Homework 1
Adder Designs Using Verilog Structural, Dataflow, and Behavioral Modeling

Handout: 2024/09/24

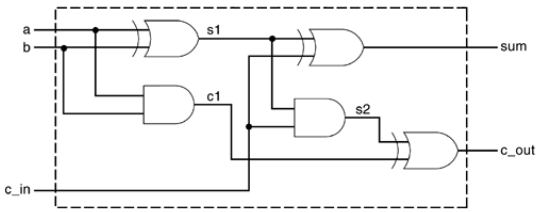
Due 2023/10/07

1. Design a 32-bit adder Using different Verilog Modeling Methods

(1) Structural Modeling

Use the Verilog built-in gate primitives (**and, or, xor, nand, nor, xnor, not**) to design a 1-bit full adder (FA) whose logic gate implementation is shown below. Name the FA module as "FA".

```
module FA (sum, c_out, a, b, c_in);
output sum;
output c_out;
input a, b;
input c_in;
...
endmodule
```



Then you can design 32-bit ripple carry adder by cascading the FA cells. Use the following names for the adder module and the corresponding input/output ports.

```
module adder_structure (s, co, a, b, ci);
parameter width = 32;
output [width-1:0] s;
output co;
input [width-1:0] a, b;
input ci;
...
// put 32 cascaded FA cells here using module instantiation
```

(2) Dataflow Modeling

Write the dataflow description for a 32-bit adder by using continuous assignment **assign** and the Verilog operator +. Use the following names for the adder module and the corresponding input/output ports.

```
module adder_dataflow (s, co, a, b, ci);
parameter width = 32;
output [width-1:0] s;
output co;
input [width-1:0] a, b;
input ci;

assign ...
…
endmodule
```

(3) Behavioral Modeling

Model a 32-bit adder using Verilog behavioral-level statements inside the **always** procedure block. Note that use Verilog operator + and the blocking assignment =. Use the following names for the adder module and the corresponding input/output ports.

```
module adder_behavior (s, co, a, b, ci);
parameter width = 32;
output [width-1:0] s;
output co;
input [width-1:0] a, b;
input ci;

reg [width-1:0] s;
reg co;

always @ (…)
begin
…
end
```

2. Outputs Stored in Registers

The above three different adder designs are pure combinational logic where the outputs are not latched, i.e, the outputs are not stored in flip flops. Use flip-flops to store the results of the outputs in the three different adder designs in the previous problems. Verilog code for a 1-bit flip-flop (FF) is shown below:

```
module D_FF (q, d, clk);
output q;
input d, clk;
reg q;
always @( posedge clk)
        q <= d;
endmodule
```

Now your module should have an additional input clock signal named “clk”. Name the 32-bit adder modules as “adder_structure_reg”, “adder_dataflow_reg”, and “adder_behavior_reg”.

3. RTL Simulation (Pre-Synthesis Simulation)

Write a single testtench module to jointly verify the function of the above six different 32-bit adder designs simultaneously. First instantiate the six 32-bit adder modules. Then, generate inputs of 10 test patterns using Verilog system task **$random** [1]. Compare with the outputs from the modules with your expected results. Note that your expected results could be generated using Verilog operator + inside the testbench module.

4. Logic Synthesis and Gate-Level Simulation (Post-Synthesis Simulation)

Use Synopsys Design Compiler (DC) with properly specified constraints to synthesize the RTL designs into gate-level netlists based on TSMC40nm standard cell library. Since the DC logic synthesis depends on user-specified constraints, you should synthesize your design with at least three different constraints: *delay-optimization*, *area-optimization*, and *in-between* (with reasonable delay constraint). Compare the differences of area/delay/power for the these three synthesis results. In general, the curve of area versus delay looks like a reciprocal curve.

Run simulation in gate level to verify the design again, using the same inputs as those in the RTL simulation. What are the area, delay, and power of the synthesis results? Note that total power consists of dynamic power and leakage power. Dynamic power is in general linearly proportional to the frequency while leakage power is usually independent of working frequency. What is the frequency used for the power?

5. Comparison of Synthesis Results

Identify the critical path delays and area cost of all the designs. Comments on the delays and area of the designs in different modeling methods. Provide your own comments on the delays of area. For example, which design has the smallest area? Which design has the

smallest delay? What are the differences of delays and area of fully combinational logic designs and their respective output-registered counterparts?

		Area (um^2)			Delay	Power (W)		
		CL	SL	Total	(ns)	dynamic	leakage	total
adder_structure	delay							
	area							
	between							
adder_structure_reg	delay							
	area							
	between							
adder_dataflow	delay							
	area							
	between							
adder_dataflow_reg	delay							
	area							
	between							
adder_behavior	delay							
	area							
	between							
adder_behavior_reg	delay							
	area							
	between							

References:

[1] IEEE Standard for Verilog Hardware Description Language, Chap. 17, IEEE Std 1364-2005, Apr. 2006.

Report Requirement

The report should clearly explain your design and how you verify your design. Upload the homework reports to the course website with proper file names showing student ID and homework number. Example: HDL HW1 M1130400xx 王小明

Your attachment should include HDL codes and all the supporting document (such as results from EDA tools and your explanations).

1. VCS Simulation and Logic Synthesis with Synopsys Design Compiler (100%)

檔案類須含有:

I. Testbench **(15%)**
   -Testbench 需使用$random 產生 10 筆亂數輸出(5%)
   - Testbench (10%)

II. Verilog RTL codes and Synthesized gate-level codes(.ddc、.v、.sdf、.sdc) **(60%)**
   -adder_structure (10%) and adder_structure_reg (10%)
   -adder_dataflow (10%) and adder_dataflow_reg (10%)
   -adder_behavior (10%) and adder_behavior_reg (10%)

PDF 報告須含有:

III. RTL and gate-level 波形 (delay optimize 即可) **(5%)**

IV. Area/Delay 比較數據&截圖 **(10%)**

V. 請觀察三種 modeling 的數據與波型是否相同，並解釋你認為的原因。 **(5%)**

VI. 心得 **(5%)**

以上打包成 MXXXXXXXXX.zip 壓縮檔並繳交