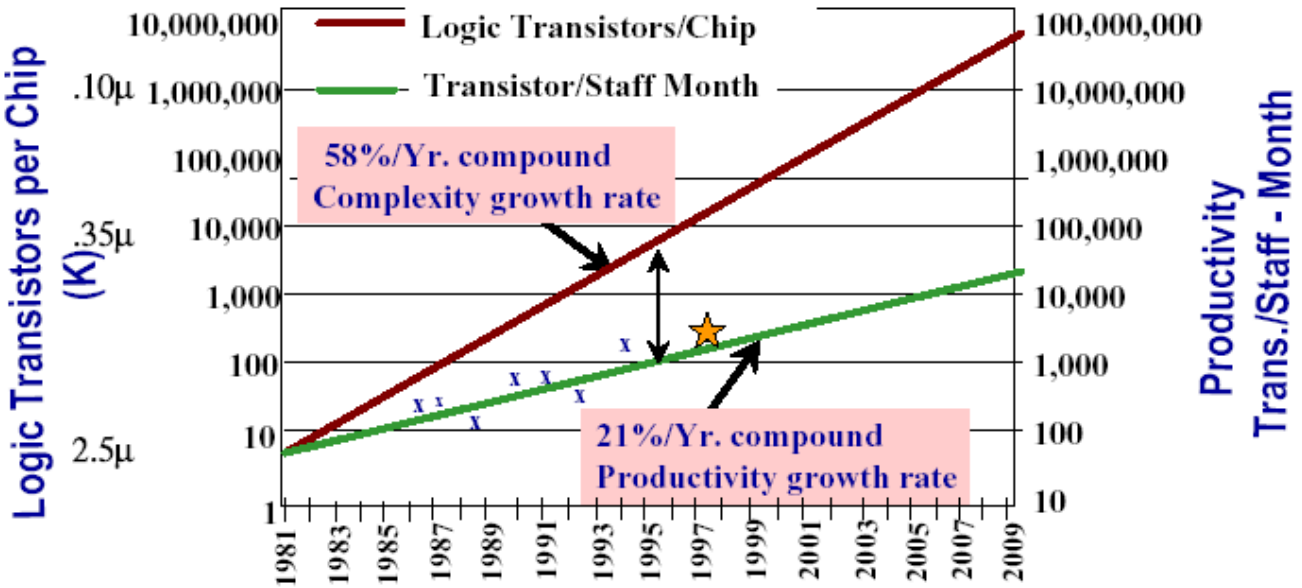
1970 1980 1990 2000+

Moore's Law

GONZAGA UNIVERSITY

- In 1963 Gordon Moore predicted that as a result of continuous miniaturization transistor count would double every 18 months
  - 53% compound annual growth rate over 45 years
  - No other technology has grown so fast so long
  - Transistors become smaller, faster, consume less power, and are cheaper to manufacture

