University of Chinese Academy of Sciences

Verification & Implementation of SoC Design

Design Constraints and STA

Chun-Zhang Chen, Ph.D.

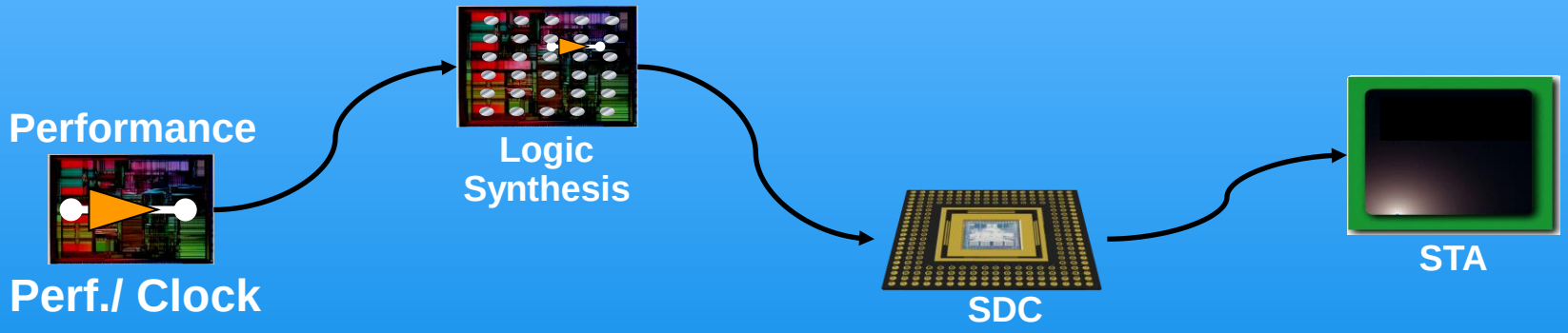
June 25-29, 2018

SDC and STA

- **Performance and Timing**
- Logic Synthesis and WLM
- SDC and STA
- SDC and Clock
- Discussion

Design Constraints and Design Performance

- “The faster, the better!”



“An integrated design methodology for logic synthesis and TTM”

Performance/Clock GHz → SDC/RTL2Gate

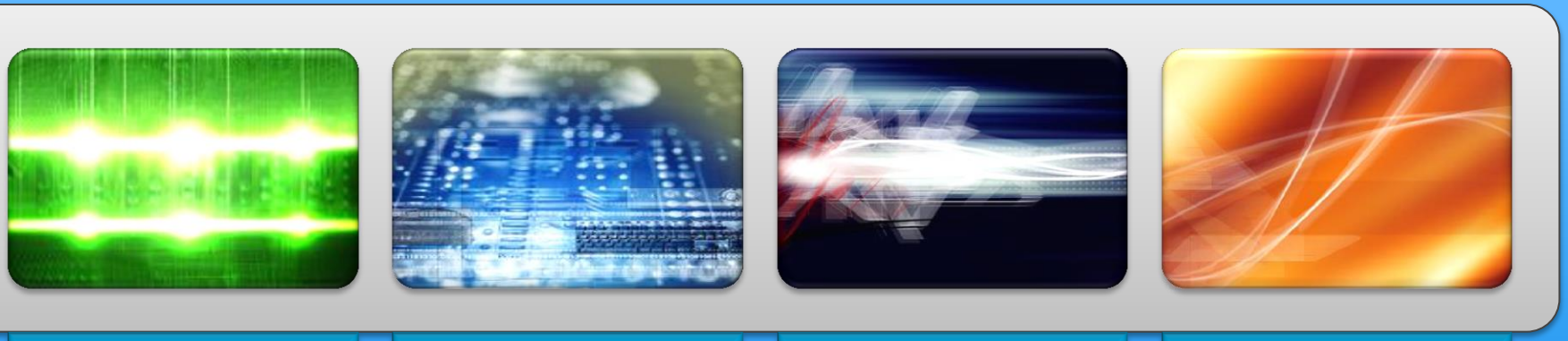
→ SDC/STA → STA/SI → SSTA

Clock Performance at Intel

Year		2010	2011	2012	2013	2014	2015	2016	2017	2020	2025
Area	Max. ASIC chip size per litho technology, mm^2	858	858	858	858	858	858	858	858	858	858
	MPU/ASIC (M1) 1/2 Pitch, nm	45	38	32	27	24	21	18.9	16.9	11.9	6.7
Power	High performance w/ heatsink, W	146	161	158	149	152	143	130	130	130	Inten. blank
	Vdd (high performance), V	0.97	0.90	0.87	0.85	0.82	0.80	0.77	0.75	0.68	0.59
Performance	On-chip local clock, GHz	*5.875**	3.744	3.894	4.050	4.211	4.380	4.555	4.737	5.329	6.483
	Functions per chip (million transistors)	10,323	14,599	20,646	29,198	36,787	46,348	58,395	73,573	147,147	467,162

11/19/00	***Pentinum 4***	***1.4 GHz***	***42,000,000***	***0.13 um***
01/07/02	Pentinum 4	2.2 GHz	55,000,000	0.13 um
08/26/02	Pentinum 4	2.8 GHz	> 55,000,000	0.13 um
06/2003	***Prescott***	***4.7 GHz***	***330,000,000***	***0.09 um***
'05-'10	??	? >5 GHz	> 500,000,000	< 0.09 um

Today's Synthesis Challenges



Performance & Capacity

- GigaGates
- GigaHertz
- Top-Down
- Hierarchical

Predictability

- Relative
- Up to HLS
- Down to P&R
- Run to Run
- Power

Turnaround Time

- ECO Complexity
- P&R Handoff
- Time to Market
- Formal Verification

Low Power

- Intent
- Analysis
- Simulation
- Verification
- Multi-Vt
- DVFS

To Meet Synthesis Goals

	Functional	Electrical	Physical
IP	ChipWare design IP library	Support for multiple libraries	Insertion of specific Low-Power & DFT IP (level shifters, ATPG engines, etc.)
SoC	Global Synthesis : Optimize for timing, area, and power in single pass, RTL& System Verilog support, design Intent capture,	Multi-VT, Multi-Supply Synthesis High Speed Logic Synthesis Low Power Synthesis	RTL Compiler with Physical: Physical-Aware congestion analysis, timing, clock gating, DFT, etc.

SDC and STA

- Performance and Timing
- Logic Synthesis and WLM
- SDC and STA
- SDC and Clock
- Discussion

Timing Optimization *Restructuring and IPO*

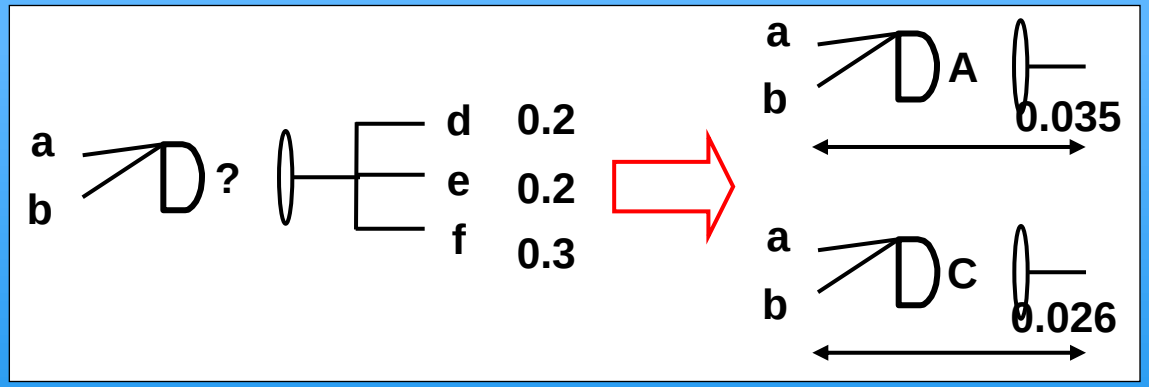


图6-38 IPO中的单元尺寸挑选法

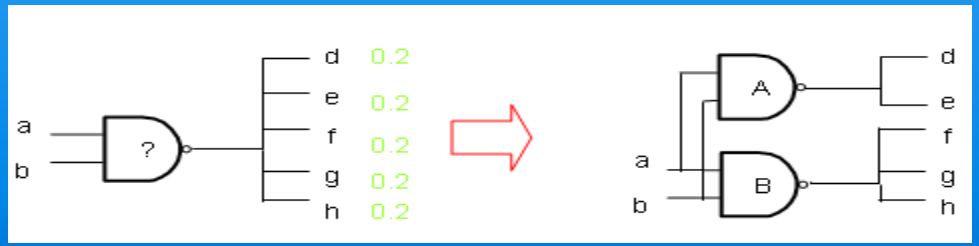


图6-39 IPO中复制单元法

Timing Optimization *Using Physical Synthesis*

- Timing Opt during synthesis
- Timing Opt during implementation
- Timing Opt using Physical Synthesis

