# Arithmetic Circuit Designs with Pipelines

*Verilog Lab 6*

Chase A. Lotito, *SIUC Undergraduate*

## Section A: 16-bit adder without pipelining.

```
1  // chase lotito - ece426 lab 6 - adder a
2
3  // Addition of two 16 bits, 2's complement nos., n1 and n2.  All the 16
       bits addition at one stroke without any pipeline.
4  // Result is 17 bits.
5
6  `timescale 1us / 1us
7
8  module adder_a (clk, n1, n2, sum);
9
10 // i/o
11 input clk;
12 input [15:0] n1, n2;
13 output [16:0] sum ;
14
15 reg [16:0] sum;
16 wire [16:0] sum_i;
17
18 assign sum_i = {n1[15], n1} + {n2[15], n2};    // Add after sign
       extension - Result (i).
19
20 // Frequency of operation is not relevant since this is a combination
       circuit.
21 // Pipeline 1, clk (1), register final result (i).
22 always @ (posedge clk) begin
23     sum <= sum_i;
24 end
25
26 endmodule
```

```
1  // Test bench for adder_a design
2
3  `define CLKPERIODBY2 5 // Operate at 50 MHz.
4
5  `timescale 1us / 1us
6  `include "adder_a.v" // This is the design.
7
8  module adder_a_testbench;
9
```

```
reg clk;
reg [15:0] n1;
reg [15:0] n2;
wire [16:0] sum;

adder_a U1 (.clk(clk), .n1(n1), .n2(n2), .sum(sum) );

// GTKWAVE (waveform simulator)
initial begin : GTKWAVE
    $dumpfile("adder_a.vcd");
    $dumpvars(0, adder_a_testbench);
end

initial begin // Apply different sets of inputs.
    clk = 1'b0; n1 = 16'h0; n2 = 16'h0;

    #10 n1 = 16'h0001; n2 = 16'h0101;
    #10 n1 = 16'h0fff; n2 = 16'h0fff;
    #10 n1 = 16'h7fff; n2 = 16'h7fff;
    #10 n1 = 16'h8000; n2 = 16'h8000;
    #10 n1 = 16'h0001; n2 = 16'h0001;
    #10 n1 = 16'hffff; n2 = 16'h0001;
    #10 n1 = 16'h7fff; n2 = 16'h7fff;
    #10 n1 = 16'h5555; n2 = 16'haaaa;
    #10 n1 = 16'h0000; n2 = 16'h0000;

    #20 $finish;
end

initial $monitor("[%0t us] clk =%b n1 = %h n2 = %h sum = %h", $time,
   clk, n1, n2, sum);

always #`CLKPERIODBY2 clk <= ~clk;  // Toggle the clock.

endmodule
```

## Section B: 16-bit adder with pipelining.

```
// chase lotito - ece426 lab 6 - adder b
// Addition of two 16 bit, 2's complement nos., n1 and n2. 8 bits
   addition at a time. Result is 17 bits.

module adder_b ( clk, n1, n2, sum);

```

```
6  // i/o
7  input clk;
8  input [15:0] n1;
9  input [15:0] n2;
10 output [16:0] sum;
11
12 reg [16:0] sum;
13 wire [8:0] sum_LSB;
14 reg [8:0] sum_LSB_1;
15
16 reg [15:8] n1_reg1;
17 reg [15:8] n2_reg1;
18
19 wire [16:8] sum_MSB;
20 wire [16:0] sum_next;
21
22 assign sum_LSB = n1[7:0] + n2[7:0]; // Add least significant byte.
       sum_LSB [8] is the carry.
23
24 always @ (posedge clk)  // Pipeline 1, clk (1), register LSB to
       continue addition of MSB.
25 begin
26     sum_LSB_1 <= sum_LSB;   // Preserve LSB sum
27     n1_reg1 <= n1[15:8];    // Preserve MSBs
28     n2_reg1 <= n2[15:8];
29 end
30
31 // Extend sign & add msbs with carry.
32 assign sum_MSB = {n1_reg1[15], n1_reg1[15:8]} + {n2_reg1[15], n2_reg1
       [15:8]} + sum_LSB_1 [8];// Add MSBs with carry.
33 assign sum_next = {sum_MSB, sum_LSB_1 [7:0]};
34
35 always @ (posedge clk)// Pipeline 2, clk (2), register result.
36 begin
37     sum <= sum_next;
38 end
39
40 endmodule
```

```
1  // chase lotito - ece426 adder_b testbench
2
3  `define CLKBYPERIOD2 5// Operate at 50 MHz.
4  `include "adder_b.v" // This is the design.
5
6  module adder_b_testbench;
7
```

```
8  reg clk;
9  reg [15:0] n1;
10 reg [15:0] n2;
11
12 wire [16:0] sum;
13
14 adder_b U1 (.clk(clk), .n1(n1), .n2(n2), .sum(sum) );
15
16 initial begin// Apply different sets of inputs.
17  clk = 1'b0; n1 = 16'h0; n2 = 16'h0;
18
19  #10 n1 = 16'h0001; n2 = 16'h0101;
20  #10 n1 = 16'h0fff; n2 = 16'h0fff;
21  #10 n1 = 16'h7fff; n2 = 16'h7fff;
22  #10 n1 = 16'h8000; n2 = 16'h8000;
23  #10 n1 = 16'h0001; n2 = 16'h0001;
24  #10 n1 = 16'hffff; n2 = 16'h0001;
25  #10 n1 = 16'h7fff; n2 = 16'h7fff;
26  #10 n1 = 16'h5555; n2 = 16'haaaa;
27  #10 n1 = 16'h0000; n2 = 16'h0000;
28
29  #10 $stop;
30  $finish;
31 end
32
33 initial $monitor("[%0t us] clk =%b n1 = %h n2 = %h sum = %h", $time,
       clk, n1, n2, sum);
34
35 always #`CLKBYPERIOD2 clk <= ~clk;// Toggle the clock.
36
37 endmodule
```