Fibonacci Series Generator and verified the design using cadence software

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

***Abstract*—Implementing a N^{th} Fibonacci Generator in Verilog, validating the design, and carrying out the physical design are the objectives of Experiment 3. It helps us become more adept at managing control flow in Verilog and familiarise oneself with Verilog Simulation and physical design using Innovus.**

I. INTRODUCTION

Two modules are needed to create a N^{th} fibonacci generator. The Fibonacci series generator is represented by one module, while the N^{th} Fibonacci number is produced by the other module. Every time the clock pulse has a positive or negative edge, a 32-bit Fibonacci series generator generates a term from the series. When a positive or negative edge N_valid is applied, it is initialized. The N^{th} Fibonacci generator receives a 3-bit integer N as input, and uses N_valid and Fibo_valid as the qualifiers for the input N and Fibonacci series generated, respectively to produce N_Fibo_num, which is the required N^{th} Fibonacci number.

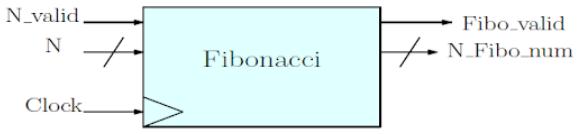

Fig. 1. Block diagram of FSM

II. IMPLEMENTATION DETAILS

Experiment 3 involves designing the N^{th} Fibonacci Generator illustrated in Fig. 1, testing this design on a testbench, and putting the physical design into practise.The current f(n) and prior f(n-1) values are stored in two registers that make up the Fibonacci generator. Additionally, it includes an adder, a combinational logic network that creates the sum of the past and current values. The register holding the current value is updated by getting the sum as the new data after the application of a positive or negative edge clock pulse, and the register storing the prior value is updated by receiving the current value. The output, or fibo_out, is set to the previous value. The register containing the prior value is initialised to a 32-bit binary zero on the positive edge of N_valid, while the register containing the current value is initialised to a 32-bit binary one.

The N^{th} Fibonacci generator module consists of a register count that keeps increasing until it hits input value N, assuming N_valid is logic 0. The N^{th} Fibonacci generator module will only accept N as an input when N_valid is on the negative edge, thus serving as a qualifier for input N. Once count reaches a value of N, the signal Fibo_valid changes to logic 1. It serves as the fibo_out qualifier to produce the N^{th} Fibonacci number, abbreviated N_fibo_num. Figure 2 depicts the architectural design.

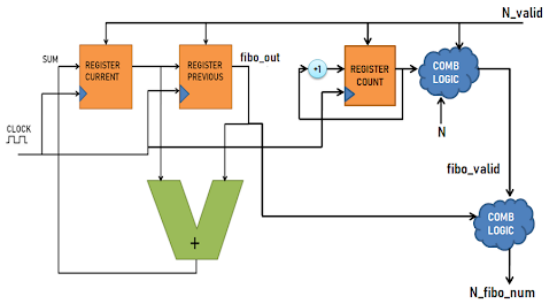

Fig. 2. Architectural diagram

Figure 3 provides the code for the RTL design of a N^{th} Fibonacci generator. Figure 4 depicts the testbench created for verifying the DUT.

```
module fibo_valid (clk, N_valid, fibo_out, fibo_valid_out, N, N_fibo_num);
  input clk, N_valid;
  input [2:0] N;
  output [31:0] N_fibo_num;
  output [31:0] fibo_out;
  output fibo_valid_out;
  reg [2:0] count = 3'b0;

  fibo fb (.clk(clk), .N_valid(N_valid), .fibo_out(fibo_out));
  always @(posedge clk)
    begin
      count <= N_valid? 3'b0: count +1;
    end
  assign fibo_valid_out = N_valid? 1'b0 : ((count == N)? 1'b1 : 1'b0);
  assign N_fibo_num = fibo_valid_out? fibo_out : 32'd0;
endmodule
module fibo (input clk, N_valid,
             output [31:0] fibo_out);
  reg [31:0] current, previous;
  always @(posedge clk)
    begin
      current <= N_valid? 32'b1 : current+previous ;
      previous <= N_valid? 32'b0 : current;
    end
  assign fibo_out = previous;
endmodule
```

