Lecture 11 Timing Constraints and High-Speed RTL Design

Outline

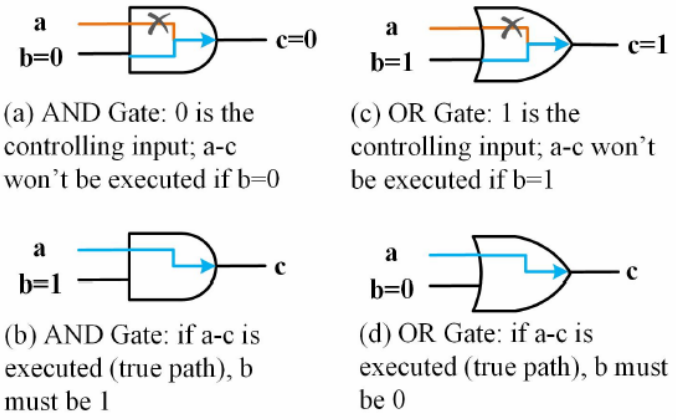
- 11.1 Critical Path and True/False Path
- 11.2 Timing Information
- 11.3 Maximum Delay Constraints
- 11.4 Minimum Delay Constraints
- 11.5 Clock Skew
- 11.6 Clock Domain Crossing

Outline

- 11.1 Critical Path and True/False Path
- 11.2 Timing Information
- 11.3 Maximum Delay Constraints
- 11.4 Minimum Delay Constraints
- 11.5 Clock Skew
- 11.6 Clock Domain Crossing

11.1 Critical Path and True/False Path

- 11.1.1 True/False Path for AND/NAND and OR/NOR Gates
  - Examine the ***control input*** of a basic AND/NAND gate
    - Logic 0 is the controlling input because if one input (input b in this figure) is logic 0 the output c must be 0/1.
    - Foe example, if b=0, the a-c path will not be executed, or a-c is actually a false path. In other words, if we want to make a-c a true path, the b input must be 1.
  - Examine the ***control input*** of a basic OR/NOR gate
    - logic 1 is the controlling input. If we want to make a-c a true path the input b must be 0.



(a) AND Gate: 0 is the controlling input; a-c won't be executed if b=0

(c) OR Gate: 1 is the controlling input; a-c won't be executed if b=1

(b) AND Gate: if a-c is executed (true path), b must be 1

(d) OR Gate: if a-c is executed (true path), b must be 0

FIGURE 11.1
Controlling Inputs of AND/NAND and OR/NOR Gates

- 11.1.2 ***True/False Path*** for Combinational Circuits
  - A. Examples of True/False Path

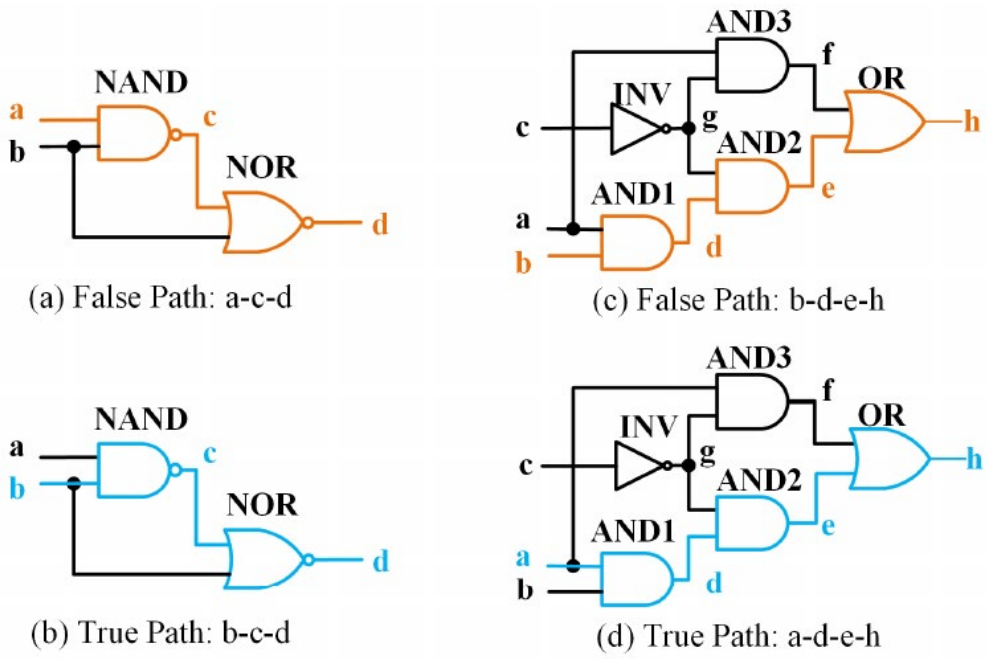


FIGURE 11.2
Two Examples of True/False Path

- 11.1.2 True/False Path for Combinational Circuits
  - B. Other Examples of True/False Path

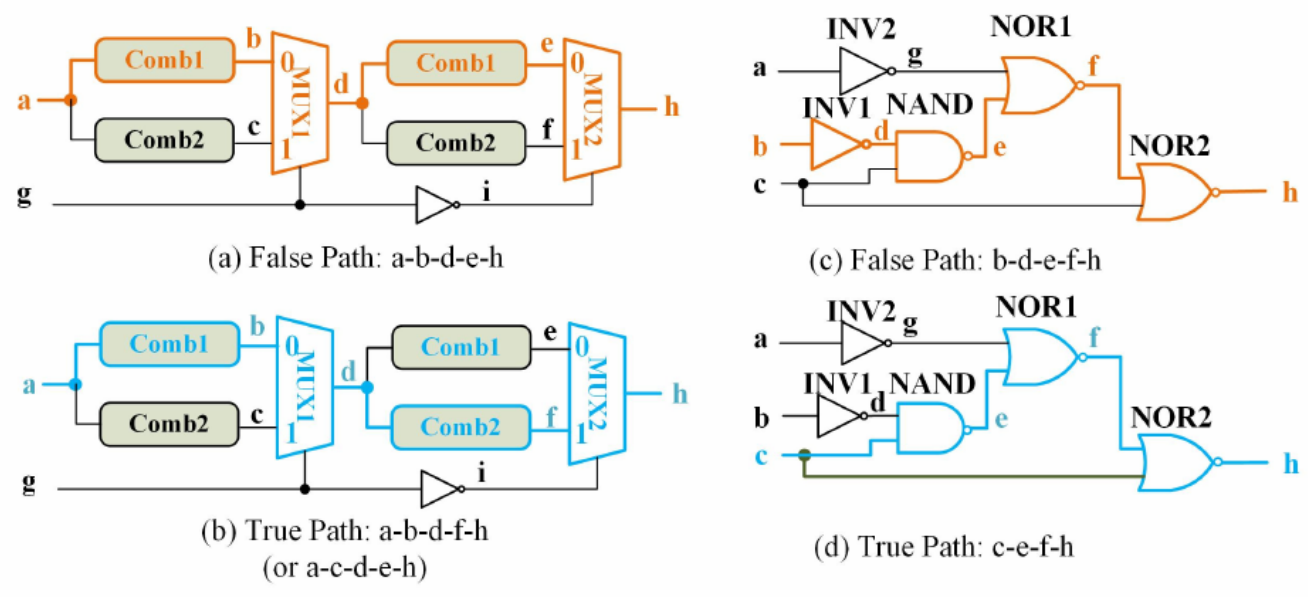


FIGURE 11.3
Other Two Examples of True/False Path

- 11.1.3 Propagation/Contamination Delay and Critical Path
  - A. Propagation/Contamination Delay
    - In a combinational circuit comprising multiple logic elements:
      - ***Contamination delay*** (referred to as ``t_cd'') is the aggregate of the contamination delays of all logic elements in the shortest true path.
      - ***Propagation delay*** (referred to as ``t_pd'') is the sum of the propagation delays of all logic elements along the longest true path.
      - It's crucial to emphasize that when analyzing propagation and contamination delays, only true paths are considered, as false paths are never executed.

- 11.1.3 Propagation/Contamination Delay and Critical Path
  - B. ***Critical Path*** within Register-Transfer Design
    - The concept of a critical path holds great significance in register-transfer level designs.
    - It pertains to the propagation delay that exists between two registers within the entire circuit.

