Lab 03
Parameterized Design

Due on Oct 30th 2016 11:59pm

黃稚存

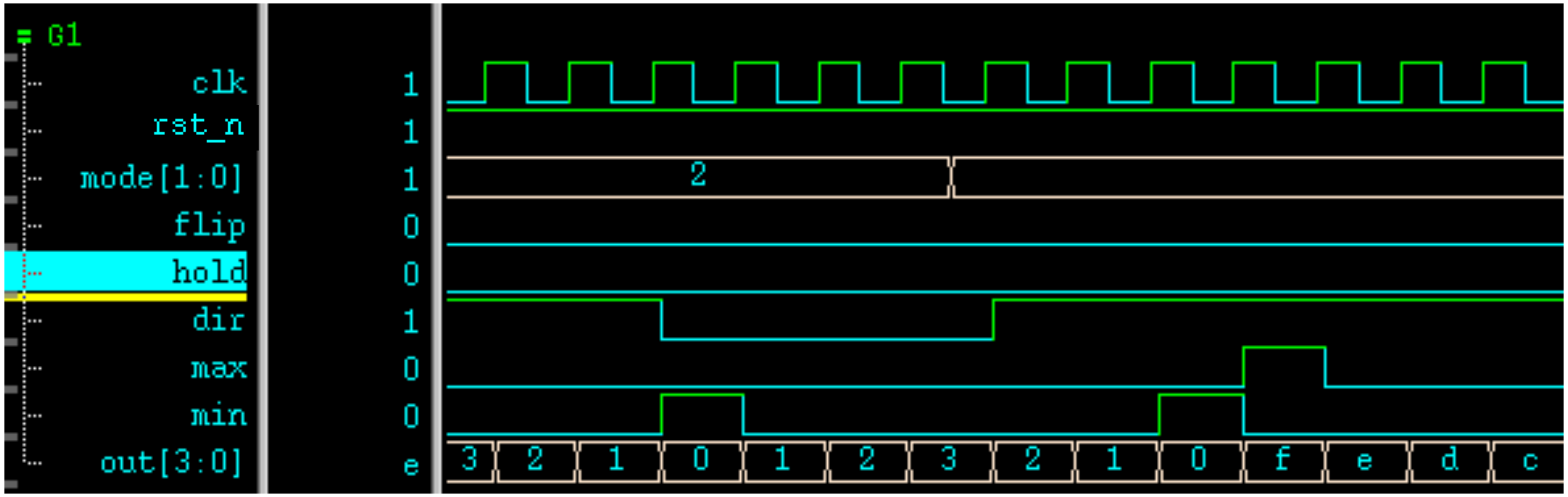
Department of Computer Science
National Tsing Hua University

Revisit Ping-Pong Counter

- Your customer does not satisfy the features of the ping-pong counter. So an additional input signal is now inserted
  - mode (2-bit input)
    - if mode == 2'b00, the circuit behaves as an up counter;
    - if mode == 2'b01, the circuit behaves as a down counter;
    - otherwise, the circuit behaves as the original ping-pong counter.
- To simplify the design, the input `flip` is kept low in the up-counter and down-counter modes.

Timing Example for Mode Change

- Using 4-bit counter as an example
  - To simplify the design, the counter output is reset to 4’b0000
  - The up-counter mode: counts from 4‘b0000 up to 4'b1111; then starts with 4'b0000 again.
  - The down-counter mode: counts from 4‘b1111 down to 4'b0000; then starts with 4'b1111 again



Test Stimulus

- Refer the code segment to build your own test stimulus

```
clk = 0;
rst_n = 1;
flip = 0;
hold = 0;
mode = 2'b10;
#(delay) rst_n = 0;
#(period/2*5+delay) rst_n = 1;
#(period/2-delay);
for (i = 0; i < pattern_count; i = i + 1) begin
  if (pattern[i] !== 4'bx) begin
    #(period)
      flip = pattern[i][3];
      hold = pattern[i][2];
      mode = pattern[i][1:0];
  end
end
$finish;
```

```
apply_pattern(pattern[i]);
```

Test Stimulus

- Also the following code segment to compare the outputs with the golden ones, cycle by cycle

```
    gold_out = golden[i][data_width - 1:0];
    gold_max = golden[i][data_width];
    gold_min = golden[i][data_width + 1];
    gold_dir = golden[i][data_width + 2];
    if (out !== gold_out) begin
      $display("Error at pattern %d: out=%h exp ected=%h!",
        i, out, gold_out);
    end
    if (dir !== gold_dir) begin
    ......
```

- Some students encounter mismatch problem even with (possibly) correct results. Sometimes you can observe a one-cycle shift between the results and the given golden responses with nWave. Why?
- If to the end you cannot resolve this issue, try to fix it in your way. But you have to identify this to TA during the demo.

Applying Some Modeling Techniques

- Apply the pattern by using a task `apply_pattern(pattern[i]);`
  - You may make use of task as much as possible
- Use an additional signal `cmp_out` to indicate any mismatched counter output
  - Cycle-based signal triggered by the positive clock edges
  - Asserted if matched; deasserted if mismatched
- Use an integer `err_out` to count the number of errors (mismatches)
- Compare other outputs as well
  - E.g., using `cmp_dir`, `cmp_min`, `cmp_max`; `err_dir`, `err_min`, and `err_max`

Text-Based Debug Information

- Show the following messages during the simulation
  - Header:
    ```
    Pattern file: lab03_pat1.dat
    Response file: lab03_pat1gold.dat
    ```
  - Error info: pattern index, error and golden outputs:
    ```
    Error at pattern    29: out=0f expected=70!
    ```
  - Summary at the end of the simulation:
    ```
    Signal out: error count =    3
    ```

Using Parameters

- Define the following parameters initially
  - width is 4
  - period is 8ns
  - delay is 1ns
  - pattern_count can be a large enough number, say, 1024
- Parameterize all constants as you can
  - Can you use upper_bound and lower_bound instead of 4’b0000 and 4’b1111?
    Verify that you can change them.

Define Macro

- Use a header.v for the `define directives
  - Keep all `define macro within header.v
  - No `define in RTL design and test stimulus

Makefile Integration

- Use a Makefile to integration all the simulation
  - Use `make pp` (or simply make) to preform Verilog simulation
  - Debug level
    ```
    make debug=2 # all messages
    make debug=1 # only header and summary
    make debug=0 # no text-based output
    ```
  - Counter width, clock period, and delay
    ```
    make width=6 clkperiod=8 delay=1
    ```
  - Input pattern file and golden response file
    ```
    make pattern=lab03_pat1.dat \
         golden=lab03_pat1gold.dat
    ```
  - Fsdb output
    ```
    make fsdb=somefile.fsdb
    ```

Shell Script to The Rescue

- Use proper shell scripts to make the demonstration easier, for example

```
#!/bin/sh
make pp \
  width=6 \
  pattern=lab03_pat1.dat \
  golden=lab03_pat1gold.dat
```

Lab Requirement

- Code your design with these files:
  - Makefile
  - lab03_header.v
  - lab03_pingpong_t.v
  - lab03_pingpong.v
- Meet all the requirements in the slides
- Follow TA's instructions about demo
- A free-format report has to be delivered to brief
  - the design concept, and
  - simulation patterns and setup.