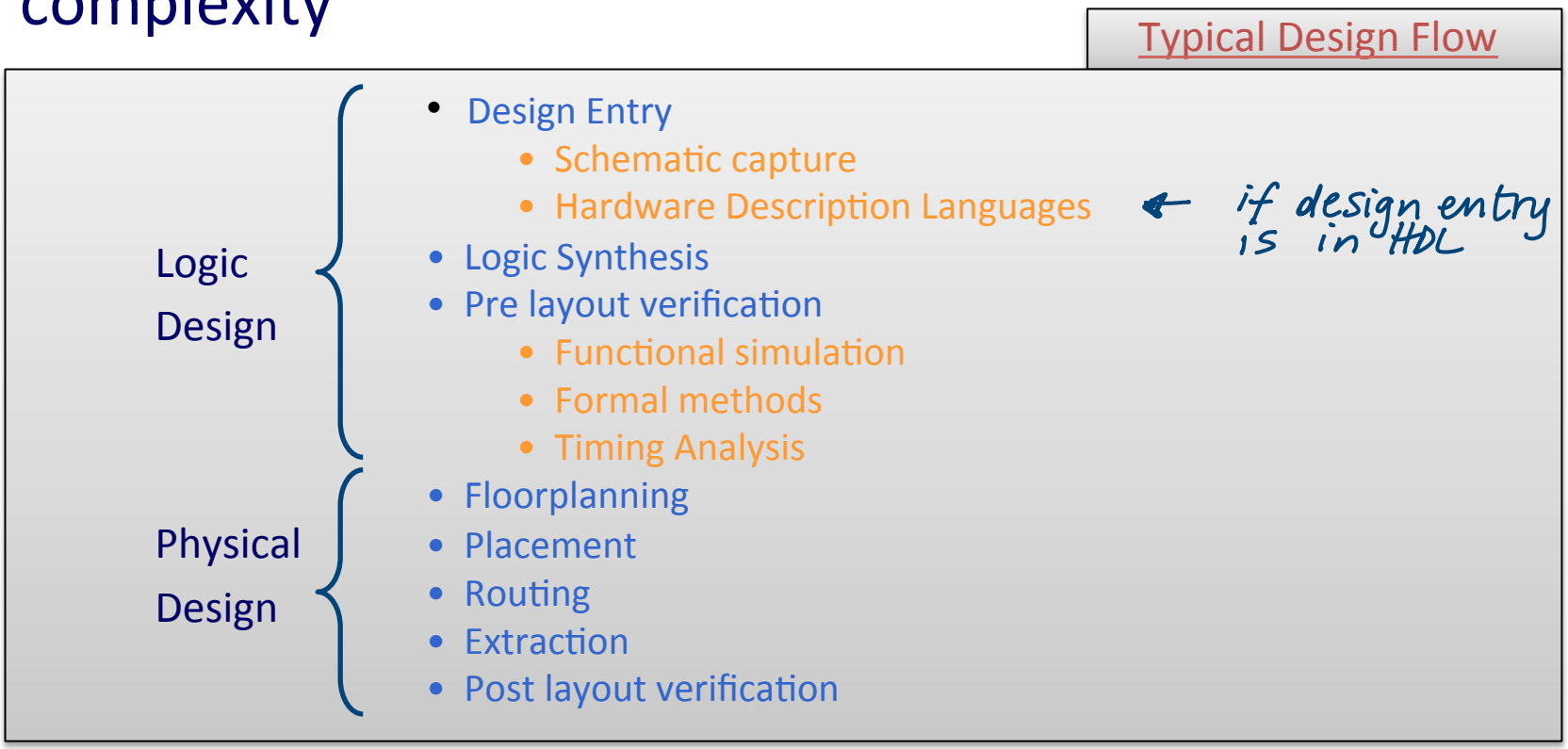
The Design Productivity Gap



GONZAGA UNIVERSITY

CAD Tools

- Designers rely on design automation software tools to seek productivity gains and to cope with increased complexity



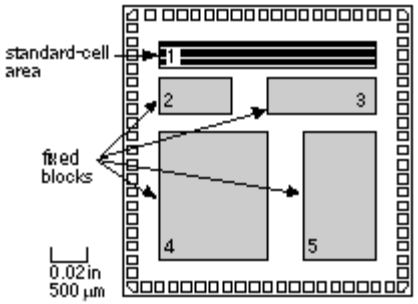
Types of ICs

Design Style

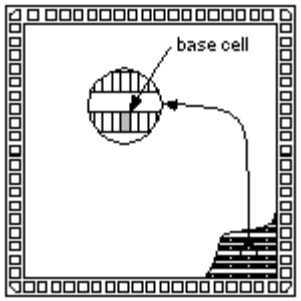
* ASSP
* ASIC

Use

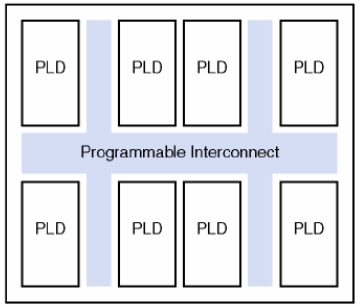
* Full-custom
* Semi-custom
  - Cell Based
  - Gate Arrays
* Programmable
  - CPLD and FPGA