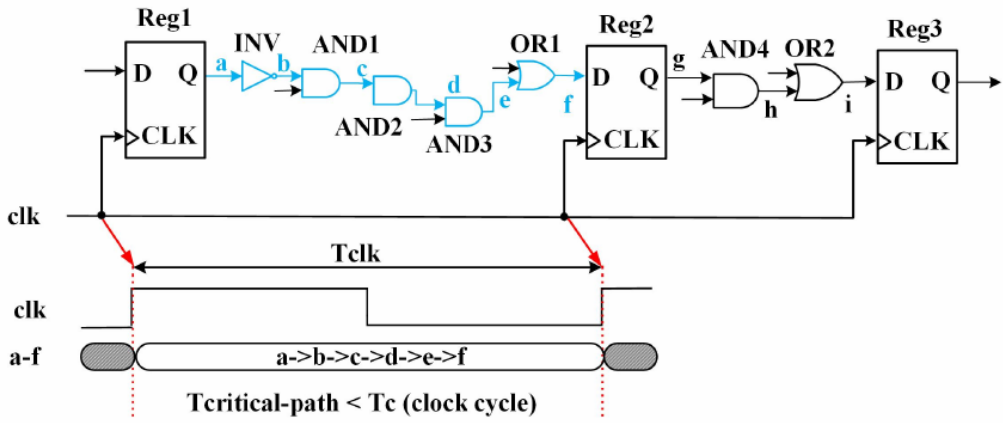


FIGURE 11.4
Critical Path within Register-Transfer Design Circuit

- 11.1.3 Propagation/Contamination Delay and Critical Path
  - B. Critical Path within Register-Transfer Design
    - Clock cycle time (Tc) cannot be smaller than the longest true path delay ($Tcritical{-}path$): $T_c \geq T_{critical-path}$
      - Failing this condition, the computation of the combinational circuit will extend beyond the allocated clock cycle, resulting in ***timing violations*** and compromising the integrity of the entire register-transfer design.
      - Consequently, the critical path holds immense significance in shaping both the ***clock frequency*** and the ***overall operational speed***.

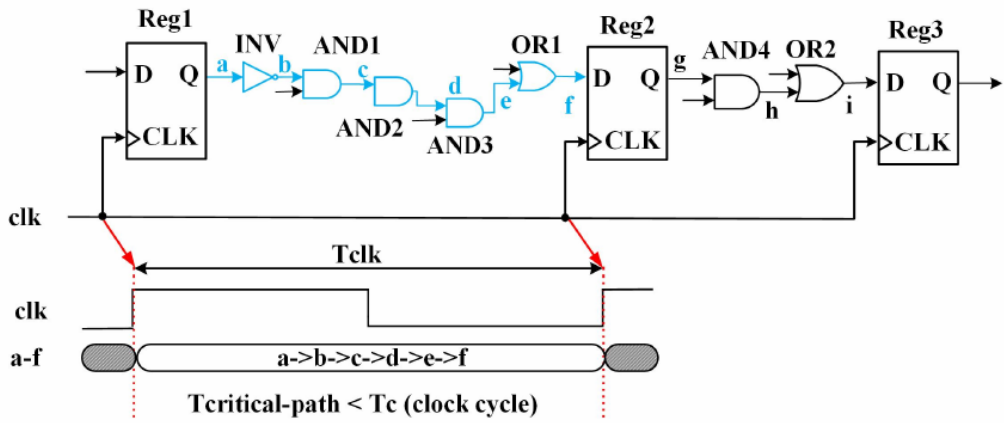


FIGURE 11.4
Critical Path within Register-Transfer Design Circuit

Outline

- 11.1 Critical Path and True/False Path
- 11.2 Timing Information
- 11.3 Maximum Delay Constraints
- 11.4 Minimum Delay Constraints
- 11.5 Clock Skew
- 11.6 Clock Domain Crossing

- 11.2.1 Timing Constraints
  - Setup Time (t_setup):
    - Signifies the duration for which the register input ``D'' must remain stable prior to the arrival of the active clock edge.
    - Failure to satisfy the setup time requirement can lead to instability in the register's operation, potentially resulting in meta-stability.
  - Hold Time (t_hold):
    - Refers to the duration during which the register input ``D'' must remain steady subsequent to the occurrence of the active clock edge.
    - If the hold time condition is not met, the register's operation can become unstable, again inviting the possibility of meta-stability.

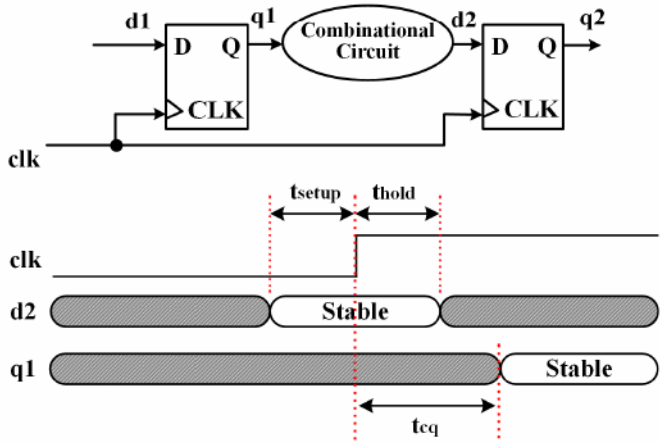


FIGURE 11.5
Setup Time, Hold Time, and Clock-to-Q Delay

- 11.2.1 Timing Constraints
  - Clock-q delay (t_cq):
    - This delay corresponds to the time elapsed from the instant of the active clock edge to the point when a valid output ``Q'' displays from the register.
    - Attempting to access the output prior to the expiration of this delay can yield inconsistent outcomes due to potential fluctuations in the register's output ``Q''.

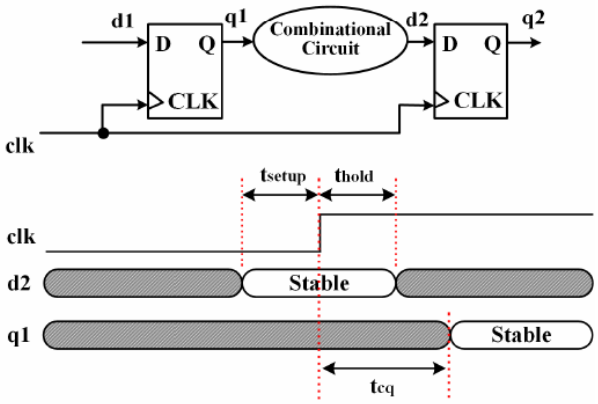


FIGURE 11.5
Setup Time, Hold Time, and Clock-to-Q Delay

TABLE 11.1
Timing Attributes

Timing Info	Descriptions
t_{pd}	Combinational Circuit Propagation Delay
t_{cd}	Combinational Circuit Contamination Delay
t_{cq}	Register Clock-to-Q Delay
t_{setup}	Register Setup Time Constraint
t_{hold}	Register Hold Time Constraint

- 11.2.2 Propagation/Contamination Delay Examples
  - A. Timing Attributes
  - B. Examples of Propagation/Contamination Delay

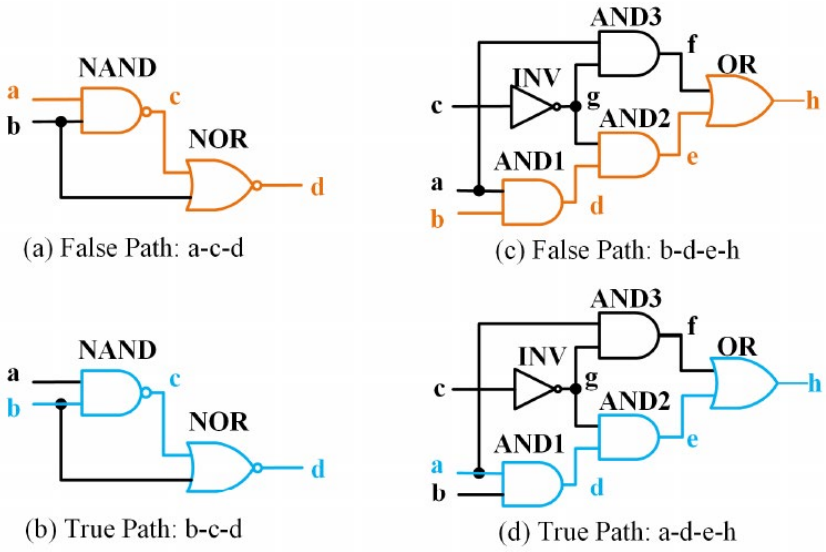


FIGURE 11.2
Two Examples of True/False Path

$$t_{pd} = t_{pd}(AND1) + t_{pd}(AND2) + t_{pd}(OR) = 3 \times 1.5 = 4.5\,ns.$$

$$t_{cd} = t_{cd}(AND3) + t_{cd}(OR) = 2 \times 0.5 = 1.0\,ns.$$

$$t_{pd} = t_{pd}(NAND) + t_{pd}(NOR) = 1.5 + 1.5 = 3.0\,ns.$$

$$t_{cd} = t_{cd}(NOR) = 0.5\,ns.$$

TABLE 11.2
Timing Example for Combinational Circuits

Circuits	t_{pd} (ns)	t_{cd} (ns)
AND/NAND	1.5	0.5
OR/NOR	1.5	0.5
XOR/XNOR	1.5	0.5
INV (Inverter)	1.0	0.5
BUF (Buffer)	1.5	1.0
Comparator	1.0	0.5
MUX (Multiplexer)	3.0	1.0
ADD (Adder)	2.5	1.5
SUB (Subtractor)	2.5	1.5
MUL (Multiplier)	3.5	2.0
SHF (Shifter)	0.5	0.5

- 11.2.2 Propagation/Contamination Delay Examples
  - C. Other Examples of Propagation/Contamination Delay

