IC Design
HW4 Tutorial

TA: Pi-Chuan Chen
Advisor: Tzi-Dar Chiueh
2023/11/30

Outline

- Workflow & Notification of HW4
- Standard Cell Library
- Tips
- Verification
- Reminder

Workflow (1/2)

- Grading
  - Approximation error (35%)
  - Performance (35%)
  - Report (30%)

- First, design a circuit that can finish the simulation.

```
============================================
          Simulation finished
============================================
```

- Second, decrease the CYCLE in tb.v until it fail (CYCLE < critical path). e.g some output values and corresponding MSE become x.

Workflow (2/2)

- Third, find the MSE and number of transistors in your design.

```
================================================
                    Summary
================================================
  Clock cycle:                  10.0 ns
  Number of transistors:        1061
  Total excution cycle:         256
  Approximation Error Score:    17987801.0
  Performance Score:            2716160.0
================================================
```

- Finally, modify you design to achieve better accuracy and performance.
  - Trade-off between area & speed. e.g. Try different adders, carry-skip, carry-lookahead, etc.
  - Try different algorithms. e.g. try different number of segments, different points, different slopes.

Notification (1/2)

- In this HW, all the logic operations MUST consist of standard cells (defined in lib.v). You can NOT use logic operators.

```
wire a, b, c;
assign a = b & c;
```

→ Behavioral Modeling

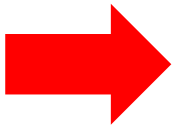
```
wire a, b, c;
AN2 an(a, b, c);
```

→ Structural Modeling

Notification (2/2)

- Do NOT change any module name and port name in sigmoid.v, just modifiy the module description, otherwise you can’t pass the simulation.

```
module sigmoid (
    input          clk,
    input          rst_n,
    input          i_in_valid,
    input  [ 7:0] i_x,
    output [15:0] o_y,
    output         o_out_valid,
    output [50:0] number
);
```



Don’t Change

- Use FD2 (positive edge) module for flip flop.

臺灣大學

Standard Cell Library

Standard Cell Library (lib.v)

- Choose what you need
- Compose your circuit according to I/O connections

- IV // not
- AN3
- AN4
- AN2
- EN // xnor
- EN3
- EO // xor
- EO3
- FA1 // full adder
- FD1 // negative edge DFF
- FD2 // positive edge DFF
- ND2 // nand
- ND3
- ND4
- NR2 // nor
- NR3
- OR2 // or
- OR3
- OR4
- HA1 // half adder
- MUX21H // 2-to-1 MUX

Number of Transistors

- “+” operator is only allowed to add up the transitor count of each cell.

```
module Reg3(
    input         CLK,
    input         RESET,
    input  [ 2:0] DD,
    output [ 2:0] Q,
    output [50:0] Reg3_num
);

wire [50:0] FD_num0, FD_num1, FD_num2;
assign Reg3_num = FD_num0 + FD_num1 + FD_num3;

FD2 fd0(Q[0], DD[0], CLK, RESET, number);
FD2 fd1(Q[1], DD[1], CLK, RESET, number);
FD2 fd2(Q[2], DD[2], CLK, RESET, number);

endmodule
```

```
module DUT (
    input         clk,
    input         rst_n,
    input  [ 2:0] A,
    input  [ 2:0] B,
    output [ 2:0] C,
    output [50:0] DUT_num
);

wire [50:0] num0, num1;
assign DUT_num = num0 + num1;

wire [2:0] A_reg, B_reg;
Reg3 rr0(A_reg, A[2:0], clk, rst_n, num0);
Reg3 rr0(B_reg, B[2:0], clk, rst_n, num1);

endmodule
```

臺灣大學

Tips

Parameterized module

- Verilog “generate” statement
  - Copy the codes.
  - Instantiate multiple modules easily.

- Use generate and parameter to define parameterized module.
  - High flexibility.
  - High readability.
  - Shorten your codes.
  - Prevent wire misconnection.

Example - n-bit DFF

```
module REGP#(
    parameter BW = 2
)(
    input           clk,
    input           rst_n,
    output [BW-1:0] Q,
    input  [BW-1:0] D,
    output [  50:0] number
);

wire [50:0] numbers [0:BW-1];

genvar i;
generate
    for (i=0; i<BW; i=i+1) begin
        FD2 f0(Q[i], D[i], clk, rst_n, numbers[i]);
    end
endgenerate

//sum number of transistors
reg [50:0] sum;
integer j;
always @(*) begin
    sum = 0;
    for (j=0; j<BW; j=j+1) begin
        sum = sum + numbers[j];
    end
end

assign number = sum;

endmodule
```

→ Generate FD2 according to BW.

→ Use for loop to sum transistor counts.

Instantiate in top module.

