Lecture 5: Timing

David Black-Schaffer
davidbbs@stanford.edu
EE183 Spring 2003

Overview

- Timing
  - Our designs are limited by getting data between FFs
  - Combinational Logic delay, Routing delay, and FF parameters dictate the maximum speed
- Clock Distribution
  - H-Tree format for getting clocks at the 'same' time
  - Dedicated clock routing tries to reduce skew and jitter
- FF Timing
  - Make sure the slowest path is fast enough for the FF based on the clock (MaxPath)
  - Make sure the fastest path is not too fast for the FF based on the clock (MinPath)

Logistics

- Any questions?
  - Lab 1 requirements clear?
  - Demo due Friday at 5pm.
    We'll be in the lab from 3-5 on Friday.

FSM Review

A Finite State Machine is simply a **state register** that holds the current state and some **combinational logic** which calculates the next state and outputs based on the current state and the inputs.

- A feedback system which updates on each clock

What is it really?

- A bunch of MUXes which select the next state & outputs based on the current state (FFs) & inputs.
- Keep this in mind when writing your verilog.


```
always @(current_state_q or button)
   begin
        case(current_state_q)
        `STOP: begin
                go = 1'b0;
                if (button) begin
                        next_state_d = `GO;
                        end
                else    begin
                        next_state_d = `STOP;
                        end
                end
        `GO: begin
                go = 1'b1;
                if (button) begin
                        next_state_d = `STOP;
                        end
                else    begin
                        next_state_d = `GO;
                        end
                end
        default: begin
                go = 1'b0;
                next_state_d = `STOP;
                end
```

The next state must **ALWAYS** be defined.

Any output must be defined in **EVERY** state.

If any states are ever not used they **MUST** be included in a **`default`** statement.

an `else` for every `if`
a `default` for every `case`
every output must be defined in every state

Public Service Announcement

- Xilinx Programmable World
  - Tuesday, May 6th
  - http://www.xilinx.com/events/pw2003/index.htm
- Guest Lectures
  - Monday, April 28th
    Ryan Donohue on Metastability and Synchronization
  - Wednesday, May 7th
    Gary Spivey on ASIC & FPGA Design for Speed
  - The content of these lectures will be on the Quiz

Timing

- What limits our speed?
- RTL design - Register Transfer Logic
  - Speed limited by the time it takes to get from one register (flip flop) to another
  - State machines and pipeline data stages
- What gets in the way?
  - Combinational logic delay
  - Routing delay
  - Clock skew and delay
  - FF setup and hold time requirements

Combination Logic Delay

- Time to get through gates
  - The more gates (more complicated logic) the slower
  - One-hot encoding
- Well, sort of:
  - FPGA doesn't have "gates" we have those 4-input LUTs (look-up-tables)
  - All functions of 4 inputs or less are the same speed
  - >4 inputs and we have to hook up multiple CLBs (routing delay)

Routing Delay

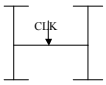
- This can be (**IS!**) the real killer
- FPGA wiring slow compared to ASIC
  - Lots of switches to go through
  - Wires don't go exactly where you want
- Routing can be ~50% of your total delay
- Manual placement? NP-hard problem?
- Hope the tools work well!

Clock Distribution

- How do we get the same clock signal **everywhere** at the same time?
- H-Tree distribution
- Is is perfect? No, we have:
  - Skew - static variation in time of arrival at different FFs
    - Due to path length differences
  - Jitter - cycle-varying variation in arrival at the same FF
    - Oscillator/PLL inaccuracies, differential heating, cross-talk

Skew

- FPGA Structure: very regular, "easy" distribution
- ASIC: Random logic, different #s of FFs in different places



XILINX® Spartan-II 2.5V FPGA Family: DC and Switching Characteristics

Clock Distribution Guidelines(1)

Symbol	Description	Speed Grade -6 Max	Speed Grade -5 Max	Units
GCLK Clock Skew				
$T_{GSKEWIOB}$	Global clock skew between IOB flip-flops	0.13	0.14	ns

Notes:
1. These clock distribution delays are provided for guidance only. They reflect the delays encountered in a typical design under worst-case conditions. Precise values for a particular design are provided by the timing analyzer.

Jitter

DLL Clock Tolerance, Jitter, and Phase Information

All DLL output jitter and phase specifications were determined through statistical measurement at the package pins using a clock mirror configuration and matched drivers.

Figure 1, page 13, provides definitions for various parameters in the table below.

Symbol	Description	F_{CLKIN}	CLKDLLHF Min	CLKDLLHF Max	CLKDLL Min	CLKDLL Max	Units
T_{IPTOL}	Input clock period tolerance		-	1.0	-	1.0	ns
T_{IJITCC}	Input clock jitter tolerance (cycle-to-cycle)		-	±150	-	±300	ps
T_{LOCK}	Time required for DLL to acquire lock	> 60 MHz	-	20	-	20	µs
		50-60 MHz	-	-	-	25	µs
		40-50 MHz	-	-	-	50	µs
		30-40 MHz	-	-	-	90	µs
		25-30 MHz	-	-	-	120	µs
T_{OJITCC}	Output jitter (cycle-to-cycle) for any DLL clock output(1)		-	±60	-	±60	ps
T_{PHIO}	Phase offset between CLKIN and CLKO(2)		-	±100	-	±100	ps
T_{PHOO}	Phase offset between clock outputs on the DLL(3)		-	±140	-	±140	ps
T_{PHIOM}	Maximum phase difference between CLKIN and CLKO(4)		-	±160	-	±160	ps
T_{PHOOM}	Maximum phase difference between clock outputs on the DLL(5)		-	±200	-	±200	ps