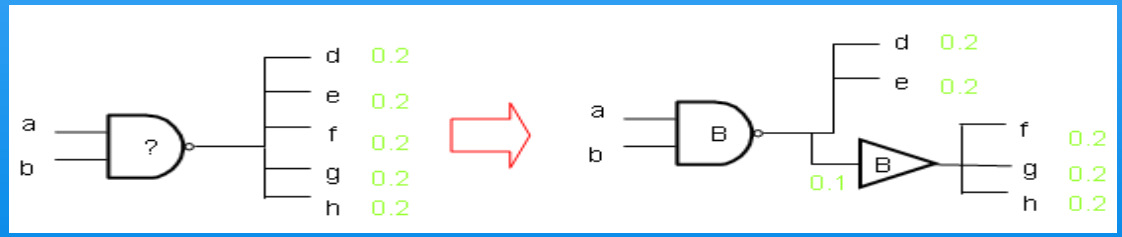


图6-40 IPO中的缓冲器法

Wire Load Models

- SWLMs, statistical, are based on averages over many similar designs using the same or similar physical libraries.
- SWLMs, structural, use information about neighboring nets, rather than just fanout and module size information.
- CWLMs, custom, are based on the current design after placement and routing, but before the current iteration of replacement synthesis.

```
wire_load("WLM1")     {
  resistance  :       0.0006      ;------>R per unit length
  capacitance :       0.0001      ;------> C per unit length
  area :              0.1         ;------> Area per unit length
  slope       :       1.5         ;------> Used for linear extrapolation
  fanout_length(1,  0.002)        ; ------> at fanout "1" length of the wire is 0.002

fanout_length(2,  0.006);
fanout_length(3,  0.009);
fanout_length(4,  0.015);
fanout_length(5,  0.020);

fanout_length(7,  0.028);         ------> at fanout "7" length of the wire is 0.028
fanout_length(8,  0.030);
fanout_length(9,  0.035);
fanout_length(10, 0.040);
}
```

Wire Load Model Flow

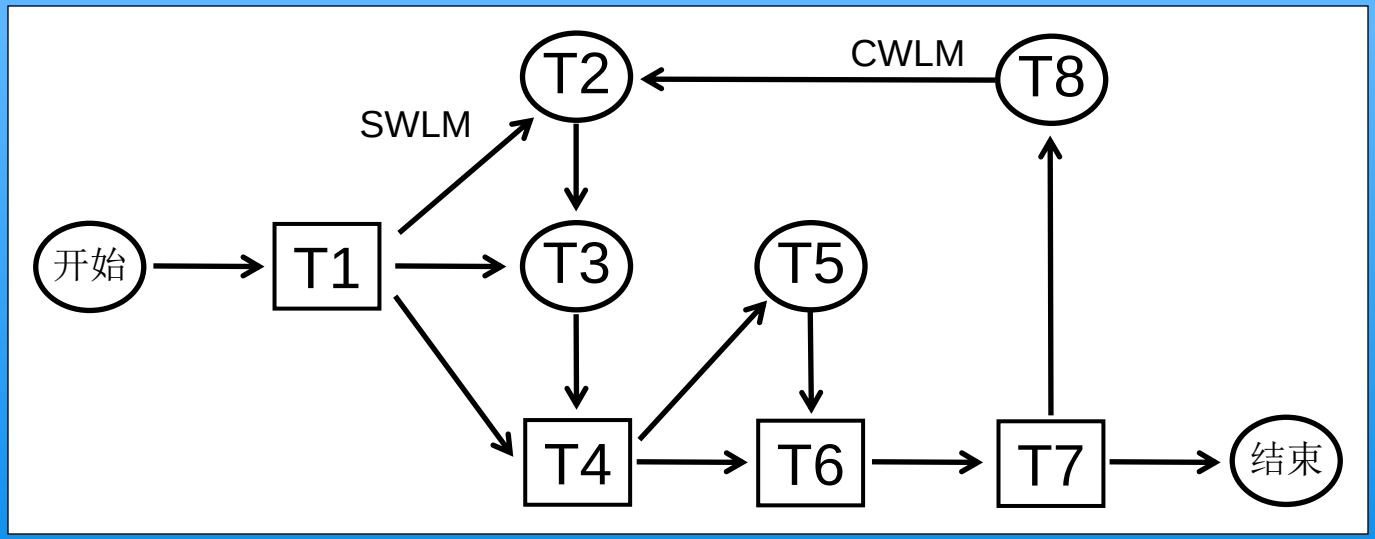


图6-4：WLM在物理实施流程中的应用

T1=synthesis, T2=read SWLM, T3=pre-placement opt, T4=placement;
T5=post-place opt, T6=global routing, T7=detail routing, T8=generate CWLM
Reference: Andrew B. Kahng and Stefanus Mantik

SDC and STA

- Performance and Timing
- Logic Synthesis and WLM
- SDC and STA
- SDC and Clock
- Discussion

The Basic: Timing Path and Timing Check

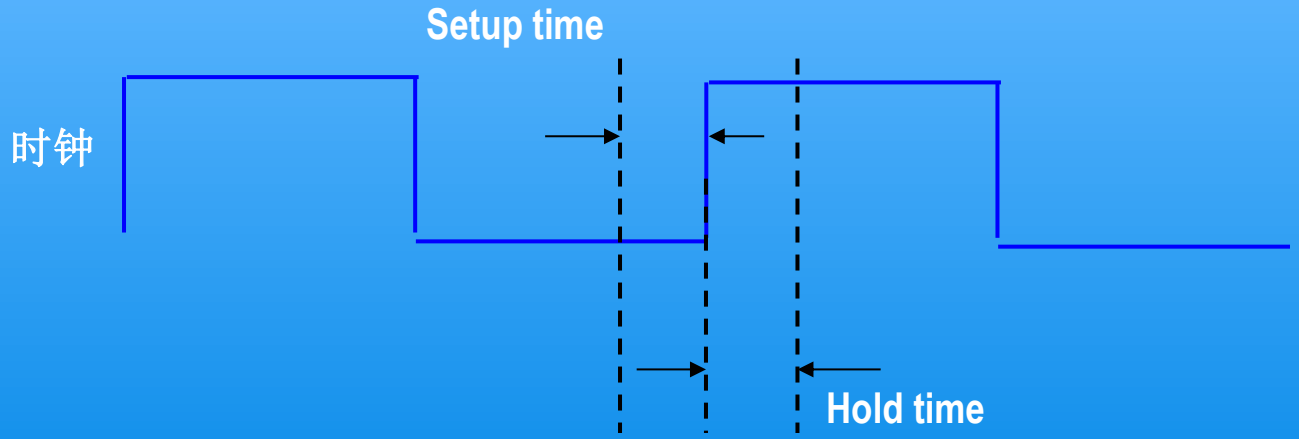
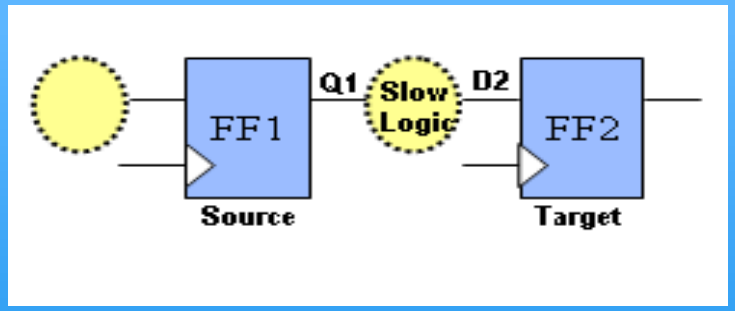
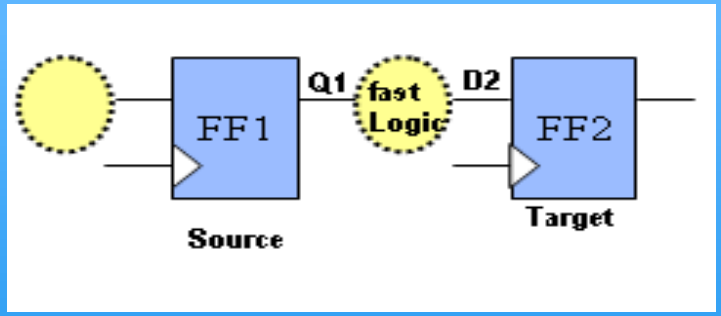