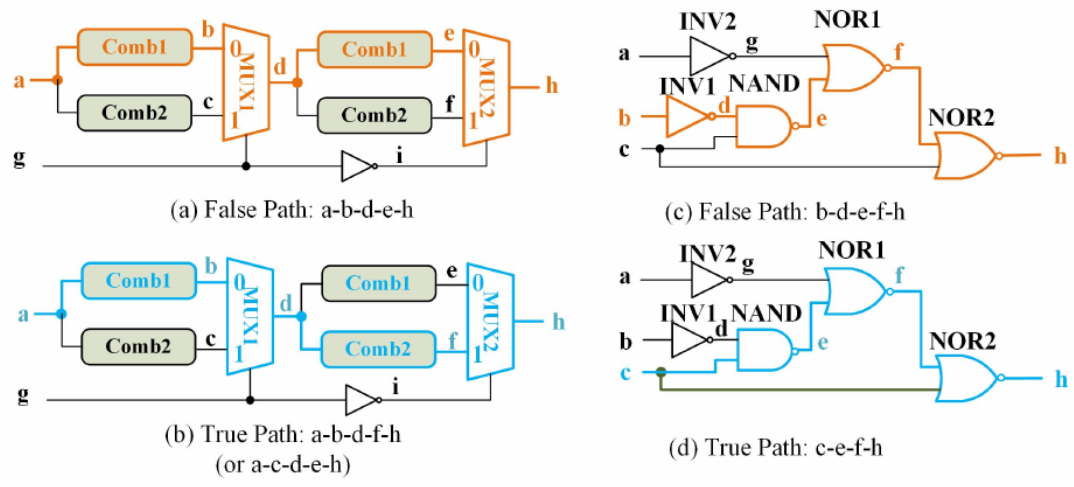


TABLE 11.2
Timing Example for Combinational Circuits

Circuits	t_{pd} (ns)	t_{cd} (ns)
AND/NAND	1.5	0.5
OR/NOR	1.5	0.5
XOR/XNOR	1.5	0.5
INV (Inverter)	1.0	0.5
BUF (Buffer)	1.5	1.0
Comparator	1.0	0.5
MUX (Multiplexer)	3.0	1.0
ADD (Adder)	2.5	1.5
SUB (Subtractor)	2.5	1.5
MUL (Multiplier)	3.5	2.0
SHF (Shifter)	0.5	0.5

FIGURE 11.3
Other Two Examples of True/False Path

$$t_{pd} = t_{pd}(Comb1)+t_{pd}(Comb2)+2\times t_{pd}(MUX) = 10.0+5.0+2\times 3.0 = 21.0\,ns.$$

$$t_{cd} = t_{cd}(INV) + t_{cd}(MUX) = 0.5 + 1.0 = 1.5\,ns.$$

$$t_{pd} = t_{pd}(NAND)+t_{pd}(NOR1)+t_{pd}(NOR2) = 1.5+2\times 1.5 = 4.5\,ns.$$

$$t_{cd} = t_{cd}(NOR2) = 0.5\,ns.$$

Outline

- 11.1 Critical Path and True/False Path
- 11.2 Timing Information
- 11.3 Maximum Delay Constraints
- 11.4 Minimum Delay Constraints
- 11.5 Clock Skew
- 11.6 Clock Domain Crossing

- 11.3.1 Maximum Delay Constraint and Setup Time Violation
  - A. Maximum Delay Constraint
    - The synchronization of timing delay and the setup time window must align within the bounds of the clock cycle.
      - Clock to Q delay of DFF (t_cq)
      - Propagation delay through combinational logic (t_pd)
      - Required setup time (t_setup) of registers

$$T_{clk} \geq t_{cq} + t_{pd} + t_{setup}.$$

$$T_{clk(min)} = t_{cq} + t_{pd} + t_{setup}.$$

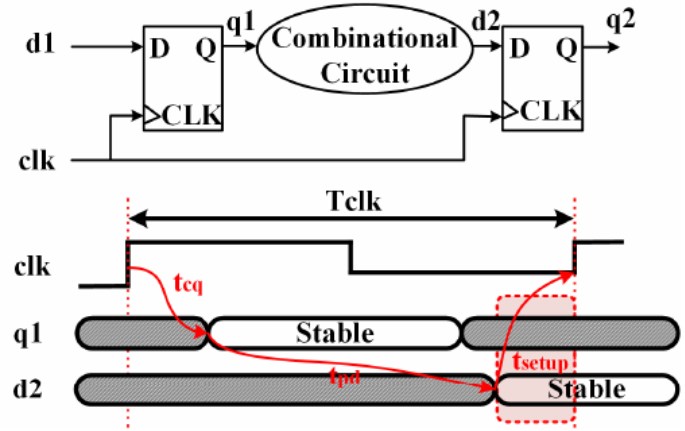


FIGURE 11.6
Maximum Delay Constraint

- 11.3.1 Maximum Delay Constraint and Setup Time Violation
  - B. Solutions to Setup Time Violation
    - If a ***setup time violation*** occurs, it is possible that the data being transferred into the second register may enter a metastable state. This instability has the potential to disrupt the overall system functionality.
    - Addressing setup time violations can be approached in a couple of ways.
      - Extending the clock period, allowing sufficient time for data to be updated before the setup window closes. However, this adjustment would result in a reduction of the clock frequency.
      - Modifying the combinational circuit to minimize its propagation delay. In cases where the critical path is excessively lengthy, it can be partitioned into multiple shorter segments, with additional registers introduced along the data path.

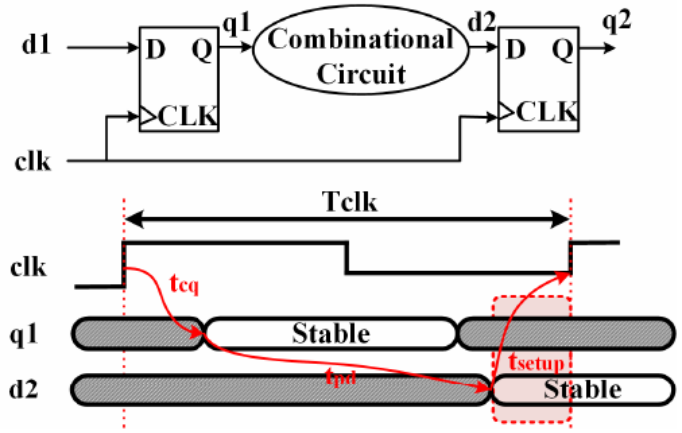


FIGURE 11.6
Maximum Delay Constraint

- 11.3.2 Maximum Operational Frequency

$$T_{clk(min)} = t_{cq} + t_{pd} + t_{setup}.$$

$$MOF = \frac{1}{T_{clk(min)}}.$$

  - To attain the utmost MOF, it is imperative to:
    - Harness cutting-edge technology, characterized by reduced clock-to-Q delay and setup time restrictions.
    - From RTL designers' vantage point, diminishing the critical path of the combinational circuit design can also contribute significantly to augmenting the MOF.

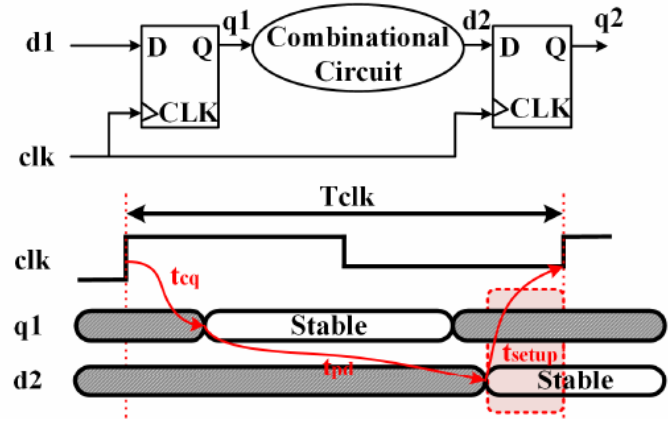


FIGURE 11.6
Maximum Delay Constraint

- 11.3.3 Setup Time Slack
  - Delineates the available time window for the combinational circuit design between registers

$$t_{setup-slack} \leq T_{clk} - t_{cq} - t_{setup}.$$

TABLE 11.3
Timing Example for Sequential Registers

Circuits	t_{cq} (ns)	t_{setup} (ns)	t_{hold} (ns)
Register	2.0	1.0	1.5

  - Example:
    - Clock frequency specification is set at or above 100 MHz, consequently yielding a minimum clock period of 10 ns.
    - In order to satisfy the speed requirement, the maximum setup slack can be calculated as

$$t_{setup-slack} \leq 10 - t_{cq} - t_{setup} = 7.0$$

    - This implies that there is a 7.0 ns timeframe available to incorporate the combinational functions between the registers. Should the critical path exceed this 7.0 ns window, it would necessitate the division of the combinational circuit into two or more shorter segments by inserting additional registers.

