System IC Design: Timing Issues and DFT

Hung-Chih Chiang

Institute of Communications Engineering
National Sun Yat-sen University
Outline

- SoC Timing Issues
  - Timing terminologies
  - Synchronous vs. asynchronous design
  - Interfaces and timing closure
  - Clock issues
  - Reset

- Design for Testability (DFT)
  - SoC test plan
  - Scan, ATPG, DFT design rules
  - Embedded memory test, embedded core test
SoC Clock Issues
SoC Clock Domains

OSC → Clock Generator
CLK1 → Clock Generator
CLK1 → CLK5
CLK2 → Clock Generator
CLK2 → CLK6
CLK3 → Clock Generator
CLK3
CLK4 → Clock Generator
CLK4 → CLK6
CLK5 → Clock Generator
Timing Terminologies

- **Cell timing specification**
  - Setup time, hold time, release time, width, period and skew

- **Max. clock frequency, timing closure**
  - Cell delay and wire delay

- **Environments and process variations**
  - Simulation best case, typical case, worst case and pseudo worst case

- **Cell delay and wire delay affecting factors**
  - Loading and driving capacity
Basic Cell Timings and Delays

- **D**: Data Input
- **Q**: Output
- **CK**: Clock Input
- **QN**: Complement of Clock Input
- **RN**: Complement of Data Input

**Interconnection Delay**

**Cell Delay**

**Setup**

**Hold**

**Recovery**

**Width**

**Skew**

**Period**
Simulation Cases

- **Best Case**
  highest operation voltage, lowest temperature, fast process, eg. 0.25\(\mu \text{m}@2.75\text{V}, 0^\circ\text{C}, \text{fast process}

- **Typical Case**
  standard operation voltage, room temperature, typical process, eg. 0.25\(\mu \text{m}@2.5\text{V}, 25^\circ\text{C}, \text{typical process}

- **Pseudo Worst Case**
  lowest operation voltage, highest temperature, typical process, eg. 0.25\(\mu \text{m}@2.25\text{V}, 125^\circ\text{C}, \text{typical process}

- **Worst Case**
  lowest operation voltage, highest temperature, slow process, eg. 0.25\(\mu \text{m}@2.25\text{V}, 125^\circ\text{C}, \text{slow process}
Loading and Cell Delay

Linear Delay Model

\[ t_{\text{typical}} = t_{\text{intrinsic}} + (K_{\text{load}} \times C_{\text{load}}) \] (Databook)

Non-Linear Delay Model (Table Lookup)

\[ t_{\text{typical}} = F(t_{\text{rf}}, C_{\text{load}}) \] (EDA Timing Model)
Cell Datasheet: NAND2 (1)

Cell Description
The NAND2 cell provides the logical NAND of two inputs (A, B). The output (Y) is represented by the logical equation:

\[ Y = A \cdot B \]

Logic Symbol

Function Table

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>x</td>
<td>1</td>
</tr>
<tr>
<td>x</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Cell Size

<table>
<thead>
<tr>
<th>Drive Strength</th>
<th>Height (μm)</th>
<th>Width (μm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAND2XL</td>
<td>8.0</td>
<td>3.2</td>
</tr>
<tr>
<td>NAND2X1</td>
<td>8.0</td>
<td>3.2</td>
</tr>
<tr>
<td>NAND2X2</td>
<td>8.0</td>
<td>4.8</td>
</tr>
<tr>
<td>NAND2X4</td>
<td>8.0</td>
<td>6.4</td>
</tr>
</tbody>
</table>
## Cell Datasheet: NAND2 (2)

### AC Power

<table>
<thead>
<tr>
<th>Pin</th>
<th>Power (µW/MHz)</th>
<th>Pin</th>
<th>Capacitance (pF)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>XL</td>
<td>X1</td>
<td>X2</td>
</tr>
<tr>
<td>A</td>
<td>0.0134</td>
<td>0.0270</td>
<td>0.0510</td>
</tr>
<tr>
<td>B</td>
<td>0.0158</td>
<td>0.0336</td>
<td>0.0647</td>
</tr>
</tbody>
</table>