Fig. 3. RTL code

```
module fbv_tb();
  reg clk, N_valid;
  reg [2:0] N;
  wire [31:0] N_fibo_num;
  wire [31:0] fibo_out;
  wire fibo_valid_out;

 fibo_valid fbv (.clk(clk), .N_valid(N_valid), .N(N), .fibo_out(fibo_out),
.N_fibo_num(N_fibo_num), .fibo_valid_out(fibo_valid_out));

  initial begin
  forever #1 clk = ~clk;
  end
  initial begin
    $monitor ("At time = %t, fibo_valid_out = %d, N_fibo_num = %d", $time,
fibo_valid_out, N_fibo_num);
    clk = 0; N_valid = 1; N = 3'b011;
    #2 N_valid = 0;
    #10 N_valid = 1;
    #2 N_valid = 0; N = 3'b101;
    #20 $stop;
  end
  initial begin
  $dumpfile("dump.vcd");
  $dumpvars(0);
  end
endmodule
```

Fig. 4. Testbench

III. Experimental Details

Cadence software is used to simulate and synthesise the Verilog design of a N^{th} Fibonacci generator, which produces fibo_valid and N_fibo_num signals after accepting N, N_valid, and clock as inputs. In Figures 5 and 6, respectively, the observed waveform and the synthesised circuit are depicted.

Fig. 5. Waveform

IV. Physical Design

The basis for physical design is a netlist, which is the result of the synthesis procedure. Synthesis transforms the RTL design into gate-level netlist that the following group of tools can read and understand. This netlist contains a list of the cells utilized, their connections, the area covered, and other details.

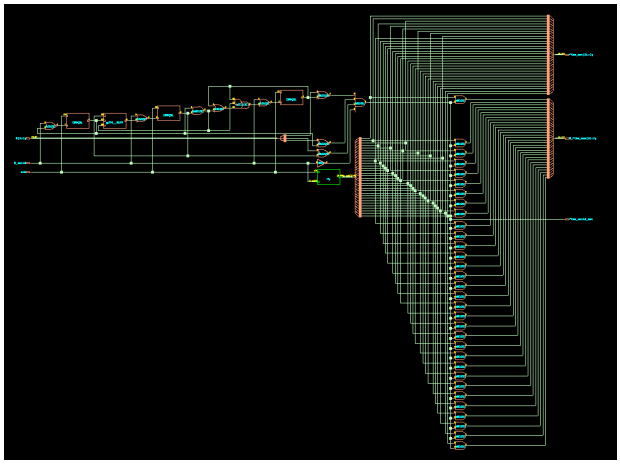

Fig. 6. Synthesized circuit

During the synthesis process, constraints are utilised to ensure that the design satisfies the required functionalities and speed criteria (specifications).

Only when it has been timed and functionally tested is the netlist submitted for the physical design flow.

The second phase of physical design is floorplanning. Selecting structures to be built adjacent to one another and allocating space for them in a way that satisfies the somewhat conflicting objectives for performance, close vicinity of components, and available space is the process of floorplanning.

Making ensuring that all of the design's macros, standard cells, and other cells have power is known as power planning. Power and ground nets are frequently positioned on the metal layers. This creates power and ground structures with both IO pads and core logic. The IO pads' power and ground buses are built within the pad itself, and they will be joined by an abutment. For core logic, a core ring encircling the core with one or multiple sets of power and ground rings is employed. The next consideration is the design of cell power and ground internal to core logic, commonly known as power and ground stripes. Within the design, these stripes are repeated at regular intervals across the zone. A ring distributes VDD and VSS around the chip, while a stripe conveys VDD and VSS from the ring.

After floorplanning and power planning which includes the creation of the core area, the placement of the macros, and the selection of the power network architecture for the design, standard cell placement is carried out. The placement process involves finding an acceptable physical location for each block cell. In addition to positioning the standard cell present in the synthesised netlist, placement optimises the design. The tool determines the location of each standard cell on the core. Multiple factors, such as time, congestion, and power optimization, will affect placement. Timing analysis is now performed to look for setup violations, and as shown in Figure 7, no violating pathways are found.