- 11.3.4 Design Example #1: Binary Counter MOF

$$t_{pd} = t_{pd}(ADD) + t_{pd}(MUX) = 2.5 + 3.0 = 5.5 \text{ ns.}$$

$$MOF = \frac{1}{t_{cq} + t_{pd} + t_{setup}} = \frac{1}{2.0 + 5.5 + 1.0} = 117.65\,MHz$$

```
1  module binary_counter (input            clk,
2                         input            rst,
3                         output reg [3:0] cnt);
4  wire [3:0] nxt_cnt = (cnt==4'h9) ? 4'h0 : cnt+4'h1;
5  always @(posedge clk, negedge rst) begin
6    if (~rst) begin
7      cnt <= 4'h0   ;
8    end else begin
9      cnt <= nxt_cnt;
10   end
11 end
12 endmodule
```

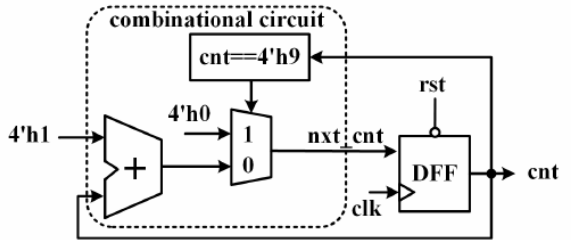


FIGURE 11.7
Design Example #1: Counter from Hexadecimal 0 to 9

TABLE 11.3
Timing Example for Sequential Registers

Circuits	t_{cq} (ns)	t_{setup} (ns)	t_{hold} (ns)
Register	2.0	1.0	1.5

TABLE 11.2
Timing Example for Combinational Circuits

Circuits	t_{pd} (ns)	t_{cd} (ns)
AND/NAND	1.5	0.5
OR/NOR	1.5	0.5
XOR/XNOR	1.5	0.5
INV (Inverter)	1.0	0.5
BUF (Buffer)	1.5	1.0
Comparator	1.0	0.5
MUX (Multiplexer)	3.0	1.0
ADD (Adder)	2.5	1.5
SUB (Subtractor)	2.5	1.5
MUL (Multiplier)	3.5	2.0
SHF (Shifter)	0.5	0.5

11.3 Maximum Delay Constraints

- 11.3.5 Design Example #2: Sequential Circuit MOF

$$t_{pd} = t_{pd}(MUX) + t_{pd}(AND2) = 3.0 + 1.5 = 4.5$$

$$MOF = \frac{1}{t_{cq} + t_{pd} + t_{setup}} = \frac{1}{1.0 + 4.5 + 2.0} = 133.33\,MHz.$$

```
1  module seq_circuit (input clk, rst, a, b, c, output reg h);
2  reg  a1,b1,c1,c2,d2,e2;
3  wire d1 = a1 & b1 ;
4  wire e1 = a1 | b1 ;
5  wire f2 = c2 ? e2 : d2 ;
6  wire g2 = f2 & c2 ;
7
8  always @(posedge clk, negedge rst) begin
9    if (~rst) begin
10     a1<=1'b0; b1<=1'b0; c1<=1'b0;
11     c2<=1'b0; d2<=1'b0; e2<=1'b0;
12     h <=1'b0;
13   end else begin
14     a1 <= a ;
15     b1 <= b ;
16     c1 <= c ;
17     c2 <= c1;
18     d2 <= d1;
19     e2 <= e1;
20     h  <= g2;
21   end
22 end
23 endmodule
```

TABLE 11.3
Timing Example for Sequential Registers

Circuits	t_{cq} (ns)	t_{setup} (ns)	t_{hold} (ns)
Register	2.0	1.0	1.5

TABLE 11.2
Timing Example for Combinational Circuits

Circuits	t_{pd} (ns)	t_{cd} (ns)
AND/NAND	1.5	0.5
OR/NOR	1.5	0.5
XOR/XNOR	1.5	0.5
INV (Inverter)	1.0	0.5
BUF (Buffer)	1.5	1.0
Comparator	1.0	0.5
MUX (Multiplexer)	3.0	1.0
ADD (Adder)	2.5	1.5
SUB (Subtractor)	2.5	1.5
MUL (Multiplier)	3.5	2.0
SHF (Shifter)	0.5	0.5

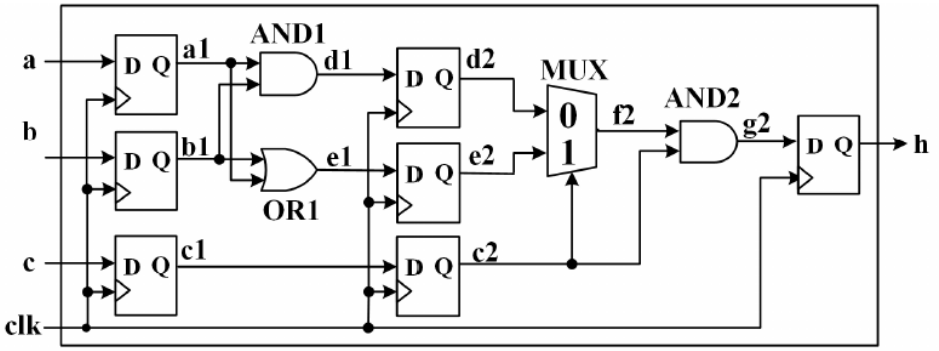
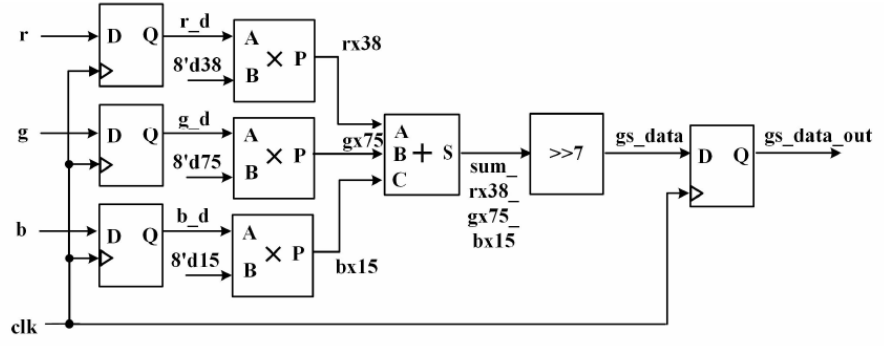


FIGURE 11.8
Design Example #2: Register-Transfer Design Circuit

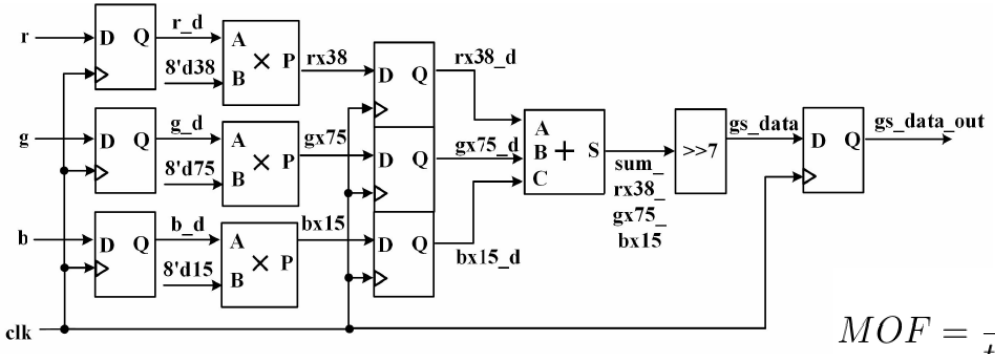
- 11.3.6 Design Example #3: Pipeline Design MOF and Latency

$$t_{pd} = t_{pd}(MUL) + t_{pd}(ADD) + t_{pd}(SHF) = 3.5 + 2.5 + 0.5 = 6.5 \text{ ns}$$

$$MOF = \frac{1}{t_{cq} + t_{pd} + t_{setup}} = \frac{1}{1.0 + 6.5 + 2.0} = 105.26\,MHz.$$



(a) Critical Path including Multipliers, an Adder, and a Shifter



(b) Registers Insert Within Critical Path

TABLE 11.3
Timing Example for Sequential Registers

Circuits	t_{cq} (ns)	t_{setup} (ns)	t_{hold} (ns)
Register	2.0	1.0	1.5

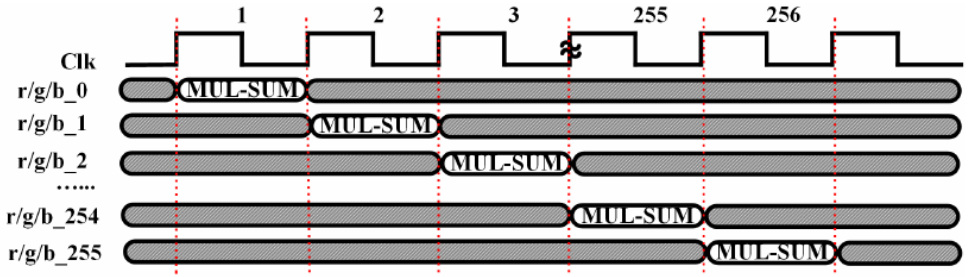
TABLE 11.2
Timing Example for Combinational Circuits