图6-20 时序路径中的时序检查

Static Timing Analysis

setup and hold





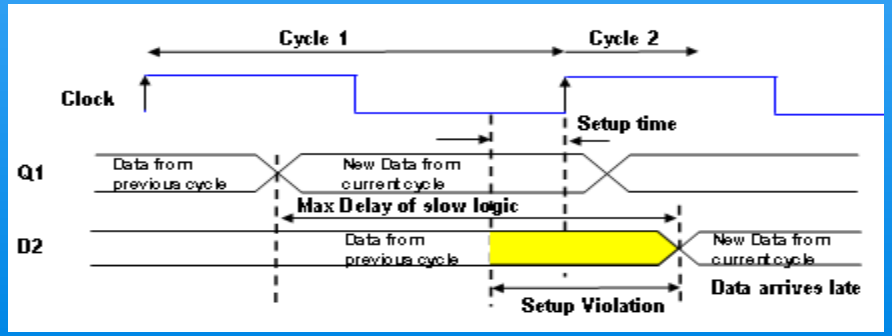


图6-23 分析setup 时序

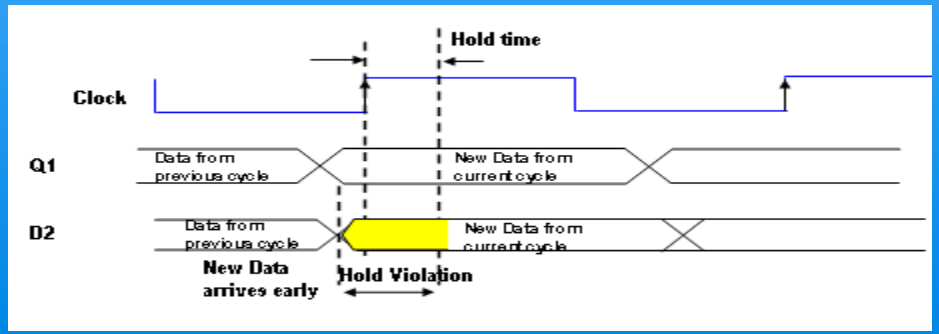


图6-24 分析hold 时序

Standard Design Constraints - SDC

表 6-3 设计约束类别（SDC 1.4版本）

设计环境约束：	时序约束：	set_false_path
set_drive	create_clock	set_max_delay
set_driving_cell	create_generated_clock	set_multicycle_path
set_fanout_load	set_clock_latency	
set_input_transition	set_clock_transition	必须的 8 条时序约束：
set_load	set_clock_uncertainty	create_clock
set_port_fanout_number	set_disable_timing	set_clock_uncertainty
	set_input_delay	set_input_delay
设计规则约束：	set_max_time_borrow	set_output_delay
set_max_capacitance	set_output_delay	set_load
set_max_fanout	set_propagated_clock	set_driving_cell
set_max_transition		set_operating_conditions
	时序特例：	set_wire_load_model

