Synthesis and Timing (Verilog)

Module 11
Overview

- Metastability
- Constraints
- Clock Skew
- Clock Domains
  - Core Generator
Metastability

• Flip-flops may go metastable if input signals do not meet setup and hold specifications relative to clock signal

• Rules:
  – Input only drives one FF, Add 2-FF synchronizer

```verilog
always @(posedge clk)
begin
    inp_d  <= inp;
    inp_dd <= inp_d;
end
```
Creating one clk pulse synchronizer

- Sometimes an input signal is asynchronous and much longer than clock period
- Add 3FF synchronizer and generate single pulse

```verilog
always @(posedge clk)
begin
    inp_d     <= inp;
    inp_dd    <= inp_d;
    inp_ddd   <= inp_dd;
    inp_pulse <= inp_dd & ~inp_ddd;
```
Timing Constraints

- Used to guide the synthesis tools
- Example 32-bit counter - no constraints (speed grades -4 and -5)

Advanced HDL Synthesis Report
Macro Statistics
# Counters : 1
32-bit up counter : 1

Timing Summary:

**Speed Grade: -4**
- Minimum period: **6.680ns** (Maximum Frequency: 149.703MHz)
- Minimum input arrival time before clock: No path found
- Maximum output required time after clock: 8.094ns
- Maximum combinational path delay: No path found

**Speed Grade: -5**
- Minimum period: **5.767ns** (Maximum Frequency: 173.400MHz)

Jim Duckworth, WPI
Adding timing constraint

• Add to UCF file:
  - NET "clk" PERIOD = 6ns HIGH 50%;

WARNING:Par:62 - Your design did not meet timing.

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Check</th>
<th>Worst Case</th>
<th>Best Case</th>
<th>Timing</th>
<th>Timing</th>
<th>Slack</th>
<th>Achievable</th>
</tr>
</thead>
<tbody>
<tr>
<td>* NET &quot;clk_BUFGP/IBUFG&quot;</td>
<td>SETUP</td>
<td>-0.456ns</td>
<td>6.456ns</td>
<td>9</td>
<td>1430</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>HOLD</td>
<td>2.432ns</td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1 constraint not met.
Try relaxing constraint

- \text{NET} \ "clk" \ \text{PERIOD} = 6.5\text{ns} \ \text{HIGH} \ \text{50\%};

\begin{verbatim}
Constraint | Check| Worst Case | Best Case | Timing | Timing | Slack| Achievable |
Errors | Score
\hline
\text{NET} \ "clk\_BUFGP/IBUF\_G" \ \text{PERIOD} = 6.5 \text{ ns HIG} | SETUP| 0.203ns| 6.297ns| 0| 0
| HOLD | 2.280ns| 0| 0
\hline
\end{verbatim}

All constraints were met.

- Also see Timing Constraint User Guide
- Examples:
  - \text{NET} \ ʻabcʻ \ \text{OFFSET} = \text{OUT} xx \text{ns} \ \text{AFTER} \ ʻclkʻ;
  - \text{NET} \ ʻdefʻ \ \text{OFFSET} = \text{IN} xx \text{ns} \ \text{BEFORE} \ ʻclkʻ;