Circuits	t_{pd} (ns)	t_{cd} (ns)
AND/NAND	1.5	0.5
OR/NOR	1.5	0.5
XOR/XNOR	1.5	0.5
INV (Inverter)	1.0	0.5
BUF (Buffer)	1.5	1.0
Comparator	1.0	0.5
MUX (Multiplexer)	3.0	1.0
ADD (Adder)	2.5	1.5
SUB (Subtractor)	2.5	1.5
MUL (Multiplier)	3.5	2.0
SHF (Shifter)	0.5	0.5

$$MOF = \frac{1}{t_{cq} + t_{pd} + t_{setup}} = \frac{1}{1.0 + 3.5 + 2.0} = 153.85\,MHz.$$

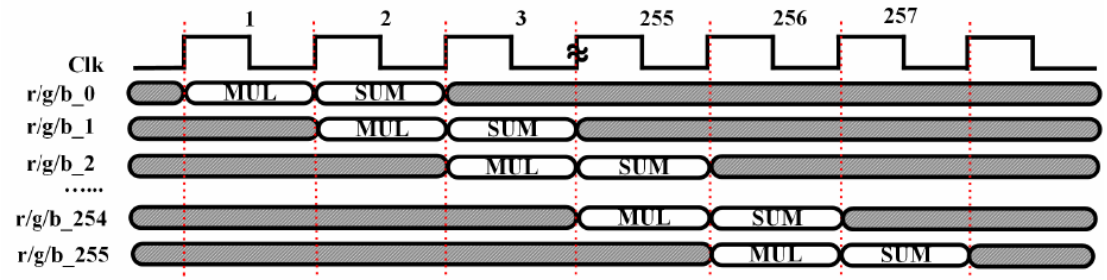
FIGURE 11.9
Design Example #3: Pipeline Design with Higher MOF

- 11.3.7 Design Example #3: Pipeline Design MOF and Latency
  - B. Latency Analysis of Design Example \#3



(a) Timing Diagram of Unpipelined Design

$$256 \times \frac{1}{105.26} = 2,432 \text{ ns.}$$



(b) Timing Diagram of Pipelined Design

$$257 \times \frac{1}{153.85} = 1,670.5 \text{ ns}$$

FIGURE 11.10
Timing Diagram of Pipeline Design with Higher MOF

Outline

- 11.1 Critical Path and True/False Path
- 11.2 Timing Information
- 11.3 Maximum Delay Constraints
- 11.4 Minimum Delay Constraints
- 11.5 Clock Skew
- 11.6 Clock Domain Crossing

11.4 Minimum Delay Constraints

- 11.4.1 Minimum Delay Constraint and Hold Time Violation
  - Encompasses both the clock-to-Q delay (t_cq) and the contamination delay specific to the combinational circuit (t_cd).
    - Clock to Q delay of DFF (t_cq)
    - Contamination delay through combinational logic (t_cd)
  - Hold time violation
    - When this interval is too short or if the hold time (t_hold) is too long, there is a risk that the data at the ``d2'' might be updated within the hold window, potentially leading to a **hold time violation**.

$$t_{cq} + t_{cd} > t_{hold}.$$

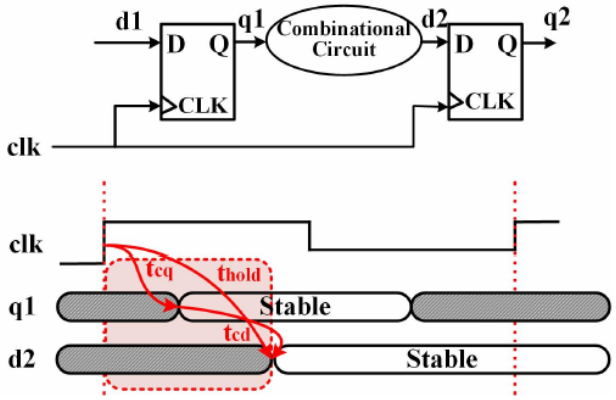


FIGURE 11.11
Minimum Delay Constraint

11.4 Minimum Delay Constraints

- 11.4.1 Minimum Delay Constraint and Hold Time Violation
  - Failure to prevent a hold time violation can result in unstable data entering the second register and corrupting the circuit's state.
    - In such cases, circuit redesign may be necessary, such as adding buffers to delay the signal updating without changing the design functions.
    - Designers should exercise caution to prevent such failures, as redesigning RTL code can be time-consuming, requiring regression testing and other procedures in the design flow.

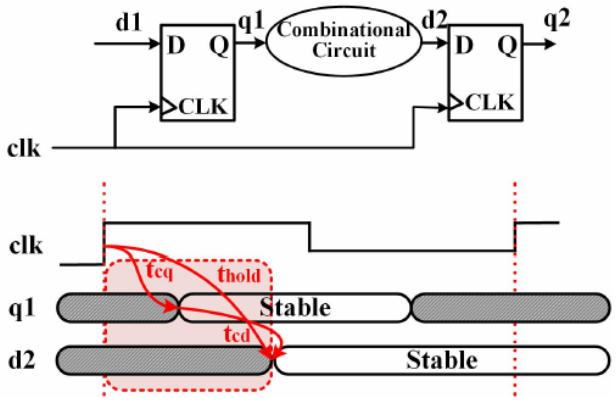


FIGURE 11.11
Minimum Delay Constraint

- 11.4.2 Design Example #1: Hold Timing Analysis of Binary Counter

$$t_{cd} = t_{cd}(Comparator) + t_{cd}(MUX) = 0.5 + 1.0 = 1.5 \text{ ns}$$

$$t_{cd} + t_{cq} = 1.5 + 2.0 = 3.5 \text{ ns}.$$

```
1  module binary_counter (input             clk,
2                         input             rst,
3                         output reg [3:0]  cnt);
4  wire [3:0] nxt_cnt = (cnt==4'h9) ? 4'h0 : cnt+4'h1;
5  always @(posedge clk, negedge rst) begin
6    if (~rst) begin
7      cnt <= 4'h0   ;
8    end else begin
9      cnt <= nxt_cnt;
10   end
11 end
12 endmodule
```

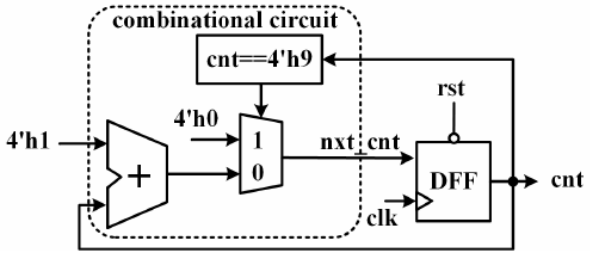


FIGURE 11.7
Design Example #1: Counter from Hexadecimal 0 to 9

TABLE 11.3
Timing Example for Sequential Registers

Circuits	t_{cq} (ns)	t_{setup} (ns)	t_{hold} (ns)
Register	2.0	1.0	1.5

TABLE 11.2
Timing Example for Combinational Circuits

Circuits	t_{pd} (ns)	t_{cd} (ns)
AND/NAND	1.5	0.5
OR/NOR	1.5	0.5
XOR/XNOR	1.5	0.5
INV (Inverter)	1.0	0.5
BUF (Buffer)	1.5	1.0
Comparator	1.0	0.5
MUX (Multiplexer)	3.0	1.0
ADD (Adder)	2.5	1.5
SUB (Subtractor)	2.5	1.5
MUL (Multiplier)	3.5	2.0
SHF (Shifter)	0.5	0.5

- 11.4.2 Design Example #2: Hold Timing Constraint of Sequential Circuit

```
1  module seq_circuit (input clk, rst, a, b, c, output reg h);
2  reg  a1,b1,c1,c2,d2,e2;
3  wire d1 = a1 & b1 ;
4  wire e1 = a1 | b1 ;
5  wire f2 = c2 ? e2 : d2 ;
6  wire g2 = f2 & c2 ;
7
8  always @(posedge clk, negedge rst) begin
9    if (~rst) begin
10     a1<=1'b0; b1<=1'b0; c1<=1'b0;
11     c2<=1'b0; d2<=1'b0; e2<=1'b0;
12     h <=1'b0;
13   end else begin
14     a1 <= a ;
15     b1 <= b ;
16     c1 <= c ;
17     c2 <= c1;
18     d2 <= d1;
19     e2 <= e1;
20     h  <= g2;
21   end
22 end
23 endmodule
```

$$t_{cq} > t_{hold}.$$

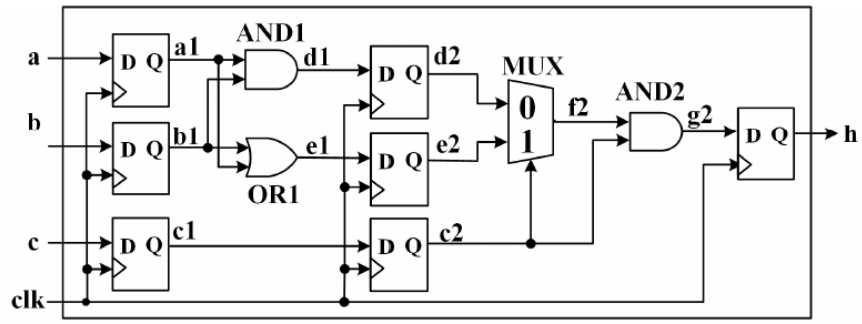


FIGURE 11.8
Design Example #2: Register-Transfer Design Circuit

TABLE 11.3
Timing Example for Sequential Registers

Circuits	t_{cq} (ns)	t_{setup} (ns)	t_{hold} (ns)
Register	2.0	1.0	1.5

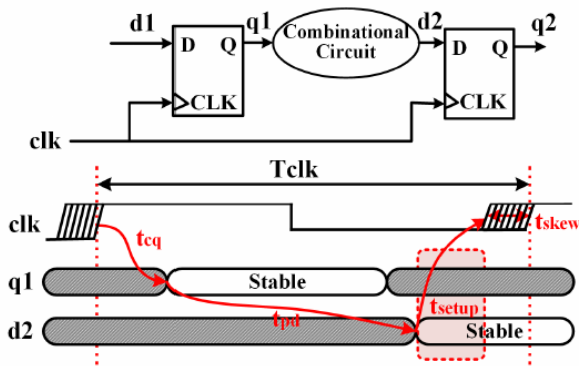
TABLE 11.2
Timing Example for Combinational Circuits