### Delays @ 25°C, 2.5V, Typical Process

<table>
<thead>
<tr>
<th>Description</th>
<th>Intrinsic Delay</th>
<th>K&lt;sub&gt;load&lt;/sub&gt; (ns/pF)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>XL</td>
<td>X1</td>
</tr>
<tr>
<td>A→Y ↑</td>
<td>0.059</td>
<td>0.038</td>
</tr>
<tr>
<td>A→Y ↓</td>
<td>0.051</td>
<td>0.038</td>
</tr>
<tr>
<td>B→Y ↑</td>
<td>0.072</td>
<td>0.047</td>
</tr>
<tr>
<td>B→Y ↓</td>
<td>0.059</td>
<td>0.046</td>
</tr>
</tbody>
</table>

### Pin Capacitance
Cell Datasheet: DFF (1)

Cell Description
The DFF cell is a positive-edge triggered static D-type flip-flop.

Logic Symbol

Function Table

<table>
<thead>
<tr>
<th>D</th>
<th>CK</th>
<th>Q[n+1]</th>
<th>QN[n+1]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>↑</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>↑</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>x</td>
<td>↓</td>
<td>Q[n]</td>
<td>QN[n]</td>
</tr>
</tbody>
</table>

Cell Size

<table>
<thead>
<tr>
<th>Drive Strength</th>
<th>Height (µm)</th>
<th>Width (µm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFFXL</td>
<td>8.0</td>
<td>14.4</td>
</tr>
<tr>
<td>DFFX1</td>
<td>8.0</td>
<td>14.4</td>
</tr>
<tr>
<td>DFFX2</td>
<td>8.0</td>
<td>15.2</td>
</tr>
<tr>
<td>DFFX4</td>
<td>8.0</td>
<td>21.6</td>
</tr>
</tbody>
</table>
## Cell Datasheet: DFF (2)

### AC Power

<table>
<thead>
<tr>
<th>Pin</th>
<th>Power (mW/MHz)</th>
<th>Power (mW/MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>XL</td>
<td>X1</td>
</tr>
<tr>
<td>D</td>
<td>0.0604</td>
<td>0.0856</td>
</tr>
<tr>
<td>CK</td>
<td>0.0712</td>
<td>0.0984</td>
</tr>
<tr>
<td>Q</td>
<td>0.0738</td>
<td>0.1311</td>
</tr>
</tbody>
</table>

### Pin Capacitance

<table>
<thead>
<tr>
<th>Pin</th>
<th>Capacitance (pF)</th>
<th>Capacitance (pF)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>XL</td>
<td>X1</td>
</tr>
<tr>
<td>D</td>
<td>0.0604</td>
<td>0.0856</td>
</tr>
<tr>
<td>CK</td>
<td>0.0712</td>
<td>0.0984</td>
</tr>
<tr>
<td>Q</td>
<td>0.0738</td>
<td>0.1311</td>
</tr>
</tbody>
</table>

### Delays @ 25°C, 2.5V, Typical Process

<table>
<thead>
<tr>
<th>Description</th>
<th>Intrinsic Delay</th>
<th>K_{load} (ns/pF)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>XL</td>
<td>X1</td>
</tr>
<tr>
<td>CK→Q ↑</td>
<td>0.515</td>
<td>0.346</td>
</tr>
<tr>
<td>CK→Q ↓</td>
<td>0.573</td>
<td>0.289</td>
</tr>
<tr>
<td>CK→Q ↑</td>
<td>0.635</td>
<td>0.388</td>
</tr>
<tr>
<td>CK→Q ↓</td>
<td>0.686</td>
<td>0.474</td>
</tr>
</tbody>
</table>
## Timing constraints @ 25°C, 2.5V, Typical Process

