Lab 04

Verilog: Datapath and Control Circuit in Verilog

In this lab, you are going to learn how to model a shift-and-add multiplier with hierarchical design method and how to create test bench to verify your design.

Getting Started

In lecture, we learned that a digital system can be built with two components: datapath and control circuit. We used shift-and-add multiplier as a starting example. The datapath of the multiplier is shown in Figure 1.

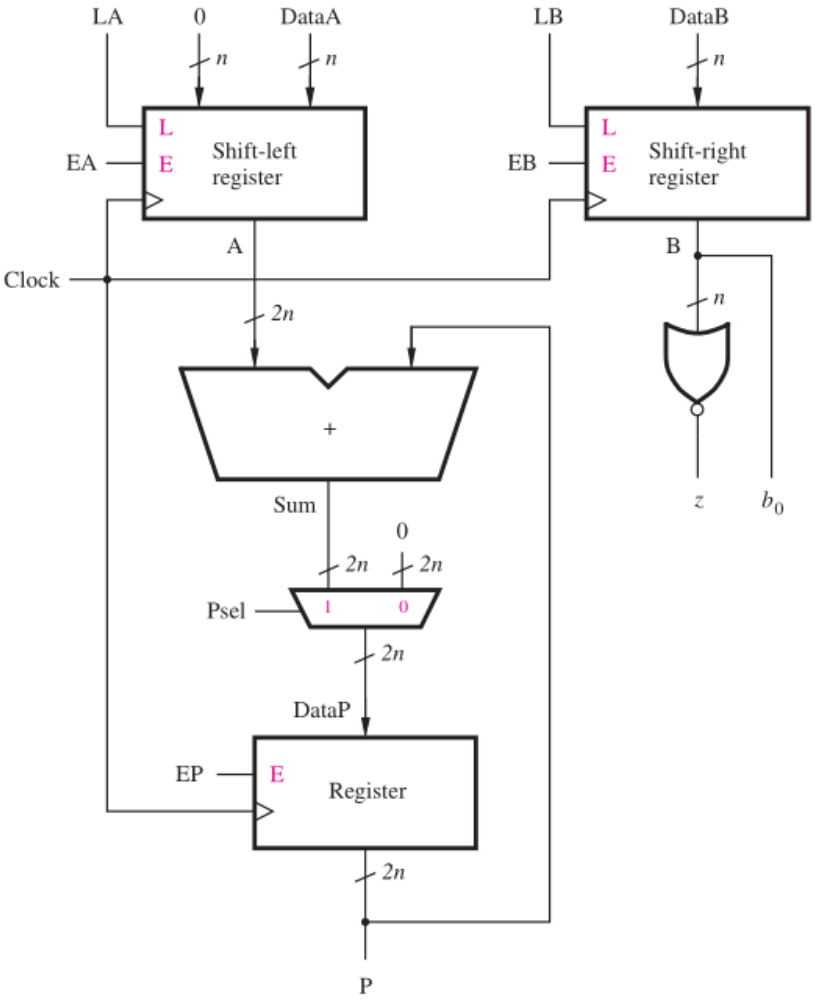
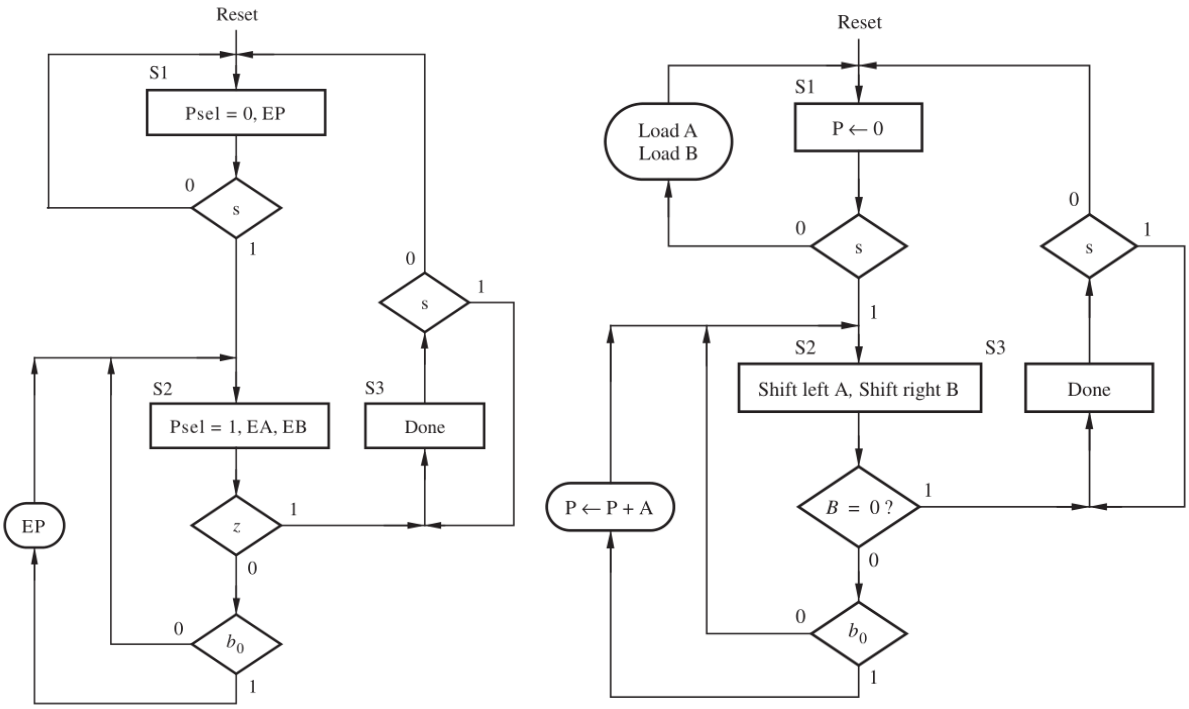


Figure 1, Datapath circuit for Shift-and-Add Multiplier

The flow charts for datapath operation and control circuit are shown below. The Verilog description in RTL of the shift-and-add multiplier is provided in the lecture note.