Circuits	t_{pd} (ns)	t_{cd} (ns)
AND/NAND	1.5	0.5
OR/NOR	1.5	0.5
XOR/XNOR	1.5	0.5
INV (Inverter)	1.0	0.5
BUF (Buffer)	1.5	1.0
Comparator	1.0	0.5
MUX (Multiplexer)	3.0	1.0
ADD (Adder)	2.5	1.5
SUB (Subtractor)	2.5	1.5
MUL (Multiplier)	3.5	2.0
SHF (Shifter)	0.5	0.5

Outline

- 11.1 Critical Path and True/False Path
- 11.2 Timing Information
- 11.3 Maximum Delay Constraints
- 11.4 Minimum Delay Constraints
- 11.5 Clock Skew
- 11.6 Clock Domain Crossing

- 11.5.1 Maximum Delay Constraints with Clock Skew
  - Clocks arriving at different registers bring with them inherent uncertainties about their exact arrival times. This intrinsic unpredictability can lead to a reduced timeframe available for data processing.
  - In the most unfavorable scenario involving the maximum delay constraint, the *initial register encounters a delayed clock signal while the second register experiences an early clock signal*, ultimately reducing the available timing window.
    - The hashed lines represent the possible range of clock arrival times due to clock skew.



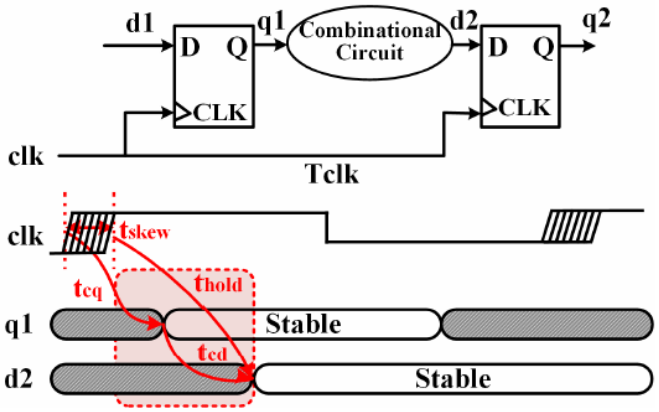
$$T_{clk} \geq t_{cq} + t_{pd} + t_{setup} + t_{skew}.$$

$$MOF = \frac{1}{t_{cq} + t_{pd} + t_{setup} + t_{skew}}.$$

$$T_{setup-slack} \leq T_{clk} - t_{cq} - t_{setup} - t_{skew}.$$

FIGURE 11.12
Maximum Delay Constraint with Clock Skew

- 11.5.2 Minimum Delay Constraints with Clock Skew
  - The most challenging situation concerning the minimum delay constraint arises when the first register receives its clock signal ahead of time, and the second register experiences a delayed clock
  - In such instances, clock skew effectively extends the hold time requirement for the second register.



$$t_{cq} + t_{cd} > t_{hold} + t_{skew}.$$

FIGURE 11.13
Minimum Delay Constraint with Clock Skew

- 11.5.3 Design Example #1: Clock Skew Analysis of Binary Counter:
  - Clock skew: 1 ns

TABLE 11.3
Timing Example for Sequential Registers

Circuits	t_{cq} (ns)	t_{setup} (ns)	t_{hold} (ns)
Register	2.0	1.0	1.5

$$T_{min} = t_{cq} + t_{pd} + t_{setup} + t_{skew} = 2.0 + 5.5 + 1.0 + 1.0 = 9.5 \text{ ns}.$$

$$MOF = \frac{1}{T_{min}} = 105.26 \text{ MHz}$$

  - No hold time violation

$$t_{hold} + t_{skew} = 1.5 + 1.0 = 2.5 \text{ ns}$$

$$t_{cd} + t_{cq} = 1.5 + 2.0 = 3.5 \text{ ns}$$

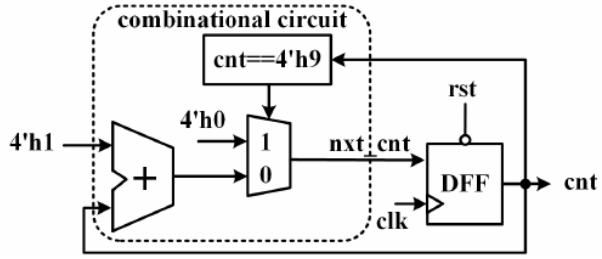


FIGURE 11.7
Design Example #1: Counter from Hexadecimal 0 to 9

TABLE 11.2
Timing Example for Combinational Circuits

Circuits	t_{pd} (ns)	t_{cd} (ns)
AND/NAND	1.5	0.5
OR/NOR	1.5	0.5
XOR/XNOR	1.5	0.5
INV (Inverter)	1.0	0.5
BUF (Buffer)	1.5	1.0
Comparator	1.0	0.5
MUX (Multiplexer)	3.0	1.0
ADD (Adder)	2.5	1.5
SUB (Subtractor)	2.5	1.5
MUL (Multiplier)	3.5	2.0
SHF (Shifter)	0.5	0.5

- 11.5.4 Design Example #2: Clock Skew Analysis of Sequential Circuit:
  - MOF: $t_{cq} + t_{pd} + t_{setup} + t_{skew} = 2.0 + 4.5 + 1.0 + 1.0 = 8.5$ ns

$$MOF = \frac{1}{t_{cq} + t_{pd} + t_{setup} + t_{skew}} = \frac{1}{2.0 + 4.5 + 1.0 + 1.0} = 117.65 \text{ MHz}.$$

  - Hold time violation happens, so that add a buffer to address the problem.

$$t_{hold} + t_{skew} = 1.5 + 1.0 = 2.5 \text{ ns}. \qquad t_{cd}(BUF) + t_{cq} = 1.0 + 2.0 = 3.0 \text{ ns}$$

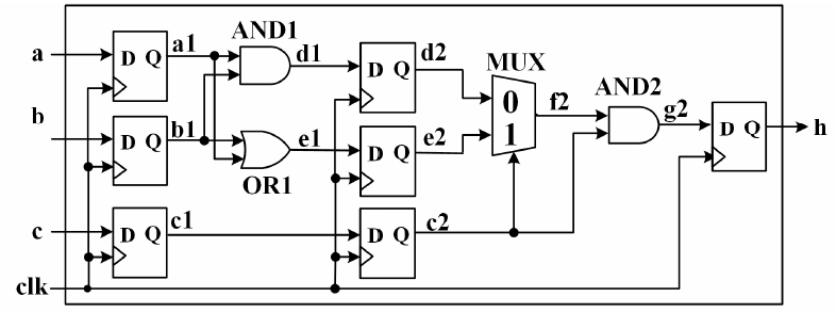


FIGURE 11.8
Design Example #2: Register-Transfer Design Circuit

TABLE 11.3
Timing Example for Sequential Registers

Circuits	t_{cq} (ns)	t_{setup} (ns)	t_{hold} (ns)
Register	2.0	1.0	1.5

TABLE 11.2
Timing Example for Combinational Circuits

Circuits	t_{pd} (ns)	t_{cd} (ns)
AND/NAND	1.5	0.5
OR/NOR	1.5	0.5
XOR/XNOR	1.5	0.5
INV (Inverter)	1.0	0.5
BUF (Buffer)	1.5	1.0
Comparator	1.0	0.5
MUX (Multiplexer)	3.0	1.0
ADD (Adder)	2.5	1.5
SUB (Subtractor)	2.5	1.5
MUL (Multiplier)	3.5	2.0
SHF (Shifter)	0.5	0.5

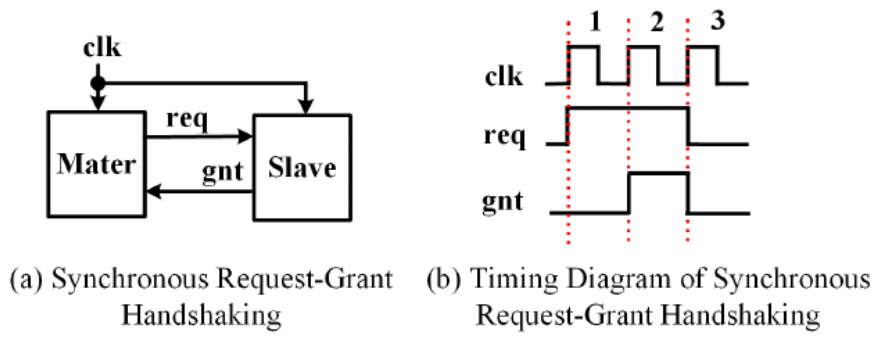
Outline

- 11.1 Critical Path and True/False Path
- 11.2 Timing Information
- 11.3 Maximum Delay Constraints
- 11.4 Minimum Delay Constraints
- 11.5 Clock Skew
- 11.6 Clock Domain Crossing

11.6 Clock Domain Crossing

- Clock Domain Crossing
  - In RTL design, the dependable capture and storage of signal values within registers rely on the activation of clock edges. Nevertheless, signals that enter a ***metastable*** state during setup or hold windows have the potential to disrupt the exact timing requirements for registers, which can ultimately result in malfunctions within the circuit and system.
  - One of the most common scenarios prone to introducing signal race conditions is the RTL design of signals crossing between different clock domains.
  - Timing violations associated with CDC (clock domain crossing) are elusive and extremely challenging to debug, underscoring the importance of correctly designing synchronization logic from the outset.