Clock tree synthesis (CTS) is a technique for equally

Setup mode	all	reg2reg	default
WNS (ns):	1.431	1.431	8.684
TNS (ns):	0.000	0.000	0.000
Violating Paths:	0	0	0
All Paths:	67	67	67

Fig. 7. Pre-CTS timing analysis

spreading clock among all sequential elements of a VLSI design. The goal of clock tree synthesis is to reduce skew and insertion delay. The clock is not transmitted before CTS. Hold slack should improve after CTS. The clock tree begins at the clock source specified in .sdc and ends at the stop pins of the flip-flop. Timing analysis is now being done to check for setup and hold violations, and as can be seen in Figure 8, there are no violating paths found.

Setup mode	all	reg2reg	default
WNS (ns):	1.331	1.331	8.680
TNS (ns):	0.000	0.000	0.000
Violating Paths:	0	0	0
All Paths:	67	67	67

Hold mode	all	reg2reg	default
WNS (ns):	0.000	0.000	0.000
TNS (ns):	0.000	0.000	0.000
Violating Paths:	0	0	0
All Paths:	67	67	0

Fig. 8. Post-CTS timing analysis

Routing entails using metal layers to physically connect signal pins. Routing is the stage after CTS and optimization where precise links between standard cells, macros, and I/O pins are built. Once more, post route time analysis is performed, and as shown in Figure 9, no violating pathways are found.

Figure 10 elaborates on the physical design circuit that INNOVUS was used to create in Cadence software.

Setup mode	all	reg2reg	default
WNS (ns):	1.294	1.294	8.684
TNS (ns):	0.000	0.000	0.000
Violating Paths:	0	0	0
All Paths:	67	67	67

Hold mode	all	reg2reg	default
WNS (ns):	0.001	0.001	0.000
TNS (ns):	0.000	0.000	0.000
Violating Paths:	0	0	0
All Paths:	67	67	0

Fig. 9. Post-Route timing analysis

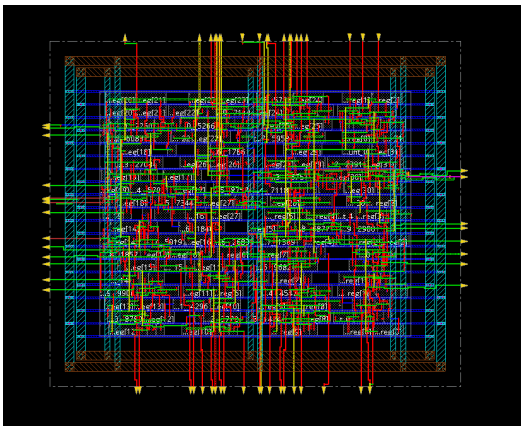

Fig. 10. physical design circuit

V. Analysis

The following table (Figure 11) compares the physical designs for the pre-CTS, post-CTS, and post-route in terms of power and area.

	Pre-CTS	Post-CTS	Post-Route
Power (mW)	0.14675289	0.17898056	0.17941010
Area (μm^2)	2182.1427	2582.5428	2582.5428

Fig. 11. Table

VI. Causing setup violations in the design

In VLSI design, a path consists of a flip-flop (FF1), combinational logic, and then another flip-flop (FF2). Data must be kept at D of FF2 for Tsetup time until the clock tree transmits the next positive or negative edge of the clock to FF2 in order for FF2 to successfully latch it. The time required for the data to propagate to FF2, counting from the clock edge at FF1, is invariably equal to $T_{clk-Q} + T_{comb}$. The equation

should therefore be as follows in order to satisfy the setup time requirement.

$$T_{clk-Q} + T_{comb} + T_{setup} \leq T_{clk}$$

Therefore, the clock period must be shortened in order to cause a setup time violation. The following timing violation is seen in this experiment when the clock period is reduced from 10 units to 0.5 units, as shown in Figure 12.

Setup mode	all	reg2reg	default
WNS (ns):	-1.388	-1.388	-0.988
TNS (ns):	-69.695	-40.039	-51.013
Violating Paths:	75	67	67
All Paths:	75	67	67

Fig. 12. Timing analysis after setup violation

VII. Conclusion

The N^{th} Fibonacci generator is successfully simulated and synthesised using Cadence software after implementation in Verilog design. INNOVUS is used for the physical design, and timing, area, and power values are examined at several stages of design, namely pre-CTS, post-CTS, and post-route. Changing the clock period also causes a setup violation, which helps us better understand timing analysis.