1. Create a folder for your lab04 assignment in your home directory “computer_system_18sp”. Name it ‘lab4’. Change to this folder.
2. Download “lab04.tar.gz” from MySTU and save it in “lab4” folder. Extract the file with command

   ```
   tar zxvf lab04.tar.gz
   ```

   You will see four files *registers.v multiplier.v, multiplier_hier.v,* and *multiplier_tb.v*. The first two files contain the complete module descriptions as shown in the lecture note. You are going to edit the remaining two files to finish this lab.

Assignments of Lab 4

1. As shown in *multipler.v,* the datapath and control circuit parts are included in the same module. You are required to divide them into two modules such that the top level multiplier is composed of two sub-modules as shown in *multipler_hier.v*.

```
// Top level multiplier module
module multiplier_hier (Clock, Resetn, LA, LB, s, DataA, DataB, P, Done);

  parameter n = 8;
  input Clock, Resetn, LA, LB, s;
  input [n-1:0] DataA, DataB;
  output [n+n-1:0] P;
  output Done;

  wire Done, EA, EB, EP, Psel;
  wire b_0, z;

  mul_control Cntl( .clock(Clock), .rstn(Resetn), .b0(b_0), .zero(z),
                    .start(s), .en_a(EA), .en_b(EB), .en_p(EP), .done(Done),
                    .psel(Psel) );

  mul_datapath DPath(.clock(Clock), .rstn(Resetn), .din_a(DataA), .din_b(DataB),
                     .ld_a(LA), .en_a(EA), .ld_b(LB), .en_b(EB),
                     .en_p(EP), .psel(Psel),
                     .b0(b_0), .zero(z), .product(P) );

endmodule // multiplier_hier
```

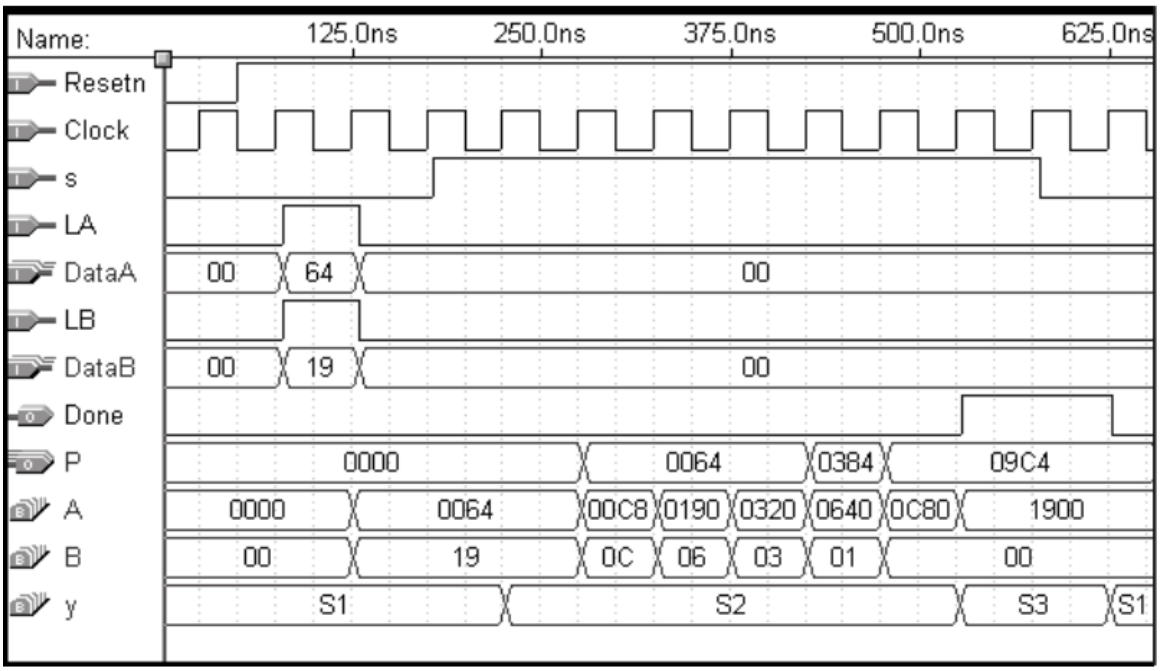
2. Finish the test sequence in *multiplier_tb.v* such that your test bench can complete two rounds of

multiplication. For example, your test should test Ain=8, Bin=7 first and then Ain=10, Bin=15.

```
`timescale 1ns/1ns

module multiplier_tb;

  reg [7:0] Ain, Bin;
  reg rstn, clk, loadA, loadB, start;
  wire [15:0] product;
  wire done;

  multiplier_hier U1(.Clock(clk), .Resetn(rstn), .LA(loadA), .LB(loadB),
                     .s(start), .DataA(Ain), .DataB(Bin),
                     .P(product), .Done(done) );

  // define clock
  always
  begin
    clk = 0;
    #5;
    clk = 1;
    #5;
  end

  // defint test sequence
  initial
  begin
    // Please define the test sequence by yourself
  end

endmodule // multiplier_tb
```

What to Turn in

- Submit to MySTU
  - Your complete Verilog file *multiplier_hier.v* and *multiplier_tb.v*..
  - Screen shot of simulation waveform that shows two rounds of multiplication. Below is an example waveform of one multiplication in the lecture note.



- Demonstrate your simulation in front of a Teaching Assistant, answer the questions, and sign your name.