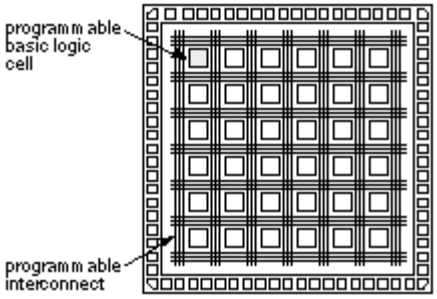
Cell based



Gate Array

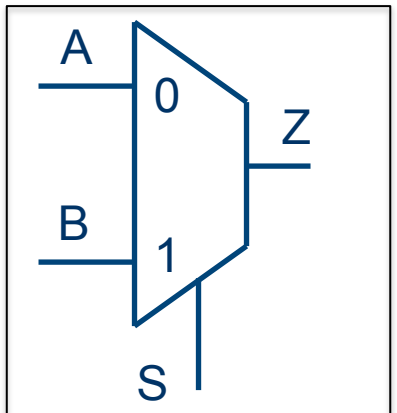


CPLD

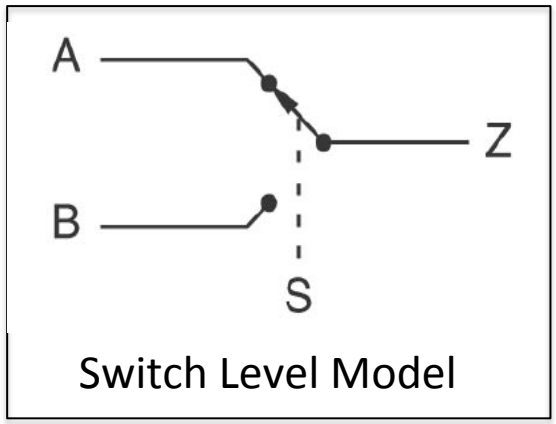


FPGA

Back … to Design Abstraction Levels



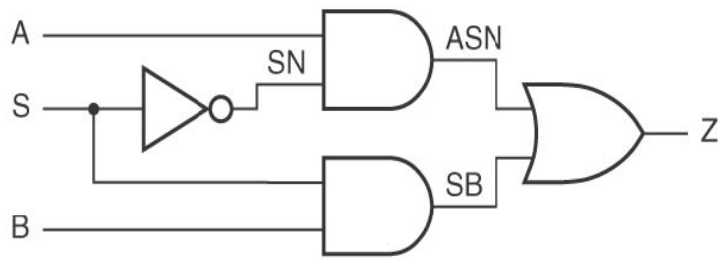
Functional Model



Switch Level Model

$$Z = A \cdot \bar{S} + B \cdot S$$

Logic Equation



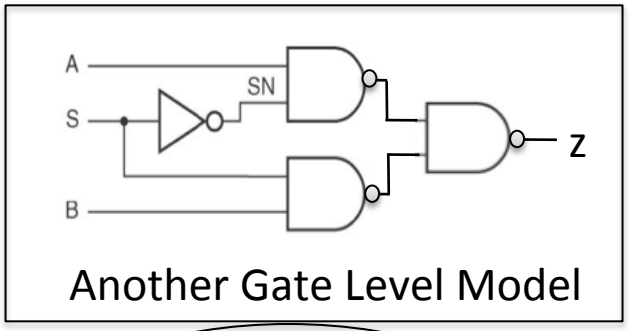
Gate Level Model

Issues:
- Level of Details
- Portability
- Physical mapping

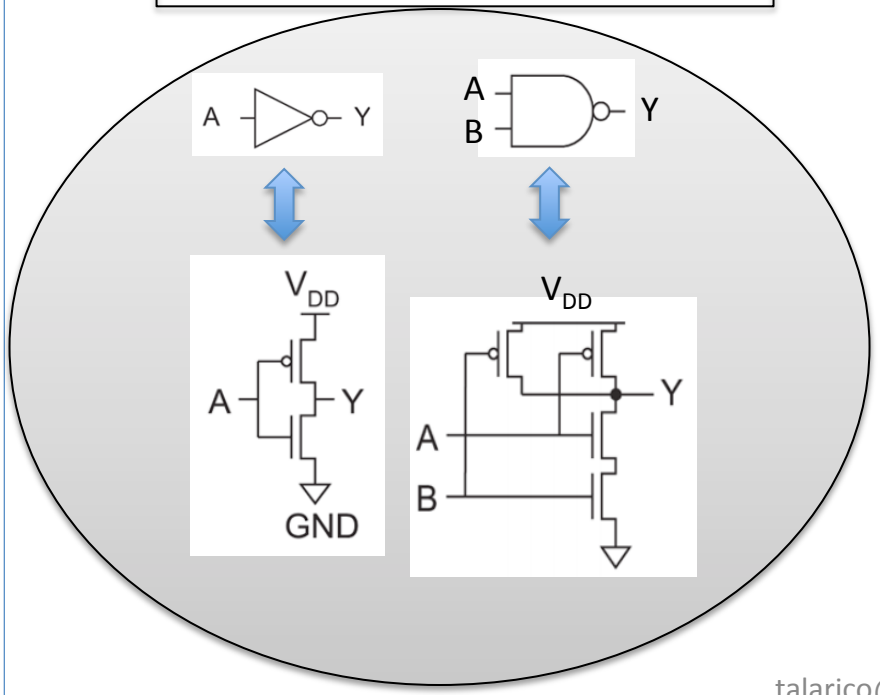
Truth Table

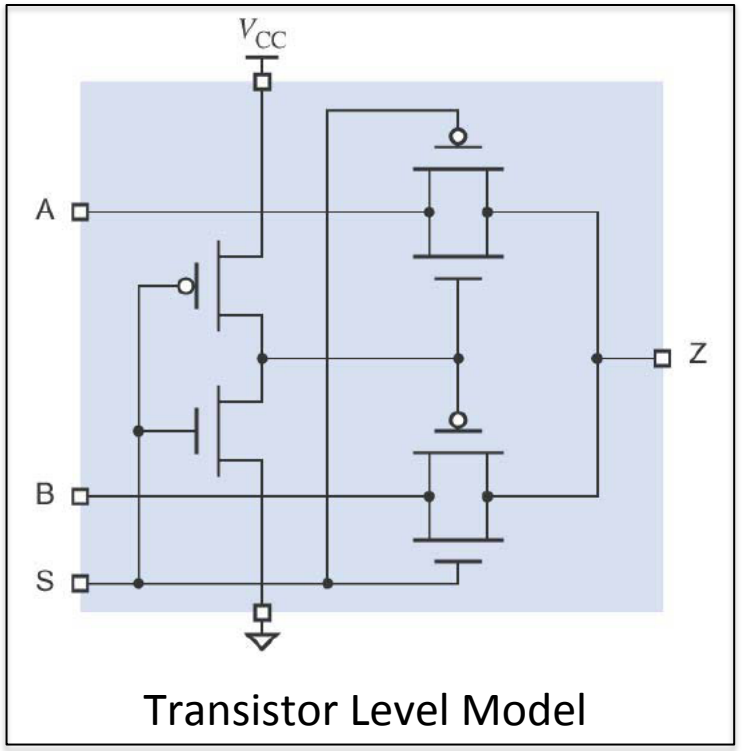
S	A	B	Z
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

… Design Abstraction Levels



Another Gate Level Model





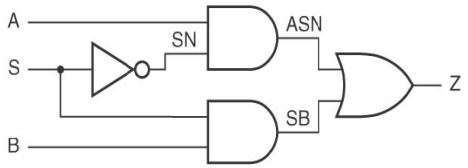
Transistor Level Model

… Design Abstraction Levels: HDL

```
library ieee;
use ieee.std_logic_1164.all;
library altera;
use altera.altera_primitives_components.all;

entity mux is
port (A, B, S: in std_logic;
        Z:      out std_logic);
end mux;

architecture struct of mux is

component INV is:
  port A: in std_logic;
        F: out std_logic);
end component;
component AND2 is:
  port A, B: in std_logic;
        F: out std_logic);
end component;
component OR2 is:
  port A, B: in std_logic;
        F: out std_logic);
end component;
```

```
signal SN, ASN, SB: std_logic;

begin
U1: port map INV (S, SN);
U2: port map AND2 (A, SN, ASN);
U3: port map AND2 (S, B, SB);
U4: port map OR2 (ASN, SB, Z);
end struct;
```

structural coding



Highly discouraged coding style except to:

- implement hierarchy
- infer “special” logic blocks (e.g. SRAM)

GONZAGA UNIVERSITY

… Design Abstraction Levels: HDL

```
library ieee;
use ieee.std_logic_1164.all;

entity mux is
port (A, B, S: in std_logic;
        Z:       out std_logic);
end mux;

architecture rtl_conc of mux is
begin
  Z <= A when S = '0' else B;
end rtl_conc;
```

concurrent RTL coding

Preferred Style

```
library ieee;
use ieee.std_logic_1164.all;

entity mux is
port (A, B, S: in std_logic;
        Z:       out std_logic);
end mux;

architecture rtl of mux is
begin
  process mux_p (A, B, S)
    if (S = '0') then
      Z <= A;
    else
      Z <= B;
    end if;
  end process mux_p;
end rtl;
```

sequential RTL coding