- 11.6.1 Request-Grant Handshaking
  - The master can initiate bus requests, and the slave grants the requests if it is available, thereby completing the handshaking protocol.
  - For the sake of simplicity, we assume that the slave is always prepared to respond, allowing the grant to be triggered whenever the request is active.



(a) Synchronous Request-Grant Handshaking

(b) Timing Diagram of Synchronous Request-Grant Handshaking

FIGURE 11.14
Synchronous Request-Grant Handshaking

- 11.6.1 Request-Grant Handshaking
  - Figure 11.5 illustrates an asynchronous design scenario involving communication between a master in clock domain A and a slave in clock domain B.
  - This specific design configuration is commonly referred to as ***clock domain crossing***, a situation that can introduce significant timing challenges.
    - This is primarily because registered signals from asynchronous clock domains may potentially experience metastability when their data inputs fall within the setup or hold time window. As a result, the implementation of specific RTL design rules is crucial to ensure reliable data transfer across distinct clock domains.

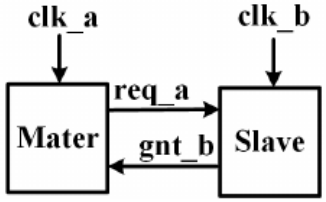


FIGURE 11.15
Asynchronous Request-Grant Handshaking

- 11.6.2 Design Specification of Request-Grant Handshaking Crossing Asynchronouse Clock Domains
  - A. Design IOs

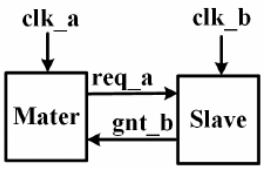


FIGURE 11.15
Asynchronous Request-Grant Handshaking

TABLE 11.4
Master IOs Description

Name	Direction	Bit Width	Description
clk_a	Input	1	Clock, rising edge, 33 MHz
rst_a	Input	1	Asynchronous reset, 0 valid
en_a	Input	1	Enable signal, 1 valid
req_a	Output	1	Request signal from clock domain *A*
gnt_b	Input	1	Grant signal from clock domain *B*

TABLE 11.5
Slave IOs Description

Name	Direction	Bit Width	Description
clk_b	Input	1	Clock, rising edge, 50 MHz
rst_b	Input	1	Asynchronous reset, 0 valid
gnt_b	Output	1	Grant signal from clock domain *B*
req_a	Input	1	Request signal from clock domain *A*

- B. Timing Diagram for Clock Domain Crossing
  - In the initial cycle of ``clk_a", the pulse signal ``en_a" triggers the commencement of the request-grant handshake. Beginning in the subsequent ``clk_a" cycle, the master transmits the request signal (``req_a") and awaits the response from the slave.
  - Moving to the slave side, the detection of the request signal ``req_a" takes place during the second ``clk_b" cycle. To prevent any potential race conditions, a delay of two ``clk_b" cycles is introduced for the input ``req_a", occurring across cycles 2 and 3 within the clock B domain. It is in the third cycle of ``clk_b" that the output ``gnt_b" materializes, utilizing the synchronized signal ``req_b_d2".

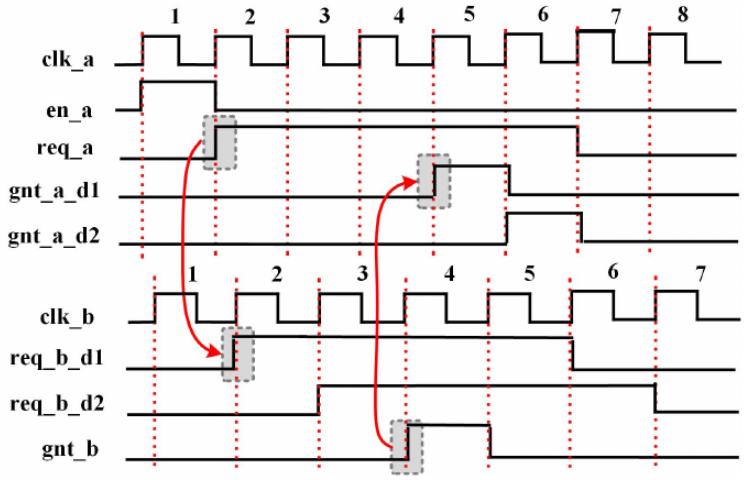


FIGURE 11.16
Timing Diagram of Request-Grant Handshaking Crossing Asynchronous Clock Domains

- B. Timing Diagram for Clock Domain Crossing
  - In subsequent steps, the ``gnt_b'' input transitions into the clock A domain, demanding a delay spanning two ``clk_a'' cycles encompassing cycles 5 and 6. By the sixth cycle of ``clk_a'', the synchronized signal ``gnt_a_d2'' makes its entrance, effectively pulling down the initial request ``req_a'' signal and formally concluding the handshaking procedure.

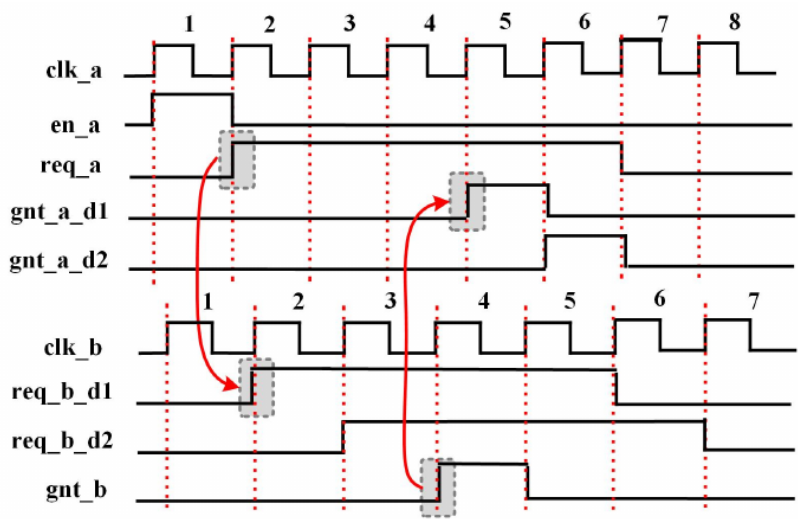
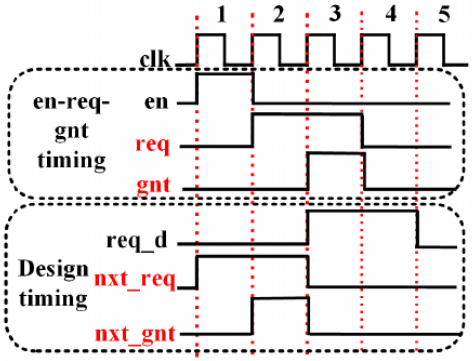


FIGURE 11.16
Timing Diagram of Request-Grant Handshaking Crossing Asynchronous Clock Domains

- 11.6.3 Design on Request-Grant Handshaking
  - The enable signal ``en'' responsible for initiating the handshake
  - The ``nxt_req'' and ``nxt_gnt'' signals that showcase the combinational logic used to determine the timing.
    - The ``nxt_req'' signal plays a crucial role in determining the timing of a request. It initiates when the ``en'' signal is activated and ceases when the ``gnt'' signal is detected.
    - The ``nxt_gnt'' signal plays a crucial role in determining the timing of the grant. It becomes active immediately upon the assertion of the request, assuming that the slave is always available for the handshake.