<table>
<thead>
<tr>
<th>Pin</th>
<th>Requirement</th>
<th>Interval (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>XL</td>
</tr>
<tr>
<td>D</td>
<td>setup ↑ → CK</td>
<td>0.09</td>
</tr>
<tr>
<td></td>
<td>setup ↓ → CK</td>
<td>0.28</td>
</tr>
<tr>
<td></td>
<td>hold ↑ → CK</td>
<td>-0.07</td>
</tr>
<tr>
<td></td>
<td>hold ↓ → CK</td>
<td>-0.10</td>
</tr>
<tr>
<td>CK</td>
<td>Minpwh</td>
<td>0.50</td>
</tr>
<tr>
<td></td>
<td>Minpwl</td>
<td>0.60</td>
</tr>
</tbody>
</table>
Outline

- SoC Timing Issues
  - Timing terminologies
  - Synchronous vs. asynchronous design
  - Interfaces and timing closure
  - Clock issues
  - Reset

- Design for Testability (DFT)
  - SoC test plan
  - Scan, ATPG, DFT design rules
  - Embedded memory test, embedded core test
Synchronous vs. Asynchronous Design

- **Synchronous Design**
  - Flip-flop based (clock based)
  - Easy timing handling
  - DFT compliant

- **Asynchronous**
  - Latch based
  - Timing ambiguity causes problems
  - Not DFT compliant
Flip-Flop (Clock) Based Design

- Poor HDL coding of combinational logics can produce unintentional latches
- Avoid using flip-flops with enable input
- Use positive edge triggered flip-flops for module RTL coding if flip-flops in cell library is triggered at positive clock edge
If negative edge triggered flip-flops are required in a design while Cell Library contains positive edge triggered flip-flops, invert the clock phase first and then write RTL codes using positive edge triggered flip-flops to avoid inverters being inserted at clock inputs of each module during logic synthesis.
Clock-Based Timing (single clock source)

\[ t_{\text{hold}} < d_1 + d_2 < T_{\text{CK}} - t_{\text{setup}} \]

Assuming all clocks arrive at the same time

Must identify multi-cycle paths and asynchronous signals during logic synthesis!
Problem of Latch: possible D/E race

- Need to ensure that there will be enough hold time for stable D after the falling edge of E
Problem of Latch: timing ambiguity
Outline

- **SoC Timing Issues**
  - Timing terminologies
  - Synchronous vs. asynchronous design
  - Interfaces and timing closure
  - Clock issues
  - Reset

- **Design for Testability (DFT)**
  - SoC test plan
  - Scan, ATPG, DFT design rules
  - Embedded memory test, embedded core test
A proper design of block interfaces makes timing closure a local problem.

A major timing issue in deep submicron technology is the wire delay due to wire load capacitance and RC delay can be much larger than intrinsic cell delays.

Timing driven APR helps deal with this problem by taking into account the wire load model.

Physical synthesis takes a further stride in achieving timing closure by combining synthesis and timing driven placement.
Both inputs and outputs should be registered.
This gives a full clock cycle to propagate the outputs of one macro to inputs of another.
Any block that is synthesized as a unit should have its own outputs registered.

Any block that is floor-planned as a unit should have its own inputs and outputs registered.
Example: interface specification

CK

3ns

3ns

DT

Don’t care

Valid

Don’t care
Example: registered vs. unregistered inputs

\[ d_1 + t_{\text{setup}} < 3\text{ns} \]

\[ d_2 + t_{\text{setup}} < 3\text{ns} \]
Outline

- SoC Timing Issues
  - Timing terminologies
  - Synchronous vs. asynchronous design
  - Interfaces and timing closure
  - Clock issues
  - Reset

- Design for Testability (DFT)
  - SoC test plan
  - Scan, ATPG, DFT design rules
  - Embedded memory test, embedded core test
Clocking Issues