The Timing Path

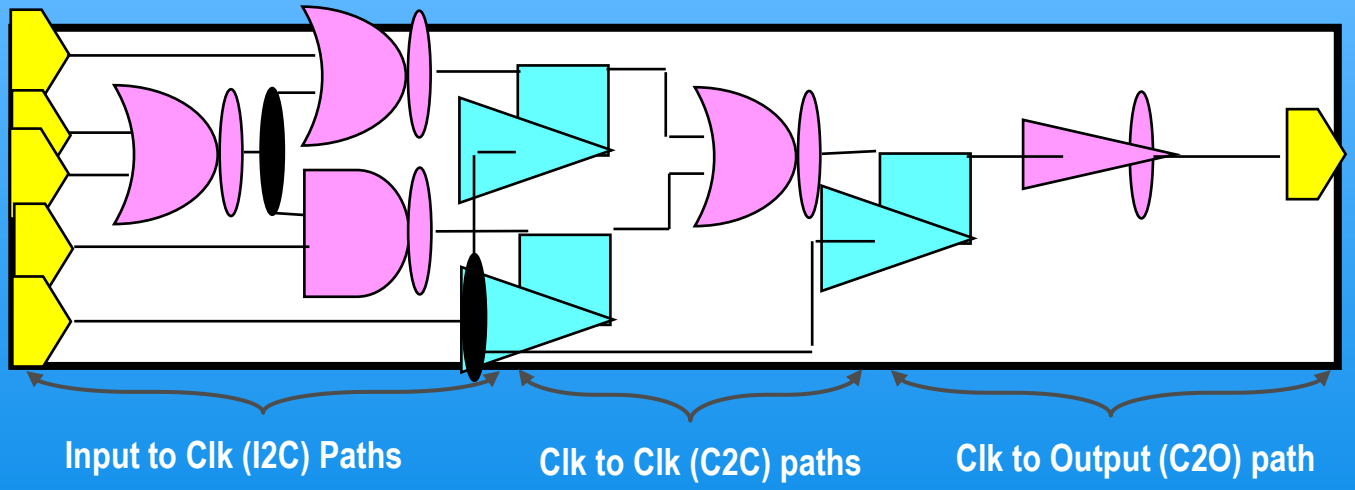


图6-18 时序的起点、终点和时序路径

Timing Path in Current Design of Interest

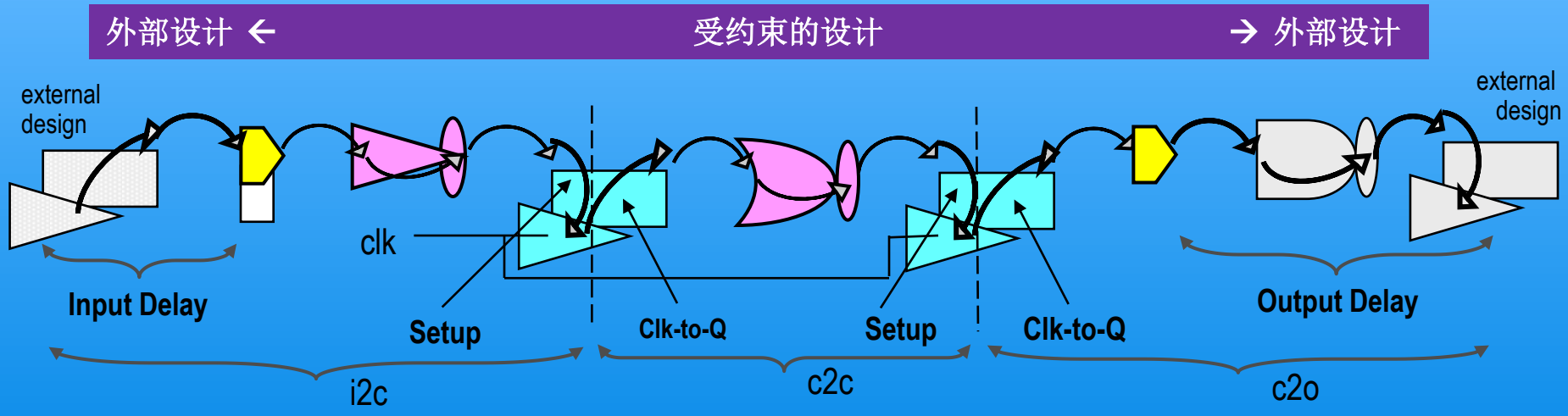


图6-19 时序路径的同步关系

Static Timing Analysis *types of timing path and definition*

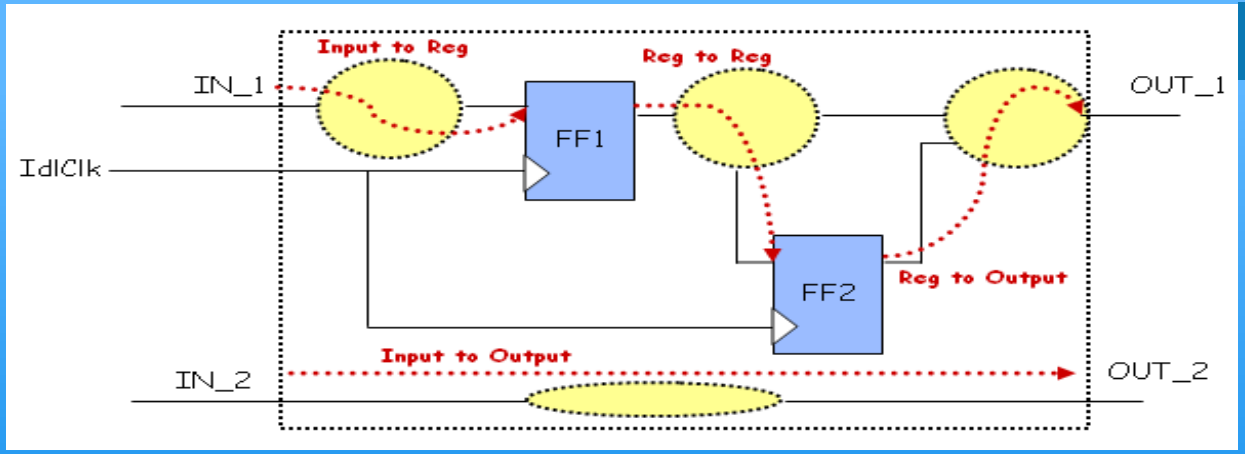


图6-25 时序路径的类型

表6-4 时序类型与时序要求的定义

类型	定义要求
R2R：	时钟
I2R：	时钟、数据到达时间
R2O：	时钟、输出延迟
I2O：	根据同步输入与输出的延迟分配时序要求

Types of Timing Paths – R2R *(Reg to Reg)*

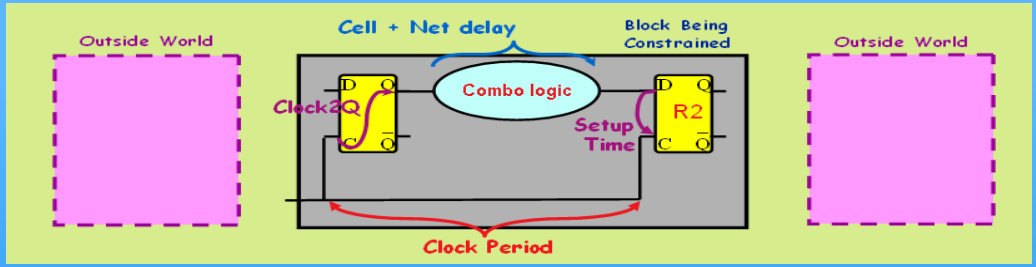
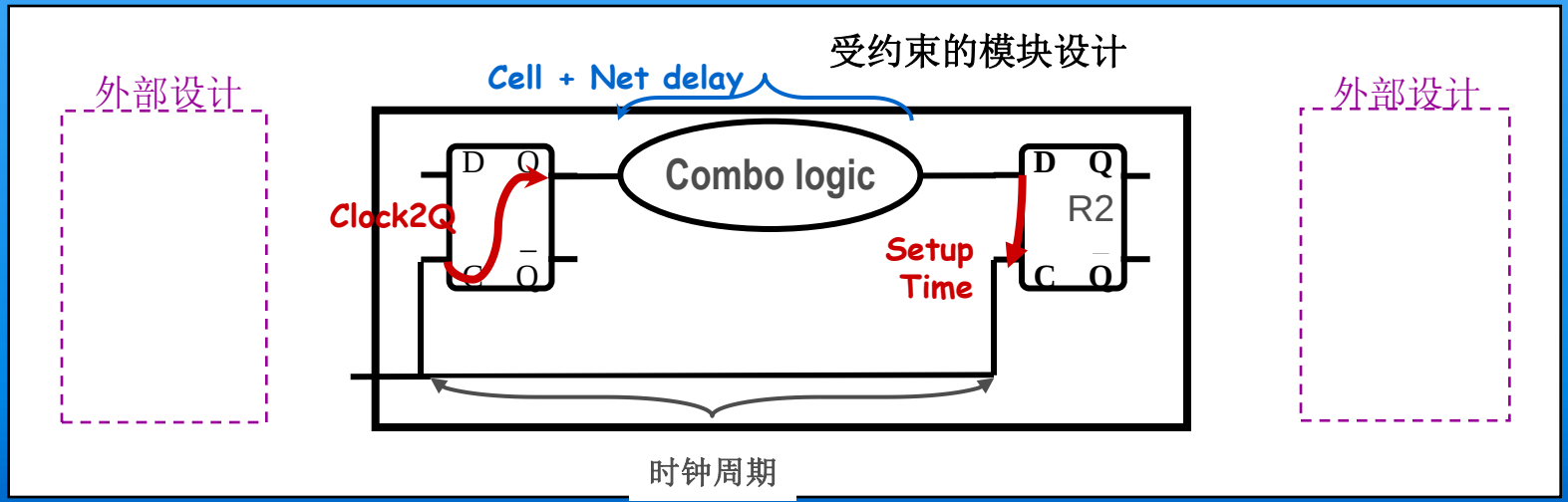


图6-26 时序路径的setup和hold分类分析R2R



Types of Timing Paths – I2R *(Input to Reg)*

- *Input_delay = clk_to_q + combo1*
- Setup requirement:

clk_launch_edge + max_Input_delay + max_combo2 =< clk_capture_edge – R2setup

- Hold requirement:

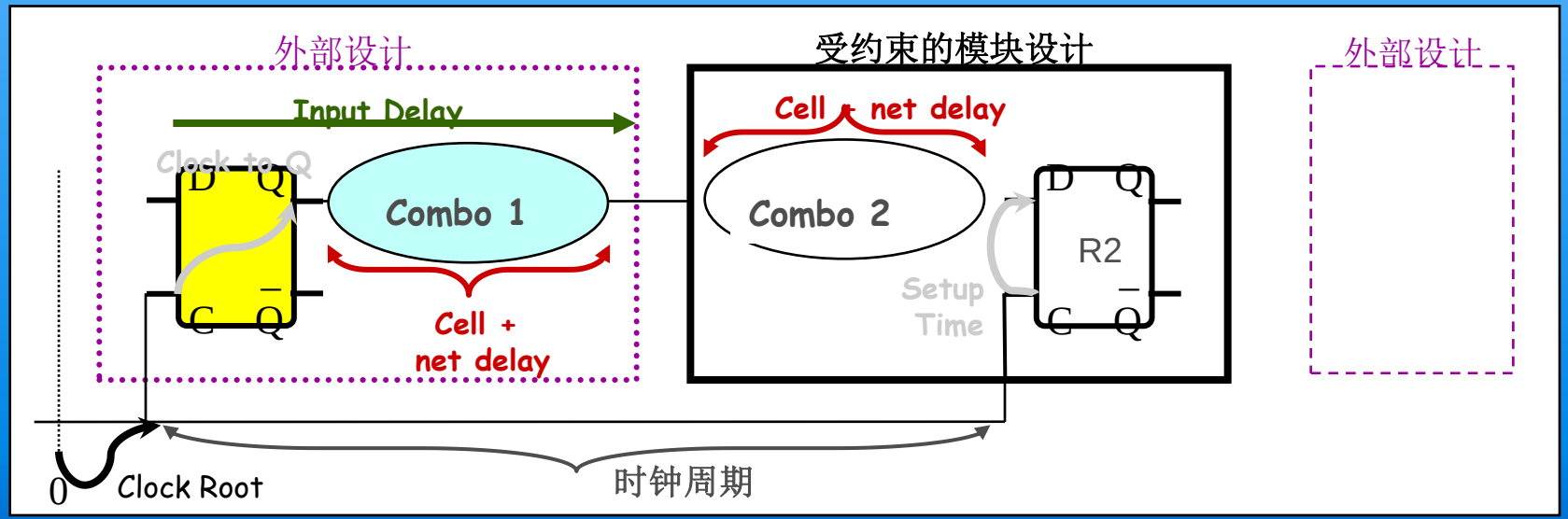


图6-27 时序路径的setup和hold分类分析I2R

Types of Timing Paths – R2O *(Reg to Output)*

ext_delay1 = external_cell + ext_net_delay + setupR2

ext_delay2 = holdR2 - min_external_cell - min_ext_net_delay

- Setup requirement:

