Timing Analysis (Cont.),
Logic Synthesis: A Practical View

Dejan Markovic
dejan@ee.ucla.edu

Timing Analysis

- Basic timing constraints
  - Long paths (cycle time / setup violations)
  - Short paths (hold time / hold violations)

- Clock nonidealities (skew and jitter)
  - Impact of Clk skew on timing
  - Impact of Clk jitter on timing

- Flip-flop vs. latch timing
  - Time borrowing (latches only)
Timing Analysis (1): Cycle Time (Long Path)

- Determines minimum cycle time (maximum speed)
  - May affect performance, do not affect functionality
- Cycle time constraint (setup time)
  \[ T_{cycle} = T_{Clk-Q} + T_{Logic,max} + T_{setup} + T_{skew} \]

Timing Analysis (2): Hold Time (Short Path)

- Min-delay paths could cause race conditions
  - Irreparable post-silicon (unless you FIB the chip)
- Min-delay constraint (hold time)
  \[ T_{hold} < T_{Logic,min} + T_{Clk-Q} \]
Timing (Cycle Time & Race Margin)

Cycle time: $T_{\text{Clk}} > t_{\text{Clk-Q}} + t_{\text{logic,max}} + t_{\text{su}}$

Race margin: $t_{\text{hold}} < t_{\text{Clk-Q}} + t_{\text{logic,min}}$

Setup and Hold Times

- Defined with respect to closing clock edge
  - Both in latches and flip-flops (built from latches)

- Could be positive or negative
  - Implementation dependent
    - Setup time
      - Master-Slave CSEs have positive setup
      - Pulse-triggered CSEs have negative setup
    - Hold time
      - Master-Slave CSEs may have close-to-zero or negative hold time
      - Pulse-triggered CSEs have large positive hold time
Positive Setup Time

Negative Setup Time
• Transparent to D only when Clk and \( \overline{\text{Clk}}_1 \) are both high
• Limited clock uncertainty absorption
• Data can arrive after the clock (negative setup time)
  • But large positive hold time / race hazard

**Clock Nonidealities**

- **Clock skew**
  - **Spatial** variation in temporally equivalent clock edges; deterministic + random, \( t_{SK} \)
- **Clock jitter**
  - **Temporal** variations in consecutive edges of the clock signal; modulation + random noise
    - Cycle-to-cycle (short-term) \( t_{JS} \)
    - Long term \( t_{tL} \)
- **Variation of the pulse width**
  - for level-sensitive clocking
Both skew and jitter affect the effective cycle time

Only skew affects the race margin
Sources of Skew and Jitter

1. Clock Generation
2. Devices
3. Interconnect
4. Power Supply
5. Temperature
6. Capacitive Load
7. Coupling to Adjacent Lines

Positive Skew

Launching edge arrives before the receiving edge
Negative Skew

Receiving edge arrives before the launching edge

Positive and Negative Skew

(a) Positive skew

(b) Negative skew
**Impact of Clock Skew on Timing**

### Cycle time (Long Path)

- Late - Early analysis

\[
t_{c-q} + t_{logic} + t_{su} < T_{Clk} + \delta
\]

\[
T_{Clk} > t_{c-q} + t_{logic} + t_{su} - \delta
\]

### Race immunity (Short Path)

- Data must not arrive before this time

\[
t_{c-q,cd} + t_{logic,cd} > t_{hold} + \delta
\]

\[
t_{hold} + \delta < t_{c-q,cd} + t_{logic,cd}
\]
Impact of Clock Jitter

Impact of Clock Jitter on Timing

Cycle time (Late-Early Problem)

Latest point of launching

Earliest arrival of next cycle

\[ t_{c-q} + t_{\text{logic}} + t_{\text{su}} < T_{\text{Clk}} - t_{\text{jitter}} - t_{\text{jitter}} \]

\[ T_{\text{Clk}} > t_{c-q} + t_{\text{logic}} + t_{\text{su}} + 2t_{\text{jitter}} \]
Impact of Skew and Jitter on Timing

- **Cycle time**
  - Positive skew improves performance
  - Negative skew reduces performance
  - Jitter reduces performance

\[ T_{\text{Clk}} > t_{c-q} + t_{\text{logic}} + t_{su} - \delta + 2 t_{\text{jitter}} \]

- **Race Margin**
  - Skew reduces race margin
  - Jitter reduces acceptable skew

\[ t_{\text{hold}} + 2 t_{\text{jitter}} + \delta < t_{c-q,cd} + t_{\text{logic,cd}} \]

\[ t_{\text{Clk}} \text{(clock uncertainty)} \]

Time Borrowing

**Classification:**

- **Dynamic time borrowing**
  - Scheduling data to arrive to CSE when CSE is transparent
    - No "hard" boundaries between stages
  - Occurs in latch-based level sensitive and soft-edge clocking.