- Clock skew and clock tree
- Divided clocks
- Asynchronous clock interface
- Clock gating
- Synchronize Hard IP
- Other considerations
Clock Skew

Din → FF0/CK → FF0 CK → D Q

Combinational

FF1/CK → CK → FF1

Skew
Clock Skew May Cause Errors

Din → D0 → FF0/CK → Q0 → FF1/CK → D1 → Q1 → Din

Din
FF0/CK
Q0
FF1/CK
Q1
Insert clock tree during APR

Clock tree can significantly increase power consumption
Clock Tree Example

Δ + Skew
Clock Tree Example
Divided Clocks

Clock Generator

- CK0 (f Hz) → Module A
- CK1 (f/2 Hz) → Module B
- Ck2 (f/4 Hz) → Module C

Module A: $\Delta_1 + \text{skew}_1$
Module B: $\Delta_0 + \text{skew}_0$
Module C: $\Delta_2 + \text{skew}_2$

Clock cycles:
- CK0
- CK1
- CK2

$t_0 \rightarrow t_1 \rightarrow t_2$
An Alternative Design Approach for a Divided Clock Domain

Clock Generator

Δ+skew

Module A

Module B

Module C

CK

En1

En2
Asynchronous Clock Interface

- Dangerous design!!
- Random logic errors may occur due to the delay time difference between $d_a$ and $d_b$. 
Asynchronous Errors

CK2

X

Y (Da)

Z (Db)

CK1a/CLK1b

QaQb:

E.g.

\( X = 0 \rightarrow 01 \rightarrow 10 \rightarrow \overline{10} \rightarrow 01 \rightarrow X = 0 \overline{0} \)

\( X = 1 \rightarrow 01 \rightarrow 11 \rightarrow \overline{11} \rightarrow 01 \rightarrow X = 1 \overline{1} \)
Not all asynchronous inputs need to be synchronized!

A single flip-flop may not be good enough for clock synchronization.
- CK=‘L’: T1 & T4 on; T2 & T3 off
- CK=‘H’: T1 & T4 off; T2 & T3 on
Metastability

X
Asynchronous to CK1

CK1

D Q

Y
Synchronous to CK1

CK1

X

Y
Two staged flip-flop to reduce the probability of metastability
Dual Flip-Flop Synchronization

![Diagram of dual flip-flop synchronization with inputs X, Y, and Z, and clocks Ck1 and Ck2.]

Block A

CK1

X

Y

Z
Peak Power Reduction

[Diagram showing clock generator and interfaces labeled as Sync. I/F and Async. I/F]
Clock Gating

Not recommended! Increase difficulties for synthesis and APR tools.
Clock Gating For Low Power Design

Clock

Enable

Clock Generator

Module A

Module B

Module C
Clock Delays For Hard Blocks

- Take into account insertion delays of hard macros
Clock Planning Guidelines

- The system clock generation and control logic should be separate from all function blocks of the system.

- Document clock domain information:
  - frequencies, PLL
  - interface timing (input and output)
  - skew requirement among clocks

- Use the standard synchronization interface for asynchronous inputs.

- Compensate insertion delays of hard macros.

- Bypass clock gating and PLL in test mode.
Outline

- SoC Timing Issues
  - Timing terminologies
  - Synchronous vs. asynchronous design
  - Interfaces and timing closure
  - Clock issues
  - Reset

- Design for Testability (DFT)
  - SoC test plan
  - Scan, ATPG, DFT design rules
  - Embedded memory test, embedded core test
Chip Reset Issues

- Synchronous or Asynchronous?
- External or Internal Power On Reset?
- Voltage Detector for Power Down Reset?
- Hard Reset and Soft Reset?
- Each Module Individually Resettable for Debugging Purposes?
Synchronous Reset