  clk_launch_edge + clk_to_q + max_combo1 =< clk_capture_edge - ext_delay1

- Hold requirement:

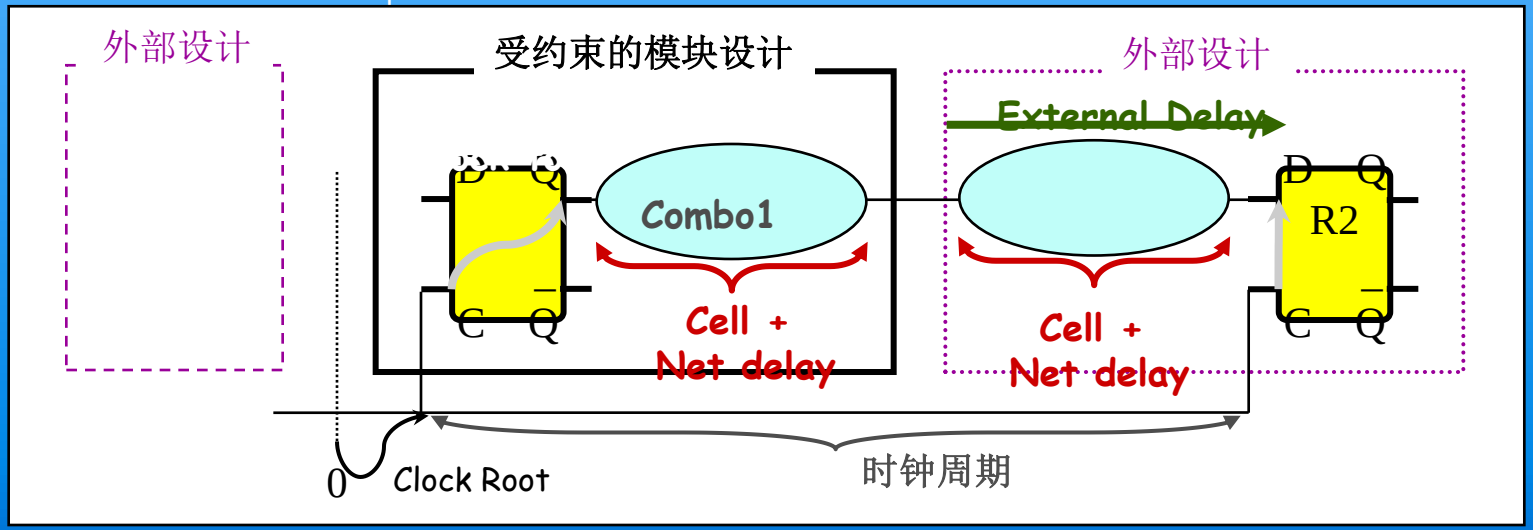


图6-28 时序路径的setup和hold分类分析R2O

Static Timing Analysis – I2O *(Input and Output)*

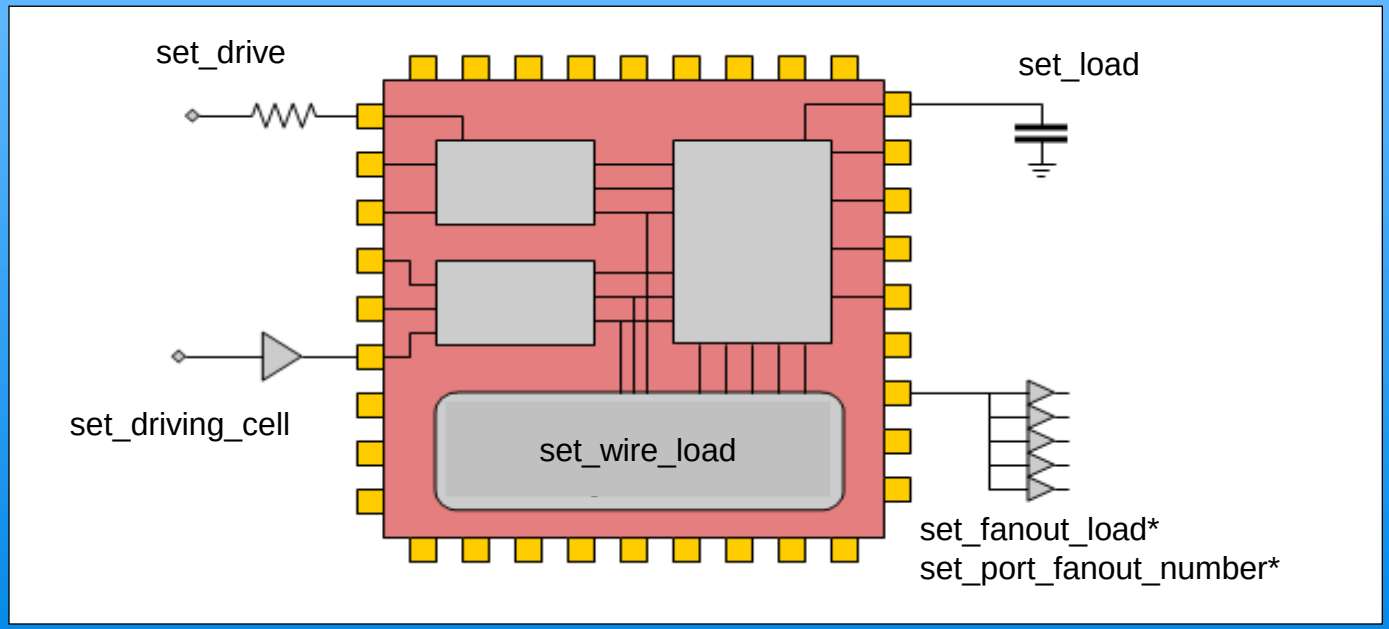


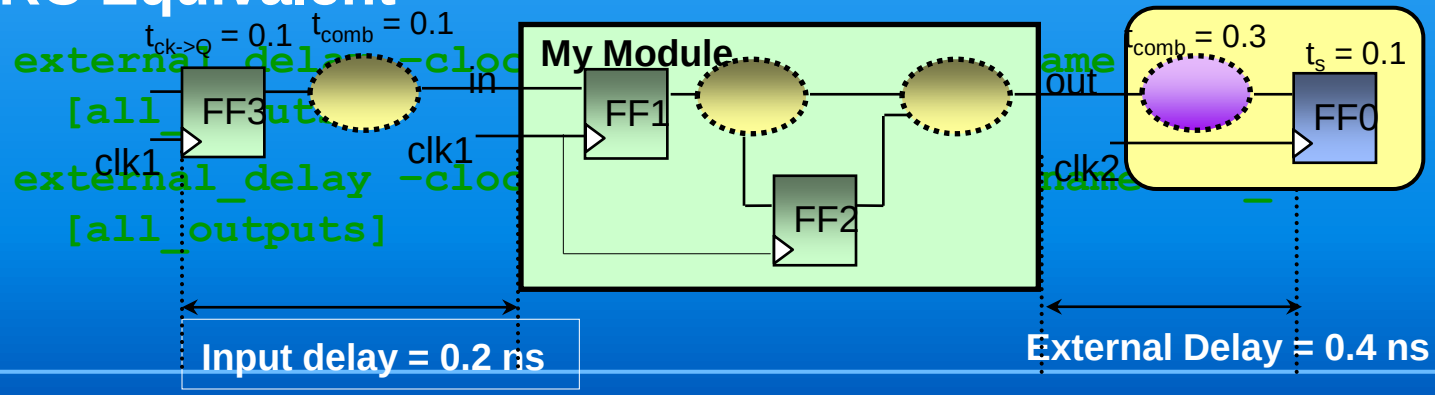
图6-30 输入输出环境参数

Input and Output Delays

- Use the following SDC commands to constrain input and output ports. External delays are not applicable to clock ports even if applied.

```
set_input_delay -clock clk1 0.2 [all_inputs]
set_output_delay -clock clk2 0.4 [all_outputs]
```