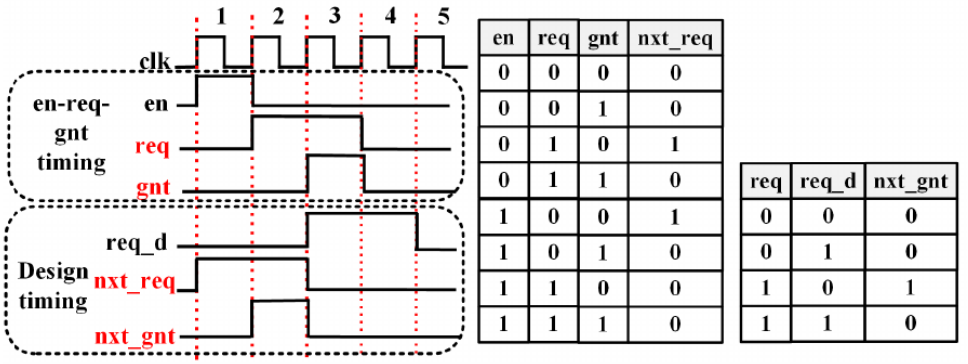
en	req	gnt	nxt_req
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

req	req_d	nxt_gnt
0	0	0
0	1	0
1	0	1
1	1	0

(a) Timing Diagram of en-req-gnt Design (b) Truth Table of Request (c) Truth Table of Grant

FIGURE 11.17
Timing Diagram of Request-Grant Handshaking

- 11.6.3 Design on Request-Grant Handshaking
  - The design circuit involving both the master and slave.



en	req	gnt	nxt_req
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

req	req_d	nxt_gnt
0	0	0
0	1	0
1	0	1
1	1	0

(a) Timing Diagram of en-req-gnt Design (b) Truth Table of Request (c) Truth Table of Grant

FIGURE 11.17
Timing Diagram of Request-Grant Handshaking

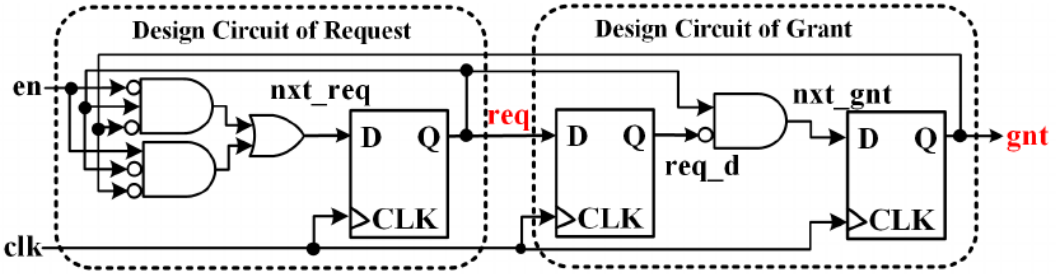


FIGURE 11.18
Block Diagram of Request-Grant Handshaking in Synchronous Clock Domains

- 11.6.3 Design on Request-Grant Handshaking
  - Verilog descrip

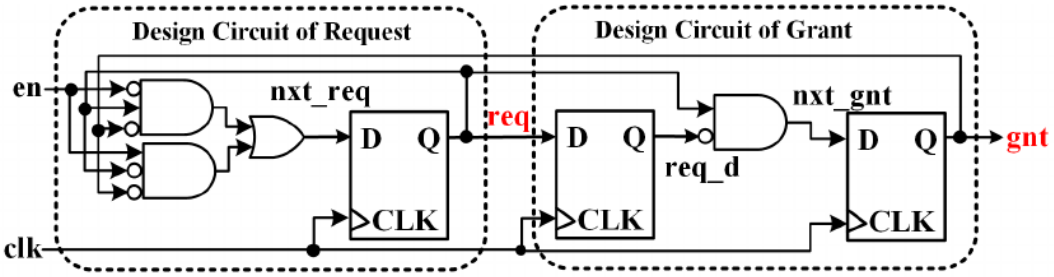
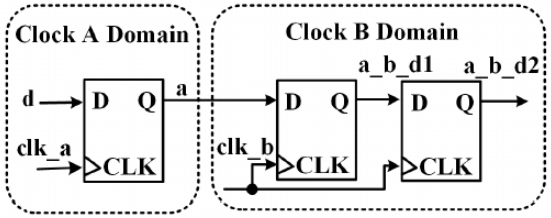


FIGURE 11.18
Block Diagram of Request-Grant Handshaking in Synchronous Clock Domains

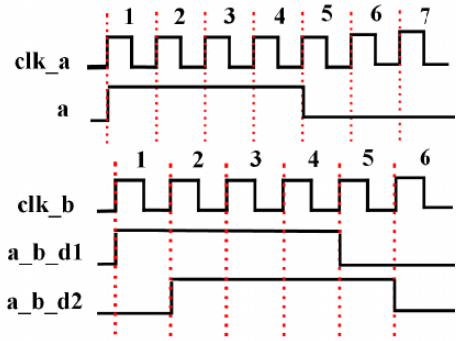
```
    req    <= nxt_req;
  end
end
endmodule

// slave circuit for response
module slave (input       clk,
              input       rst,
              input       req,
              output reg  gnt);
reg req_d;
wire nxt_gnt = req & ~req_d ;
always @(posedge clk, negedge rst) begin
  if (~rst) begin
    gnt    <= 1'b0;
    req_d <= 1'b0;
  end else begin
    gnt    <= nxt_gnt;
    req_d <= req    ;
  end
end
endmodule
```

```
// master circuit for request
module master(input       clk,
              input       rst,
              input       en ,
              input       gnt,
              output reg req);
wire nxt_req = (~en & req & ~gnt) | (en & ~req & ~gnt);
always @(posedge clk, negedge rst) begin
  if (~rst) begin
    req    <= 1'b0;
  end else begin
```

- 11.6.4 Synchronizer Design



(a) Block Diagram of Signal Crossing Different Clock Domain



(b) Timing Diagram of Signal Crossing Different Clock Domain

FIGURE 11.19
Block Diagram and Timing Diagram of Synchronizer Design

```
1  always @(posedge clk, negedge rst) begin
2    if (~rst) begin
3      req_b_d1 <= 1'b0;
4      req_b_d2 <= 1'b0;
5    end else begin
6      req_b_d1 <= req     ;
7      req_b_d2 <= req_b_d1;
8    end
9  end
```

- Design Rules -- Clock Domain Crossing

Design Rules – Clock Domain Crossing

1. Clock domain crossing refers to the process of transferring data from one source clock domain to another destination clock domain. To ensure reliable transfer, a synchronizer is typically employed, comprising two series registers. Although the output from the first register may potentially enter a metastable state for a certain period, the two-clock-cycle delay helps increase the likelihood of stabilization to a reliable level.
2. It is important to note that a synchronizer is specifically designed for scenarios involving clock domain crossings where the signal can be consistently sampled by at least one clock edge of the destination clock. If the signal does not meet this criterion, alternative methods, such as utilizing asynchronous FIFOs, should be considered to ensure reliable data transfer across clock domains.

- 11.6.5 System Integration for Request-Grant Handshaking and CDC

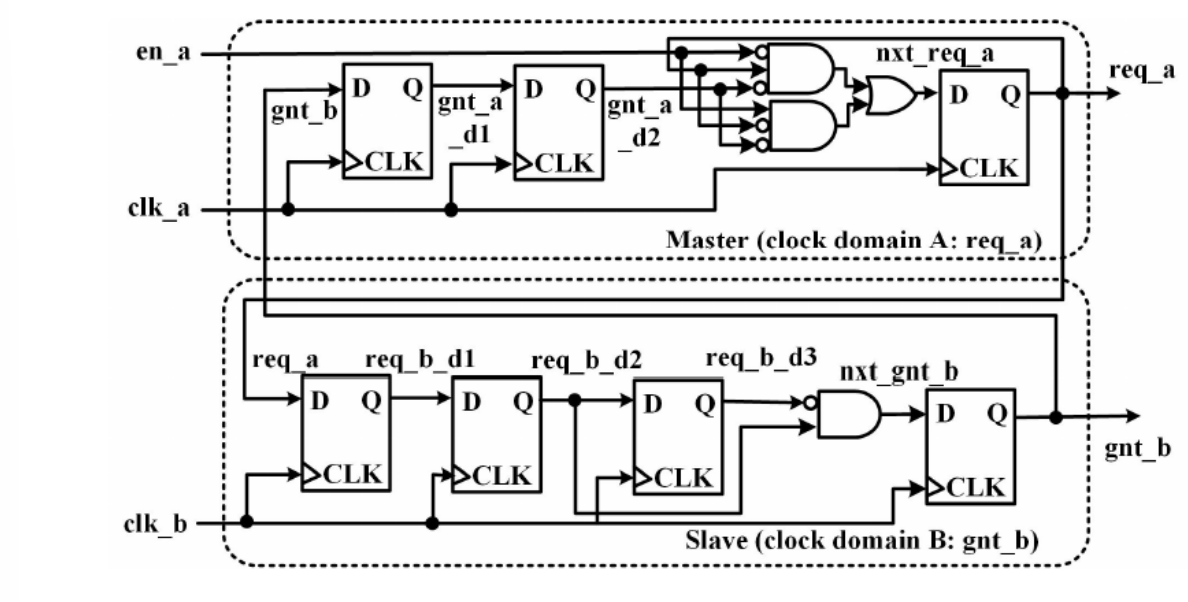


FIGURE 11.20
Block Diagram of Request-Grant Handshaking in Asynchronous Clock Domains