Jim Duckworth, WPI
Clock Skew

- The difference between the time a clock signal arrives at the source flip-flop in a path and the time it arrives at the destination flip-flop.
  - Misalignment of clock edges
  - Degrades (reduces) time for flip-flop to flip-flop timing
- Can be caused by different things but wire interconnect delays are the main cause inside FPGAs
  - There are fast dedicated clock circuits and slower data paths
Old method of deriving slower clocks

```
// create one second clock from 50MHz oscillator
always @(posedge reset or posedge clk_50M)
  if (reset == 1'b1)
    begin
      clk_1Hz <= 0;
      counter_50M <= 0;
    end
  else
    begin
      counter_50M <= counter_50M + 1;
      if (counter_50M == 25_000_000)
        begin
          counter_50M <= 0;
          clk_1Hz <= ~clk_1Hz;
        end
    end

// count seconds from 0 to 9
always @(posedge reset or posedge clk_1Hz)
  if (reset)
    count <= 0;
  else if (count == 9)
    count <= 0;
  else
    count <= count + 1;
```
**WARNING**: Route:455 – CLK Net:clk_1Hz may have excessive skew because 0 CLK pins and 1 NON_CLK pins failed to route using a CLK template.
Synthesis uses two clock signals
Two clock signals – not so good!
Modify to only use one clock signal

```vhdl
// create one second clock from 50MHz oscillator
always @(posedge reset or posedge clk_50M)
    if (reset == 1'h1)
        counter_50M <= 0;
    else
        if (counter_50M == 25_000_000 - 1)
            counter_50M <= 0;
        else
            counter_50M <= counter_50M + 1;

// count seconds from 0 to 9
always @(posedge reset or posedge clk_50M)
    if (reset)
        count <= 0;
    else if (counter_50M == 0)
        if (count == 9)
            count <= 0;
        else
            count <= count + 1;
```
Clock signals use dedicated lines
Clock drives all flip-flops
Crossing Clock Domains - FIFO

- ADC
- FIFO
- SRAM

400 MHz

200 MHz
Coregen – create new module
FIFO Generator

![FIFO Generator Image]

Jim Duckworth, WPI
Select Options (1 of 6)
Specifying Write and Read widths
Summary – uses one Block RAM
Core is added to Project

Generating FLIST file...
Finished FLIST file generation.
Preparing output directory...
Finished preparing output directory.
Moving files to output directory...
Finished moving files to output directory.
IP successfully created.
Instantiation Template is Provided

```
// The following must be inserted into your Verilog file for this core to be instantiated. Change the instance name and port connections (in parentheses) to your own signal names.

//-------- Begin Cut here for INSTANTIATION Template --------
fifo YourInstanceName (
  .rst(rst),
  .wr_clk(wr_clk),
  .rd_clk(rd_clk),
  .din(din), // Bus [3 : 0]
  .wr_en(wr_en),
  .rd_en(rd_en),
  .dout(dout), // Bus [3 : 0]
  .full(full),
  .empty(empty));

// INST_TAG_END ------ End INSTANTIATION Template -------
```
Simple Top Level to Demonstrate Use

```vhdl
module fifo_example(
    input  reset,
    input  clk_100,
    input  [3:0] data_in,
    input  clk_50,
    output [7:0] data_out
);

fifo fifo_1 (
    .rst(reset),
    .wr_clk(clk_100),
    .rd_clk(clk_50),
    .din(data_in),   // Bus [3 : 0]
    .wr_en(1'b1),
    .rd_en(1'b1),
    .dout(data_out), // Bus [7 : 0]
    .full(full),
    .empty(empty));

endmodule
```
FIFO added – RTL Schematic
FIFO added – Technology Schematic
Simple Test Bench to Show Operation

```vhdl
41   clk_50(clk_50),
42   .data_out(data_out)
43   );
44
45   // generate 100 MHz data input clock
46   always
47       begin
48         #5 clk_100 = 0;
49         #5 clk_100 = 1;
50       end
51
52   // generate 50MHz read clock
53   always
54       begin
55         #10 clk_50 = 0;
56         #10 clk_50 = 1;
57       end
58
59   initial begin
60       // Initialize Inputs
61
62   // wait for neg edge on input clock and then apply data
63       (negedge clk_100);
64       data_in = 8'h1;
65       (negedge clk_100);
66       data_in = 8'h2;
67       (negedge clk_100);
68       data_in = 8'h3;
69       (negedge clk_100);
70       data_in = 8'h4;
71       (negedge clk_100);
72       data_in = 8'h5;
73       (negedge clk_100);
74       data_in = 8'h6;
75       (negedge clk_100);
76       data_in = 8'h7;
77       (negedge clk_100);
78       data_in = 8'h8;
```

Jim Duckworth, WPI

Synthesis and Timing - Module 11
FIFO Simulation