EE5516 VLSI Architectures for Signal Processing and Machine Learning Lab Report: Experiment 1

Subham Ball
Dept. of System-on-chip Design
Indian Institute Of Technology–Palakkad
152202017@smail.iitpkd.ac.in

***Abstract*—Experiment 1's goal is to implement and validate the design of an architecture that computes the sum of the first N natural numbers for a given N.**

I. INTRODUCTION

The sum of N natural numbers is a common arithmetic operation used in many algorithms and computations. In VLSI architecture, efficient implementation of this operation is important for achieving high performance and low power consumption.
One of the most common methods for computing the sum of N natural numbers is the mathematical formula:

$$sum = \frac{(N \times (N+1))}{2}$$

This formula can be easily computed using basic arithmetic operations, but it requires a multiplier and a divider circuit which may not be efficient for VLSI implementation.
figure (Fig. 1)

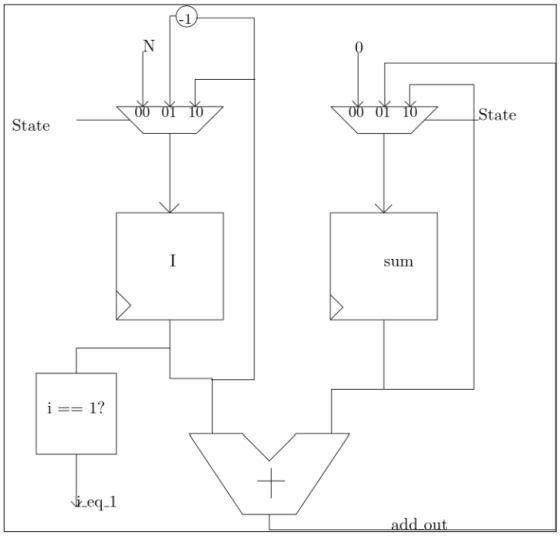

Fig. 1. Data Path of the sum of the first N natural numbers algorithm

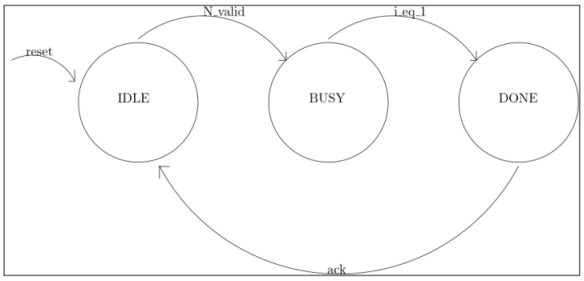

Fig. 2. Control Path of the sum of the first N natural numbers algorithm

II. IMPLEMENTATION DETAILS

Now let's see the Verilog Code:

design.sv

```
module n_natural(clk,n,n_valid,ack,reset,sum,sum_valid);
  input [2:0] n;
  input n_valid,clk,ack,reset;
  output  reg [4:0] sum;
  output sum_valid;
  reg [1:0] state;
  wire i_eq_1;
  n_natural_control_path sum1(.clk(clk),.n_valid(n_valid),.i_eq_1(i_eq_1),.ack(ack),.reset(reset),.state(state),.sum_valid(sum_valid));
  n_natural_data_path sum2(.clk(clk),.state(state),.n(n),.sum(sum),.i_eq_1(i_eq_1));
endmodule

module n_natural_control_path(clk,n_valid,i_eq_1,ack,reset,state,sum_valid);
  input clk,n_valid,i_eq_1,ack,reset;
  output reg [1:0] state;
  output sum_valid;
  reg [1:0] next;
  parameter idle=0,busy=1,done=2;
  always @(*)
    begin
    if(reset)
      next=idle;
      case(state)
        idle:next=n_valid?busy:idle;
        busy:next=i_eq_1?done:busy;
        done:next=ack?idle:done;
      endcase
    end
  always @(posedge clk)
    state<=next;
  assign sum_valid=(state==done);
endmodule

module n_natural_data_path(clk,state,n,sum,i_eq_1);
  input [1:0] state;
  input [2:0] n;
  input clk;
  output reg [4:0] sum;
  output i_eq_1;
  reg [2:0] i;
  wire [2:0] i_muxout;
  wire [4:0] sum_mux_out;
  assign i_muxout=state[1]?i:(state[0]?i-1:n);
  assign sum_mux_out=state[1]?sum:(state[0]?sum+i:0);
  always @(posedge clk)
    begin
      i<=i_muxout;
      sum<=sum_mux_out;
    end
  assign i_eq_1=(i==1);
endmodule
```

Fig. 3. RTL code

Now let's see the testbench for the same:

testbench.sv

```
module n_natural_tb;
  reg [2:0]n;
  reg clk,n_valid,ack,reset;
  wire [4:0] sum;
  wire sum_valid;
  n_natural n_nat(.clk(clk),.n(n),.n_valid(n_valid),.sum(sum),.sum_valid(sum_valid),.ack(ack),.reset(reset));
  always #5 clk=~clk;
  initial
    begin
      clk = 0;
      reset = 0;
      n_valid = 0;
      ack = 0;
      n = 7;
      #10
      reset = 1;
      #10
      reset = 0;
      n_valid = 1;
      #10
      n_valid = 0;
      #120
      $finish;
    end
  initial begin
      $dumpfile("dump.vcd");
      $dumpvars;
  end
endmodule
```

Fig. 4. Testbench OF DUT

III. Experimental results

Now, lets see the output for the simulation.

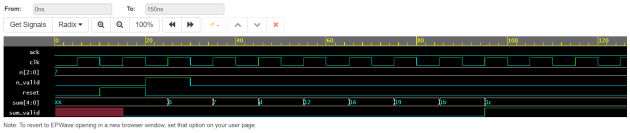

Fig. 5. Output Waveform

IV. Conclusion

As a result, we may infer from the output waveform of the simulation that the design is sound. In this experiment, we can implement the sum of n utilising the control route and datapath.The link to the EDA Playground is: playground link