Notes:

1. **Output Jitter** is cycle-to-cycle jitter measured on the DLL output clock, *excluding* input clock jitter.
2. **Phase Offset between CLKIN and CLKO** is the worst-case fixed time difference between rising edges of CLKIN and CLKO, *excluding* output jitter and input clock jitter.
3. **Phase Offset between Clock Outputs on the DLL** is the worst-case fixed time difference between rising edges of any two DLL outputs, *excluding* Output Jitter and input clock jitter.
4. **Maximum Phase Difference between CLKIN an CLKO** is the sum of Output Jitter and Phase Offset between CLKIN and CLKO, or the greatest difference between CLKIN and CLKO rising edges due to DLL alone (*excluding* input clock jitter).
5. **Maximum Phase Difference between Clock Outputs on the DLL** is the sum of Output Jitter and Phase Offset between any DLL clock outputs, or the greatest difference between any two DLL output rising edges due to DLL alone (*excluding* input clock jitter).

What is *Faster?*

- Is a 1.1GHz P4 faster than a 933MHz P3?
- Wall clock time is all that matters.*
  - How long does it take to get your answer
  - Faster clock speed doesn't mean you do more
  - Pipelining: less of each instruction per clock, but more instructions and more clocks = greater throughput

*Unless you have a really good marketing department.

Also, power matters **a lot!

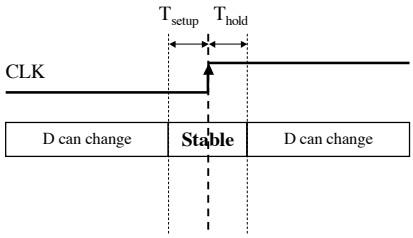
FF Timing Constraints

- What happens if D and CLK change at the same time?
  - How close can these get?
  - Determined by the Setup and Hold times
- What happens if we try to use Q right after the clock?
  - How long do we have to wait?
  - Determined by $T_{clk->Q}$
- Do we care?
  - Mostly concerned about MaxPath, but MinPath violations can be serious in ASICs
  - **Also, this will be on the midterm Quiz**

Setup and Hold Times

- Setup Time: the amount of time the synchronous input (D) must be stable **before** the active edge of the clock
- Hold Time: the amount of time the synchronous input (D) must be stable **after** the active edge of the clock
- If either is violated correct operation of the FF is not guaranteed
- Metastability can result.

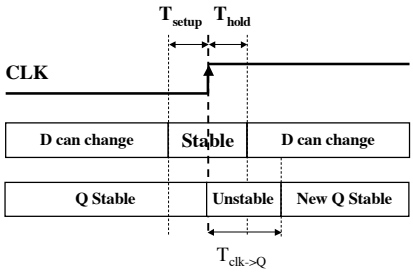
Setup and Hold Diagram



$T_{clk->Q}$

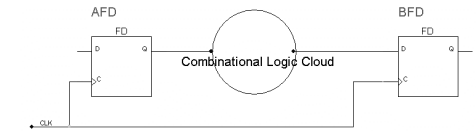
- $T_{clk->Q}$: the amount of time you have to wait after the CLK before the output (Q) is valid
- If you try to use the output before this you will get inconsistent results depending on if Q changes

$T_{clk->Q}$ Diagram



Example Parameters

- DFF values:
  - $T_{clk->q}$=1ns, T_{setup}=1ns, T_{hold}=1ns
- Clock skew is max 2ns and jitter is 2ns
- Combinational logic
  - max time: T_{cl_pdmax}=10ns
  - min time: T_{cl_pdmin}=1ns



MaxPath Timing Constraint

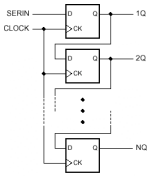
- Add up the components that result in the time budget; the period must be greater than this value.
- $T_{clk\text{->}q}+T_{cl\_pdmax}+T_{setup}+T_{skew}$<= Clock Period
- 1 + 10 + 1 + 2 <= Clock Period
- 14ns <= Clock Period
- Max Frequency is 71MHz
- Longest one is the ***Critical Path***

MinPath Timing Constraint

- Consider what happens when the same clock edge is considered at the far DFF.
- $T_{clk\text{->}q}+T_{cl\_pdmin}$ >= T_{skew} + T_{hold}
- 1 + 1 >= 2 + 1
- Whoops!! The new value from FF A can get there so fast that when the clock arrives the new value may change before it has been latched in to FF B.
- AKA, "Hold-Time Violation"

MinPath and ShiftRegisters

- Shift Registers can easily fall prey to min path timing violations.
- Fix the violations by increasing delay between Ds and Qs
  - Insert pairs of inverters
- FPGA DFF clk->q is big enough so that MinPath violations are rare.



Impacts

- You can "fix" MaxPath timing constraint violations by slowing down the clock after the circuit is implemented.
  - Give the logic more time to get the result to the FF
- You ***cannot*** "fix" MinPath timing constraint violations be modifying the clock.
  - How do you prevent the logic from changing too quickly?

Lecture 5 Key Points

- Speed is a function of how fast you can get from FF to FF.
- Routing and combinational logic delay is what will mostly bite you in this class.
- FFs themselves do have input and output timing requirements of which you need to be aware

- Logistics
  - **Lab 1 DEMO due Friday at 5pm.**
  - Office Hours: Mon/Wed/Fri 10-11am, Tue/Thur 7-9pm, and demos Fri 3-5pm

EE183 Lecture 5 - Slide 25