- **Static time borrowing**
  - Inserting delay between clock inputs of the clocked storage elements.
  - Clocks are scheduled to arrive so that the slower paths obtain more time to evaluate, taking away the time from faster paths.
  - It can operate with conventional hard-edge Flip-Flops.
  - Also called *opportunistic skew scheduling*
In fast paths, analysis must assume that the data arrives at earliest possible time -> disregard effects of time borrowing

Oklobdzija et al.
Digital System Clocking, Wiley'03

Dynamic Time Borrowing

- Works in latch-based designs
  - Clock pulse-width can be borrowed if the next stage can pay it back (with shorter-delay logic)

- Cycle-time constraint
  - $T_{Clk} + T_W > t_{logic,max} + t_{setup} + t_{Clk-Q}$

- Min-path constraint
  - $T_W + t_{hold} < t_{logic,min} + t_{Clk-Q} + t_{borrowed}$

Only if you look at path just after borrowing
Two-Phase Clocking

- Freedom to control two phases and pulse-width

\[ \Phi_1 \]
\[ \Phi_2 \]

- Cycle-time constraint
  \[ T_W + T_{\text{Clk}} > t_{\text{logic1, max}} + t_{\text{logic2, max}} + 2t_{\text{D-Q}} \]

- Bounds on pulse-width \( T_W \)
  - Max delay: \( T_W + T_{\text{Clk}}/2 > t_{\text{logic, max}} + t_{\text{D-Q}} \)
  - Min delay: \( T_{\text{Clk}}/2 + t_{\text{logic, min}} + t_{\text{Clk-Q}} > T_W + t_{\text{hold}} \)

- Max allowable skew
  \[ t_{\text{skew, max}} = (T_{W, \text{max}} - T_{W, \text{min}}) / 2 \]

Logic Synthesis: A Practical View

- Constraining your design for logic synthesis

- Design constraints
  - Environmental constraints
    - Driver
    - Load (max fanout)
  - Define clocks
    - Cycle time
    - Uncertainty
  - Optimization constraints
    - In and out delay / max transition
    - Max area etc.
  - (Timing exceptions)
Example: Automated Adder Synthesis

- Copy the following files to your work directory
  
  /usr/public.2/ee216a/cadence/SOC62/SynLib.v
  /usr/public.2/ee216a/cadence/SOC62/adder.v
  /usr/public.2/ee216a/cadence/SOC62/adder.tcl

- The top level synthesis script adder.tcl reads in the hdl file, sets timing, load and power constraints, and runs synthesis.

- To run RTL synthesis type the following command
  
  > rc –files adder.tcl –gui

- The GUI window will show detailed architecture (gate level). Use report power, report area, report timing commands in the rc command window to get Power, Area and Delay numbers.

Setup Example: Adder.tcl (1/2)

```tcl
set_attribute library
    /w/apps/apps.16/cadence/gsclib090_v2.9/timing/typical.lib
set_attribute hdl_language v2001
read_hdl adder.v
read_hdl SynLib.v
elaborate adder

dc::current_design adder

dc::set_time_unit -picoseconds

dc::set_load_unit -femtofarads

define_clock -name clk -period 1000 -design /designs/adder
    {/designs/adder/ports_in/clk}
dc::set_input_delay 20 -clock clk [all_inputs]
dc::set_output_delay 100 -clock clk [all_outputs]
set_attribute external_driver [find [find / -libcell DFFX1] -libpin D]
    {/designs/adder/ports_in/*}
set_attribute external_pin_cap 26.5488 {/designs/adder/ports_out/*}
set_attribute lp_power_unit mW
set_attribute max_dynamic_power 0.5 /designs/adder
```

Define lib
Read HDL
Elaborate
Set timing
Set env.
Opt.
Setup Example: Adder.tcl (2/2)

```
synthesize -to_mapped -effort high
report area > adder_area.rpt
report power > adder_power.rpt
report timing > adder_timing.rpt
report clocks > adder_clocks.rpt
write_encounter design adder -basename adder -lef [format "%s %s %s"
    /w/ee.00/dejan/rashmi/ee216a_test/gpdk090_9lm.lef
    /w/apps/apps.16/cadence/gsclib090_v2.9/lef/gsclib090_tech.lef
    /w/apps/apps.16/cadence/gsclib090_v2.9/lef/gsclib090_macro.lef]
```

**Synthesis**

**Reports**

---

How does synthesis work?