↑

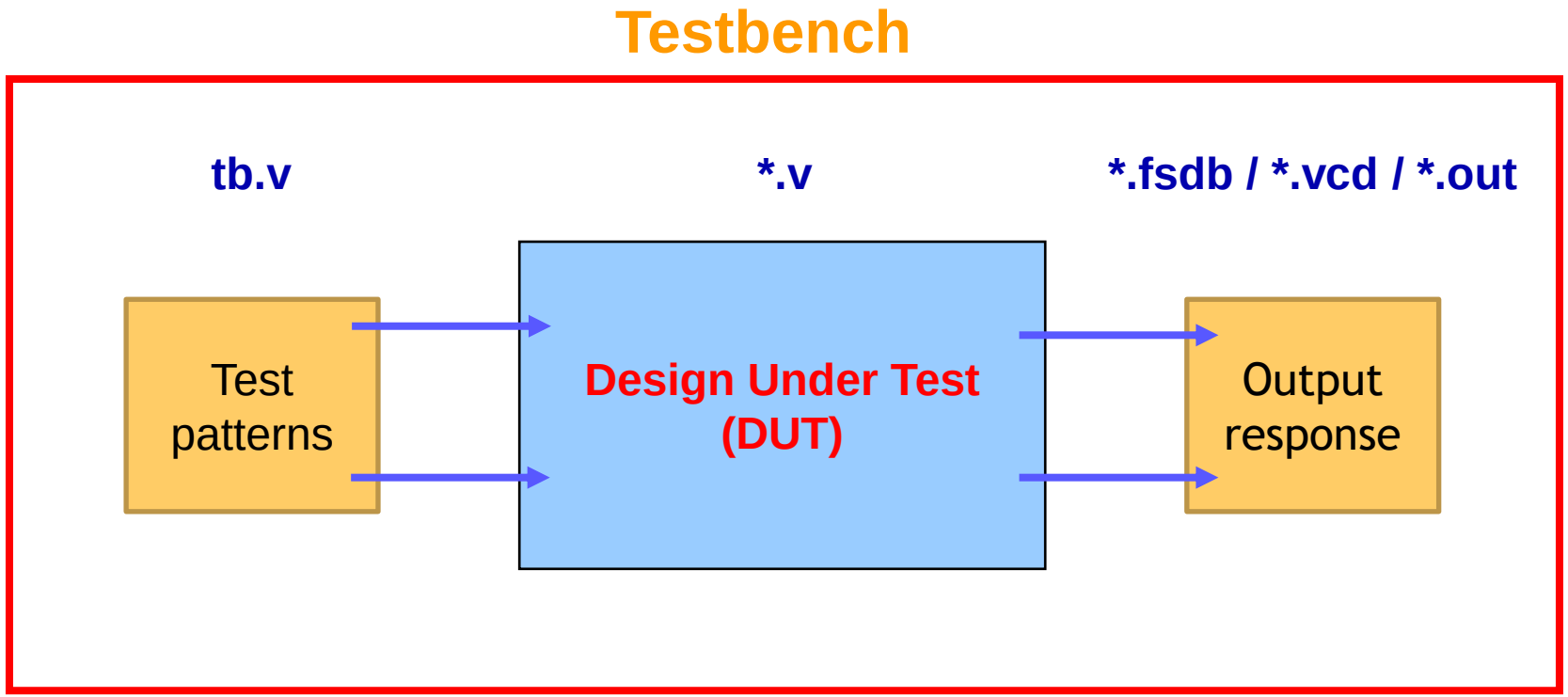
```
REGP#(.BW(5)) U_REGP( .clk(clk), .rst_n(rst_n), .Q(q), .D(D), .number(number));
```

臺灣大學

Verification

Test and Verify Your Circuit

- By applying input patterns and observing output responses



Compile and debug

- Source
  - source /usr/cad/synopsys/CIC/vcs.cshrc
- Include the tb.v & lib.v files to run simulation
  - vcs tb.v sigmoid.v lib.v -full64 -R -debug_access+all +v2k
  - vcs tb.v sigmoid.v lib.v -full64 -R -debug_access+all +v2k +define+DEBUG
  - vcs tb.v sigmoid.v lib.v -full64 -R -debug_access+all +v2k +define+PIPELINE
  - vcs tb.v sigmoid.v lib.v -full64 -R -debug_access+all +v2k +define+DEBUG+PIPELINE

Reminder (1/2)

- Loosen the clock cycle when you're checking your circuit logic.
- Once the logic is correct, start to shorten the clock period to find the critical path.
- Use basic gates provided in lib.v to design your circuit. No behavior level code will be accepted.

Reminder (2/2)

- Due on 2023/12/14 9:00
- For any further questions, contact TA.
  - r11943013@ntu.edu.tw
- To know more about Verilog, refer to
  - http://www.ece.umd.edu/courses/enee359a.S2008/verilog_tutorial.pdf