California State Polytechnic University, Pomona

Lab 3: 16-bit Full Adder with 1x2 Demultiplexer

Group F:

Danny Manzo, Bronco ID: 014451190

Maher Mateen, Bronco ID: 013891189

ECE 3101L – Signals and Systems Laboratory

Professor Mohamed Aly

27 June 2022

1. **How a Full Adder Works:**

A Full Adder adds 2 binary inputs with a carry in input and 2 outputs with a sum output and a carry out output. To find the sum, you exclusive or (XOR) the two inputs and carry in input. Meanwhile, to find the carry out, you AND the 2 inputs, AND one input with the carry in input, and AND the other input with the carry in, before ORing all 3 results together.

1. **How a Demultiplexer Works:**

A demultiplexer is the inversion of a multiplexer. It contains a single binary input with a select input which gives several outputs. Depending on if our select input is a low bit or a high bit, this affects the outputs of the demultiplexer. A demultiplexer exists because it helps reduce the complexity of a circuit since a multiplexer has gates and it helps reduce cost.

1. **How We Coded It:**

To code the 1x2 demultiplexer, we used an always block to assign the 16-bit input obtained from the switches to one of two outputs, A or B, depending on whether the select input was 1 or 0. If the select input was 1, we stored it into output B, and if the select input was 0, we stored it into output A. The sensitivity list for the always block was the 16-bit input and the select input. To code the full adder, we used an assign statement for both the sum and carry out outputs, having the sum equal to the exclusive or (XOR) of the 2 inputs A and B, which was obtained from the demultiplexer, and the carry in input, while the carry out was the ANDing of 3 combinations of the 3 inputs in pairs of 2 (A&B, A&Cin, and B&Cin), before ORing the results from the AND operation. To combine the demultiplexer and full adder together, first we created a 17-bit wire called “wire\_RCA” to hold the carry bits from the ripple carry adder design we are making. The first bit of “wire\_RCA” is our initial carry in while the final bit is our final carry out. To make sure the full adder had two inputs and a carry-in, we first called our demultiplexer module, then we generated a for loop that loops 16 times through our full adder module. This means that we are adding each of the 16 bits individually, thus generating 16 full adders. We used our genvar i as an index for the 16-bit inputs and outputs. Finally, we stored each of the carry bits in our 17-bit “wire\_RCA” as we went through each full adder in the loop. The index for our carry out was “i+1” to ensure that the carry out of the previous full adder became the carry in of the next. Lastly, we assigned our carry out output to be equal to the final bit of “wire\_RCA”, which is the carry out of the 16th full adder.

1. **Vivado Data Collection:**

In Vivado, after running the synthesis and report utilization, we found that there was 24 look-up tables (LUTs), 32 slice registers, 24 slices, 24 LUTs as logic, 36 bonded input/output bits (IOB), and 1 BUFGCTRL. The total power of the full adder with the demultiplexer was reported to be 11.752 W.

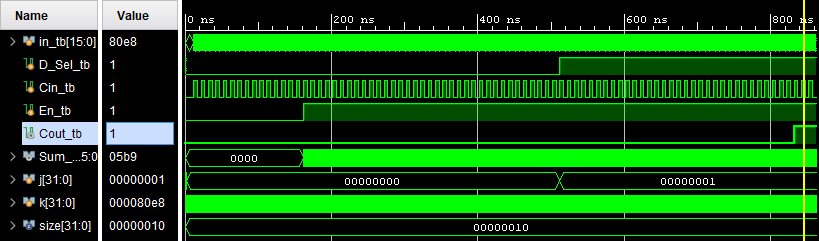


Figure 1: Testbench Results

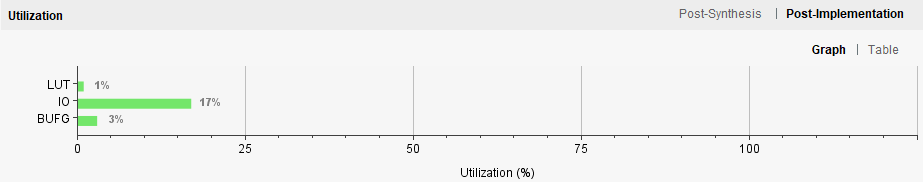
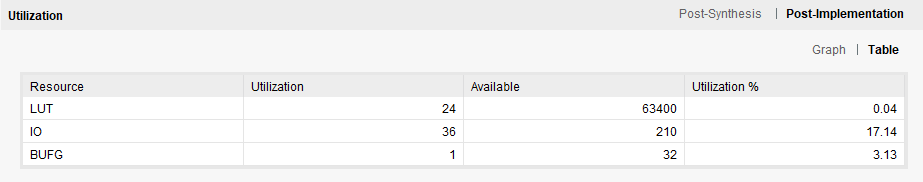


Figure 2: Resource Utilization Graph



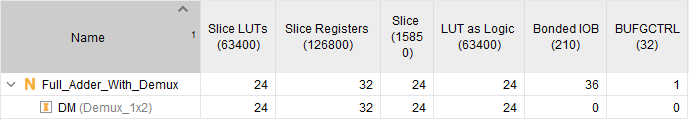


Figure 3: Resource Utilization Table

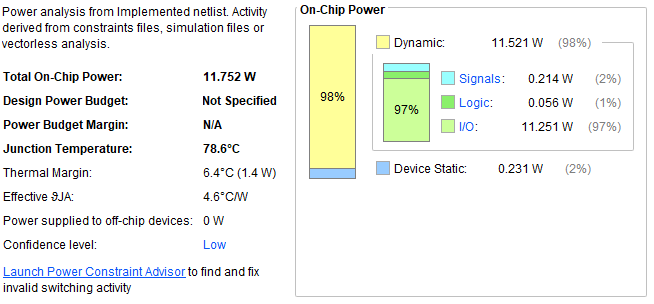


Figure 4: Power Usage