---

Logic Synthesis is Timing-Driven

- This is a generic design used during synthesis
  - Internal data-path delay
  - Relationship to in and out paths
  - Timing exceptions

![Diagram of a generic design used during synthesis]
Understanding Timing Constraints

- These three constraints is all that matters
  - blue box = current_design (to be retimed)

```
create_clock -name "Clk" -period $Tclk
set_input_delay 0.5
set_output_delay [expr $Tclk - 0.5 + $wire_margin] ;# wire_margin = 0.2
```

Example Clock, In and Out Delay [Synopsys]

- Define clock, input and output delay
Adder.tcl

The adder.tcl file sets constraints & optimization parameters

```tcl
set_attribute library /u/apps/app/16/cadence/gsc1b09j-v2.9/timing/typical.lib
set_attribute hdl_language v2001
read_hdl adder.v
read_hdl StdLib.v
elaborate adder

current_design adder
set_time_unit -picoseconds
set_load_unit -fentofarads

define_clock -name clk -period 10 -design /designs/adder (/designs/adder/ports_in/clk)
def_set_input_delay 20 -clock clk [all_inputs]
def_set_output_delay 100 -clock clk [all_outputs]
def_attribute external_driver [Find / -libcell DFFM1] -libpin D] (/designs/adder/ports_in/)
def_attribute external_pin_cap 20.5 -design /designs/adder/ports_out/
def_attribute low_power_unit AW/
def_attribute max_dynamic_power 25 /designs/adder

# could try 0 for min power
```

Timing Constraints

- **Clock Period set by** `define_clock`
- **Input and output delay**
  - **Input delay**: arrival time of an external path with respect to a Clk edge
  - **Output delay**: an external timing path from an output port to a register input of an external block
  - These delays shrink the clock period if the input or the output port is not registered

![Input and Output Delays Diagram]
**Drivers, Load…**

- To simulate realistic inputs, we can set the driving cell using the `external_driver` command, to be any cell in the library
  - This ensures that the input has a finite slew rate

- The load capacitance can be set on the output ports using the `external_pin_cap` command

- By default the Encounter RTL Compiler only tries to meet the timing constraints without optimizing power

- If the `max_dynamic_power` attribute is set to some value, the tool tries to meet the timing specs while also optimizing for power in the process

---

**Synthesis Results: Fast Adder**

- Adder synthesized to meet timing constraint of 550 ps
- Input is registered from a flip flop
  - Clk-Q delay = 157 ps
  - Setup time = 82 ps
- Effective adder delay = 311 ps

- Timing constraints are stringent, tool synthesizes a carry look ahead type of adder

- Synthesis reports
  - Area = 1035 $\mu$m$^2$
  - Energy (switching) = 0.2145 fJ
  - Power (leakage) = 0.005 mW
Synthesis Results: Slow Adder

- Adder delay = 757 ps
- Structure is somewhat like carry ripple topology
- Area = 548 $\mu$m$^2$ (half of previous)
- Energy (switching) = 0.13 fJ
  - Nearly half of previous design
  - This is expected since $V_{DD}$ was the same, so energy would depend only on $C_{sw}$, which was halved with the area being halved...
- Power (leakage) = 0.002 mW
  - Also reduced due to reduced area

Synthesis Results

- Energy-delay tradeoff plot obtained from synthesis
  - Keep timing constraint, move down the energy axis (left plot)
  - Resulting energy-area should be below reference curve (right plot)
Some Project Tips

- Do a quick analysis and see what blocks are worth optimizing
  - Keep in mind that system-level methods far outstrip lower-level tuning
- Custom design strategies
  - A) Full-custom layout
  - B) Hand-design, but restrict to use of (mostly) standard cells
  - In both cases, layout-VXL could be very useful
- Putting the blocks together
  - A) Hand-route
  - B) Structural Verilog
- Place and route flow (Ph-2)
  - Sample scripts coming soon

Cadence Manuals / Tips Etc.

- Online resources:
  - http://sourcelink.cadence.com
- Sign up for an account
  - Disclaimer (from Cadence)
    - Reference key for UCLA is: 606-ucla1108
    - Only university staff, faculty and students may use this reference key
    - You must sign up using a university domain email address