- **RC Equivalent**



Types of Timing Paths – I2O *(Input to Output)*

- Input to output timing paths should always be set with respect to a clock
- Yet, since they contain combinational logic only – they aren't associated with any clocks
  - Setup requirement:

clk_launch_edge + max_input_delay + max_Combo_delay =< clk_capture_edge – max_out_delay

  - Hold requirement:

clk_launch_edge + min_input_delay + min_combo_delay >= previous_capture_edge + min_out_delay

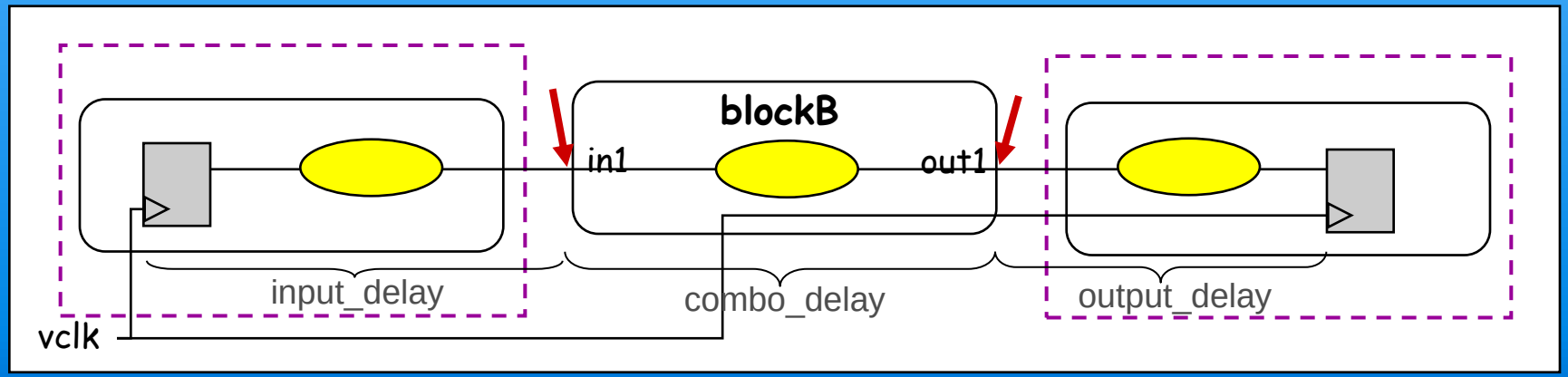


图6-29 时序路径的setup和hold分类分析I2O

Static Timing Analysis – Exceptions

Multi Clock Cycle

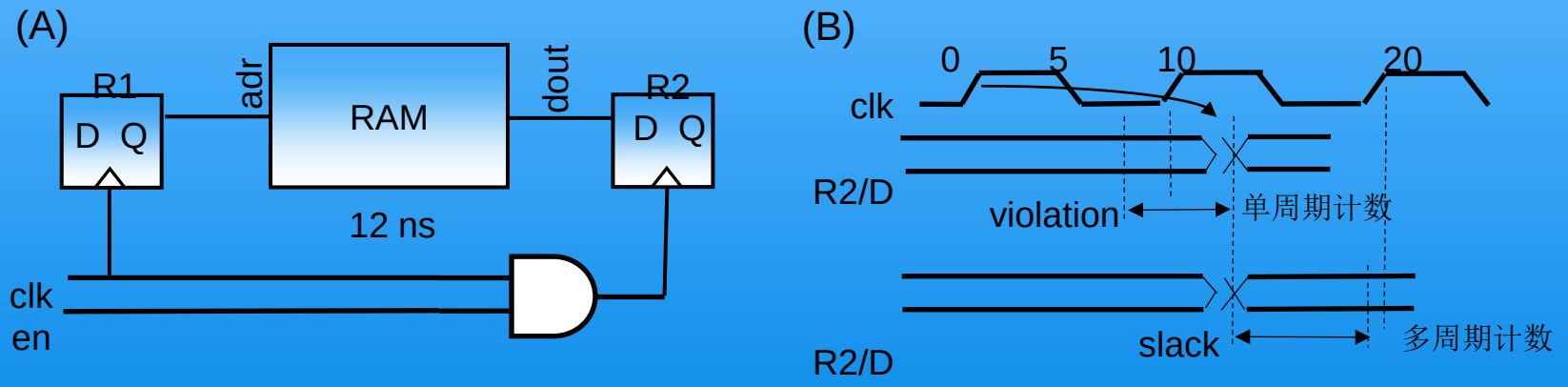
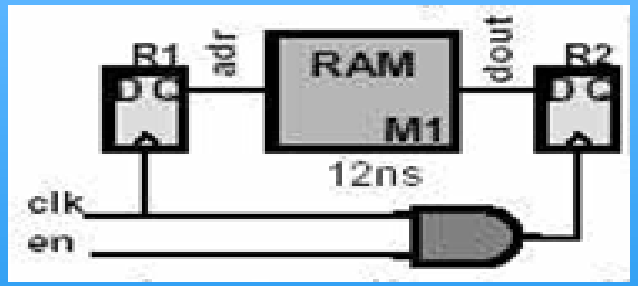