- Easy to synthesize since reset is treated as a logic signal
- Reset slightly affect data timing
- Need at least one active clock edge for reset to take place. This could become a problem upon power on
// Sync. Reset (Verilog)
always@(psoedge Clk)
if (~Rst_n)
begin
  A <= ...;
  B <= ...;
end
else
begin
  ..........;
  ..........;
End

// Sync. Reset (VHDL)
library IEEE;
use IEEE.std_logic_1164.all;
.........
process(Clk)
begin
  if rising_edge(Clk) then
    if (Rst_n='0') then
      A <= ...;
      B <= ...;
    else
      ............;
      ............;
    end if;
  end if;
end process;
Asynchronous Reset

- No clock required during reset period
- Reset does not affect data timing
- Like clock, a reset tree is usually required during APR
Verilog/VHDL for Asynchronous Reset

// Async. Reset (Verilog)
always@(psoedge Clk or negedge Rst_n)
if (~Rst_n)
begin
    A <= ...;
    B <= ...;
end
else
begin
    ........;
    ........;
end

// Async. Reset (VHDL)
library IEEE;
use IEEE.std_logic_1164.all;
.........
process(Clk, Rst_n, ..)
begin
    if (Rst_n='0') then
        A <= ...;
        B <= ...;
    elseif rising_edge(Clk) then
        ........;
        ........;
    end if;
end process;
Synchronous or Asynchronous Reset?

- If properly designed, both synchronous and asynchronous reset schemes can work on most application systems. Synchronous reset requires additional latency, while asynchronous reset is more sensitive to system noise.

- Reset must be synchronously de-asserted so that all state machine flip-flops starts at the same active clock edge.

- All flip-flops/latches should be reset to a pre-defined state ("0" or "1") to avoid ambiguity voltage output of sequential elements.
Glitches Removing

A1 active low

A1

B1

C1

B1

C1

A2 active high

A2

B2

C2

B2

C2

Process, temperature and voltage dependent!
Synchronous Reset Architecture

Timing adjusted by synthesis tools
Asynchronous Reset Architecture

Reset timing budget: almost 1 full clock
Reset Buffer Tree with Registers

Synchronous reset

buffer tree

Reset_n

Rst_n

1 full clock

Timing adjusted by synthesis tools
Reset Buffer Tree with Registers

Asynchronous reset

buffer tree

Reset_n

1

Rst_n

1

1 full clock

1 full clock
Synchronous Reset for Multiple Clock Domain

Reset_n

Rst1_n

Clk1

buffer tree

Rst2_n

Clk2

buffer tree
Asynchronous Reset for Multiple Clock Domain

![Diagram showing the asynchronous reset for multiple clock domains](image-url)
Sequential Reset Releasing (Synchronous)

- Reset_n
- Rst1_n
- Rst2_n
- Clk1
- Clk2
Sequential Reset Releasing (Asynchronous)
Outline

- **SoC Timing Issues**
  - Timing terminologies
  - Synchronous vs. asynchronous design
  - Interfaces and timing closure
  - Clock issues
  - Reset

- **Design for Testability (DFT)**
  - SoC test plan
  - Scan, ATPG, DFT design rules
  - Embedded memory test, embedded core test
IC testing vs. verification
  - manufacturing defect vs. functional defect

Importance of IC testing
  - cost of RMA

Test phases:
  - wafer test (chip probing),
  - final test (packaged IC testing)

Test principle:
  - different kinds of blocks require different test strategies
  - use a test controller at top level as a sequencer
An Example of SoC Test Plan

- **Test Mode:**
  - Processor Test
  - RAM BIST
  - ROM Check Sum
  - SCAN/ATPG
  - Functional Test
  - Analog Macros
  - ...

