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
        next_state_d = `STOP;
    end
    `GO: begin
      go = 1'b1;
      if (button) begin
        next_state_d = `STOP;
      end
      else
        next_state_d = `GO;
    end
    default: begin
      go = 1'b0;
      next_state_d = `STOP;
    end
  endcase
```

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
- 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 (ISI) 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
  - Hope the tools work well!

Clock Distribution

- How do we get the same clock signal everywhere at the same time?
- H-Tree distribution
- Is it perfect? No, we have:
  - Skew - static variation in time of arrival at different FFs
  - 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

Clock Distribution Guidelines

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Speed Grade</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCK</td>
<td>Global clock skew between I/Os/slip-rates</td>
<td>0.15</td>
<td>0.14</td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

Notes:
- 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.

Spartan-6 3.3V FPGA Family: DC and Switching Characteristics
Jitter

### DLL Clock Tolerance, Jitter, and Phase Information

A DLL is used to reference phase relationships with respect to the reference clock. The table below shows tolerances, frequencies, and duty cycles. Figure 1 shows the effect of various parameters on the package pin using a clock tree measurement and interconnect.<br>

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Unit</th>
<th>MAX</th>
<th>Nom</th>
<th>Min</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T_{DL} )</td>
<td>Jitter clock to data tolerance</td>
<td>ns</td>
<td>10</td>
<td>5</td>
<td>200</td>
<td>-</td>
</tr>
<tr>
<td>( T_{DL} )</td>
<td>Jitter clock to data frequency tolerance</td>
<td>MHz</td>
<td>280</td>
<td>125</td>
<td>150</td>
<td>-</td>
</tr>
<tr>
<td>( T_{DL} )</td>
<td>Time required for DLL to acquire lock</td>
<td>ms</td>
<td>80</td>
<td>40</td>
<td>150</td>
<td>-</td>
</tr>
<tr>
<td>( T_{DL} )</td>
<td>DLL lock to data phase variance</td>
<td>us</td>
<td>50</td>
<td>25</td>
<td>100</td>
<td>-</td>
</tr>
<tr>
<td>( T_{DL} )</td>
<td>DLL lock to data phase variance</td>
<td>us</td>
<td>50</td>
<td>25</td>
<td>100</td>
<td>-</td>
</tr>
<tr>
<td>( T_{DL} )</td>
<td>DLL lock to data phase variance</td>
<td>us</td>
<td>50</td>
<td>25</td>
<td>100</td>
<td>-</td>
</tr>
<tr>
<td>( T_{DL} )</td>
<td>DLL lock to data phase variance</td>
<td>us</td>
<td>50</td>
<td>25</td>
<td>100</td>
<td>-</td>
</tr>
<tr>
<td>( T_{DL} )</td>
<td>DLL lock to data phase variance</td>
<td>us</td>
<td>50</td>
<td>25</td>
<td>100</td>
<td>-</td>
</tr>
</tbody>
</table>

### 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

- **$T_{\text{clk} \rightarrow \text{Q}}$**
  - $T_{\text{clk} \rightarrow \text{Q}}$: 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

---

**$T_{\text{clk} \rightarrow \text{Q}}$ Diagram**

<table>
<thead>
<tr>
<th>D can change</th>
<th>Stable</th>
<th>D can change</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Q Stable</th>
<th>Unstable</th>
<th>New Q Stable</th>
</tr>
</thead>
</table>

**Example Parameters**

- DFF values:
  - $T_{\text{clk} \rightarrow \text{Q}}=1\text{ns}$, $T_{\text{setup}}=1\text{ns}$, $T_{\text{hold}}=1\text{ns}$
- Clock skew is max 2\text{ns} and jitter is 2\text{ns}
- Combinational logic
  - max time: $T_{\text{cl_pllmax}}=10\text{ns}$
  - min time: $T_{\text{cl_pllmin}}=1\text{ns}$
MaxPath Timing Constraint

- Add up the components that result in the time budget; the period must be greater than this value.
- \( T_{\text{clk} \rightarrow q} + T_{\text{cl}_{-}pd_{\text{max}}} + T_{\text{setup}} + T_{\text{skew}} \leq \text{Clock Period} \)
- \( 1 + 10 + 1 + 2 \leq \text{Clock Period} \)
- 14ns \( \leq \) 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.
- \( T_{\text{clk} \rightarrow q} + T_{\text{cl}_{-}pd_{\text{min}}} \geq T_{\text{skew}} + T_{\text{hold}} \)
- \( 1 + 1 \geq 2 + 1 \)
- 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 \( \text{clk} \rightarrow q \) 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 by 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.