图6-32 多个时钟周期内时序路径的setup和hold的特例分析

Static Timing Analysis – Exceptions

Examples of Multi Clock Cycle



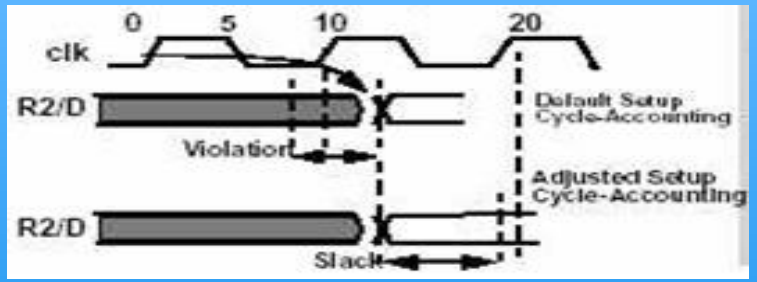


图6-32 多个时钟周期内时序路径的setup和hold的特例分析

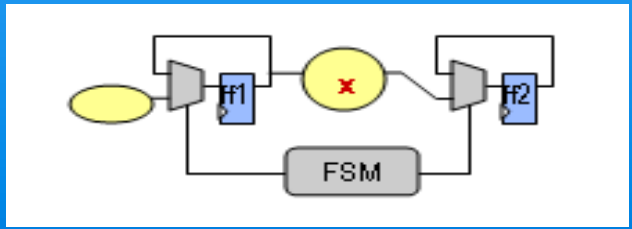


图6-33 多个时钟周期内时序路径的setup和hold的特例分析

Static Timing Analysis – Exceptions

False Path and Max Delay

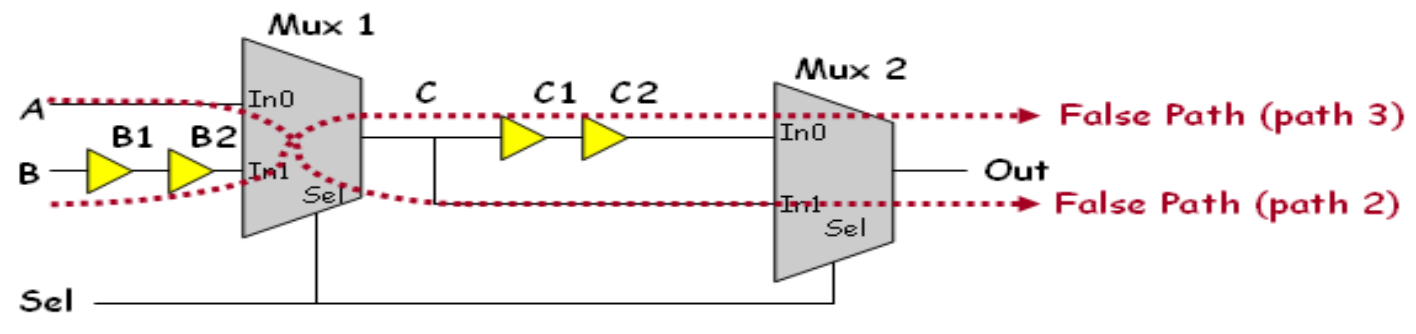


图6-34 功能虚假路径特例

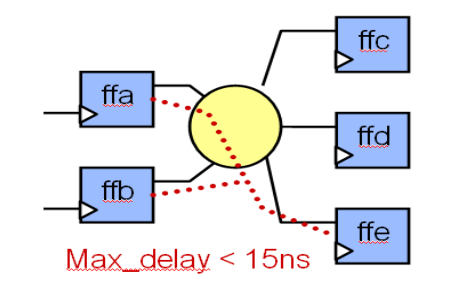


图6-35 最大延时路径特例一

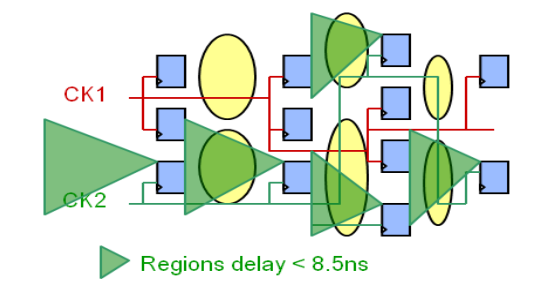


图6-36 最大延时路径特例二

SDC and STA

- Performance and Timing
- Logic Synthesis and WLM
- SDC and STA
- SDC and Clock
- Discussion

Static Timing Analysis - *Single Clock Cycle*

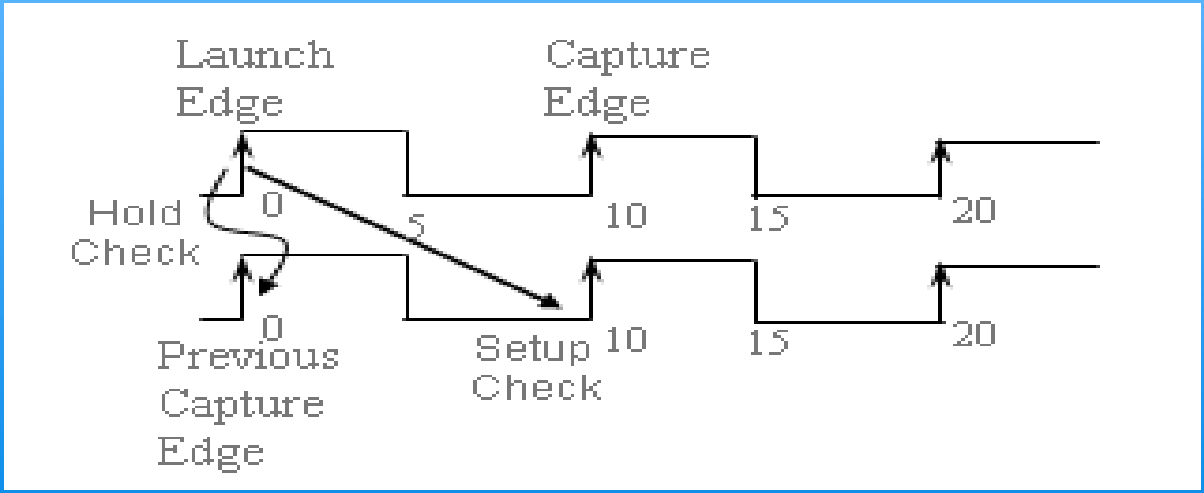


图6-31 单个时钟周期内时序路径的setup和hold的惯例分析

Defining Clock Domains

- When the clocks are defined in different clock domains, then
  - The clocks will not be considered as synchronous.
    (The timing report will show *async* for the clocks.)
  - The compiler will place functional false paths between the clock domains automatically.

```
define_clock -period 10000 -name 100MHz -domain clocka
  [find / -port clka]

define_clock -period 20000 -name 50MHz -domain clockb
  [find / -port clkb]
```

- There is no SDC equivalent command for setting clock domains. You have to specify functional false paths between asynchronous clocks in your design.

Design Constraints

Frequently used commands (clocks)

- create_clock
  - [port_pin_list] [-name clock_name] [-period period_value] [-waveform edge_list]
- create_generated_clock
  - [-name] -source [(-divide_by | -multiply_by [-duty_cycle]) [-invert]] [-edges {edge1, edge2, edge3} [-edge_shift {shift1, shift2, shift3}] pin_list

Defining a Clock

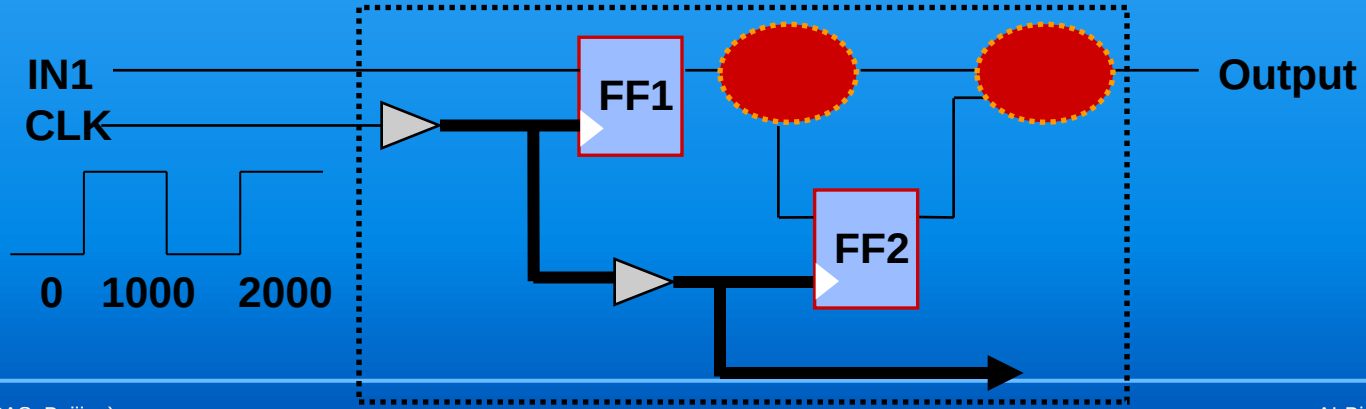
- Use the *create_clock* SDC command to define clock objects and their associated details such as a clock waveform.

```
create_clock -period 1 -name 1GHz [get_ports CLK]
```

- **Encounter RTL Compiler (RC) Equivalent**

```
define_clock –period 1000 –name 1GHz [find / -port CLK]
```