- **Parallel Test**
  - Reduce test cost
  - Need to watch out for maximal testing power consumption

- **Sequential Test**
  - Fewer test pins
SCAN Chain

Mux-D Scan Cell

Reset_n

Scan_en

combinational logic
Advantage of ATPG Scan Test with respect to Functional Test:
- a much shorter pattern with a higher fault coverage

Test Steps
1. Reset whole chip, release reset
2. Enable scan mode, read out initial register values
3. Shift in a test vector
4. Disable scan mode, run one clock
5. Enable scan mode, shift out flip-flop contents and check results
6. Repeat 3~5 until finishing all test vectors
IEEE1149.1 Boundary Scan

- Chain ICs on a PCB for board level test.
- Enable connectivity test without sending test vectors to cores of all ICs.

**Boundary Scan Cell**

![Diagram of Boundary Scan Cell with signals DI, SI, Shift_Load, D, Q, CK, DO, SO, and Test_Normal]
Boundary Scan Architecture

Boundary Scan Cell
Avoid internally gated clocks or derived clocks
Provide test control for uncontrollable signals
Avoid feeding data path with clocks
Avoid using flip-flops with an enable input (synthesis)

Enable is not controllable!
Avoid internally generated asynchronous reset signals
Avoid using latches during logic synthesis

A latch can not be inserted into a scan chain due to the uncontrollable enable input
Avoid combinational feedback

Race/unstable! ATPG is not applicable!
Embedded Memory Test Schemes

- **Direct Access**
  - Simple circuits; flexible test patterns; possible to test memory at higher than normal operation frequency.
  - Extra test pins needed

- **Embedded Process Program**
  - Minimal hardware cost
  - Slow test speed; test patterns fixed within processor codes

- **Built-in Self-Test**
  - High efficient
  - Highest hardware cost among the three test schemes
Direct Access Memory Test

Test_In

Test_Out

Embedded Memory

Test_Ctrl

Memory

Memory
Typical Memory BIST Architecture

- **Pattern Generator**
  - A/Di/CE
  - Clk
  - Test_ctrl

- **Memory Module**
  - A/Di/CE
  - Clk

- **Compressor**
  - A/Di/CE
  - Do
  - Done
  - Result

- **ROM:** Read only -> Linear Feedback Shift Register, Check SUM
- **RAM:** Read/Write Patterns
Shared RAM BIST Circuits

Pattern Generator

Memory Module 1

A₁/Đ₁/Clk₁

CE₁

Memory Module 2

A₂/Đ₂/Clk₂

CE₂

Test_ctrl

Compressor

Do₁

Do₂

Done

Result

A₁/Đ₁/CE₁/Clk₁

A₂/Đ₂/CE₂/Clk₂
RAM Bist Algorithm: March14C+

1) Lowest -> Highest address, write 01010101… (0x55….)

2) Lowest -> Highest address, read 0x55, write 0xAA, read 0xAA

3) Lowest -> Highest address, read 0xAA, write 0x55, read 0x55

4) Highest -> Lowest address, read 0x55, write 0xAA, read 0xAA

5) Highest -> Lowest address, read 0xAA, write 0x55, read 0x55

6) Highest -> Lowest address, read 0x55

- 5 writes + 9 reads
- Write/read 0 and 1 at every bit
- High speed read -> write -> read
- High speed address increasing and decreasing
Scan Chain and Memory Block Interface

Combinational logic

DQ CK

Memory Module

Combinational logic

DQ CK

Combinational logic

ATPG_test

Combination logic

Combination logic
SoC Core Test Issues

- Shared test pin design and number of test pins

- Cores with different configurations such as scan cell type, scan chain length and test frequency

- Total test power consumption for parallel test.

- Total test cost (time)
IEEE1500 Embedded Core Test

Test Access Mechanism (TAM)

P1500 Core Test Wrapper
Core 1

P1500 Core Test Wrapper
Core k

Test Controller
Summaries

- Recommend flip-flop based design. Use Latches only under a well-controlled situation.

- A proper design of block interfaces makes timing closure a local problem.

- Clock domains require special cares.

- A global reset signal is recommended.

- A proper SoC test plan is important to reduce RMA costs.

- DFT rules must be followed to ensure the testability of designs.