- *Note the difference in units: RC period 1000 ps = DC period 1 ns*



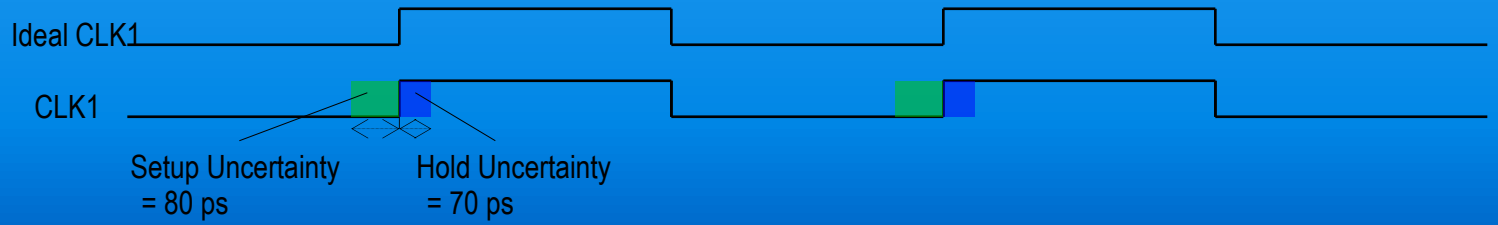
Defining Clock Uncertainty

- Set the clock uncertainty using the following SDC command:

```
set_clock_uncertainty -setup 0.08 –hold 0.07 [get_clocks
  CLK1]
```

- **RC Equivalent**

```
set_attr clock_setup_uncertainty 80 [find / -clock CLK1]
set_attr clock_hold_uncertainty 70 [find / -clock CLK1]
```



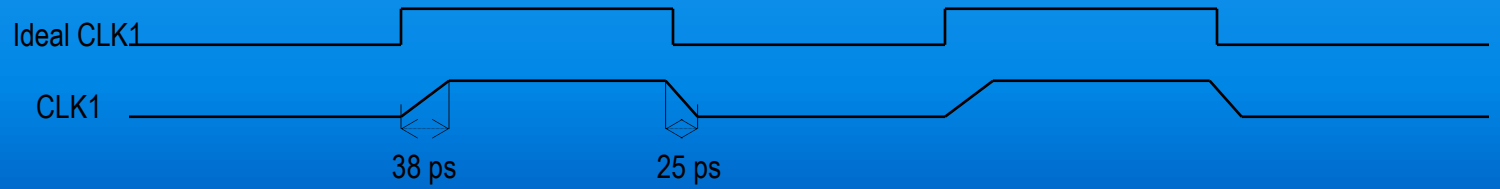
Setting Clock Slew

- To set the slew (transition) times in ideal mode, use the following SDC commands:

```
set_clock_transition 0.038 -rise [get_clocks CLK1]
set_clock_transition 0.025 -fall [get_clocks CLK1]
```

- **RC Equivalent**

```
set_attribute slew {0 0 38 25} [find / -clock
  CLK1]
```



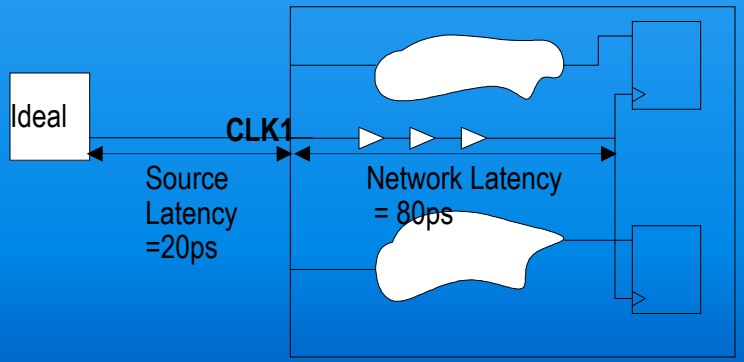
Modeling Clock Latency

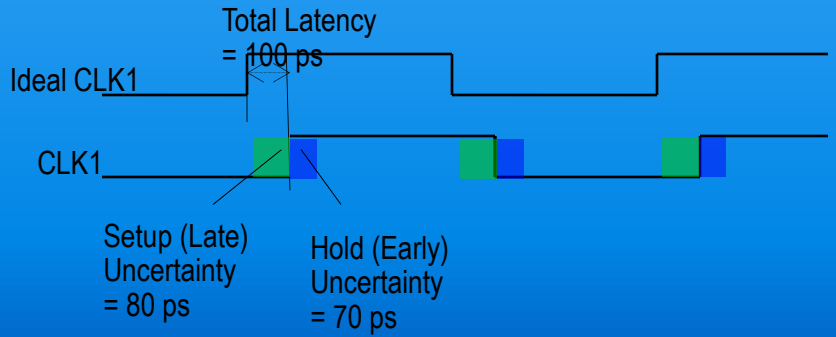
- Set the clock latency for the setup (late) violations using the following SDC command:

```
set_clock_latency 0.08 -late [get_clocks CLK1]
set_clock_latency 0.02 -source -late [get_clocks CLK1]
```

- **RC Equivalent**

```
set_attr clock_network_late_latency {80} [find / -clock CLK1]
set_attr clock_source_late_latency {20} [find / -clock CLK1]
```





Modeling Virtual Clocks

- A virtual clock is a clock object that is not associated with any source in the design.

```
create_clock –period 2 –name vclock1
```

- **RC Equivalent**

```
define_clock –period 2000 –name vclock1
```

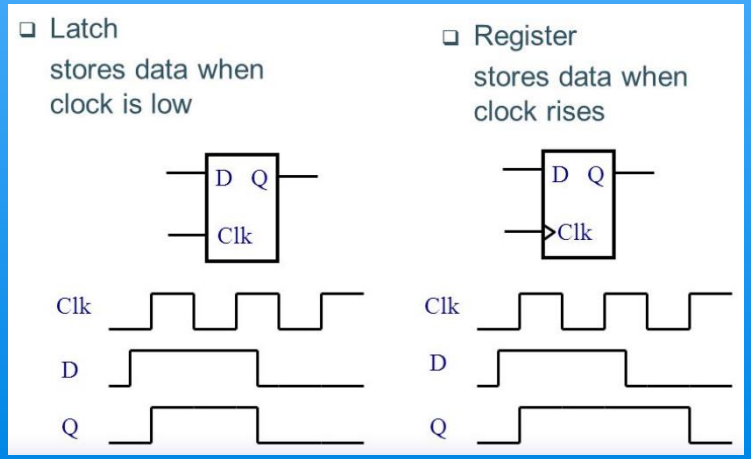
  - You can create as many clock objects as required. Use the *create_clock* command, and specify a clock name for each virtual clock.
  - Use these clocks in the design to specify the timing for a combinational logic block.

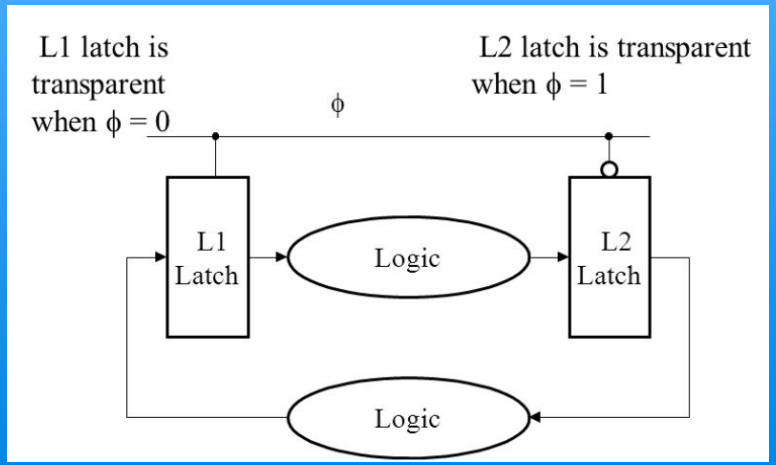
SDC and STA

- Performance and Timing
- Logic Synthesis and WLM
- SDC and STA
- SDC and Clock
- Discussion

Advanced Clock/Timing Topics

- Latches and D-Flip-Flops
- Latch-Based Clock
  - Timing Borrow





DDRx Clock

- Memory Clocks
  - DDRx Clock vs I/O Bus Clock
  - LPDDR Clocks
  - GDDR Clocks

Summary

- SDC/STA is/executed in FED and BED
  - Analysis procedure is similar but w/ different assumptions
- All Clocks in BED are set to “propagated”
- STA in BED is associated with real extracted RC data
- SI does not happen in FEB
  - SI is an engineering thing in BED