Digital Logic Design & Computer Architecture Lab report

Third assignment

PREPARED FOR

Miss. Talebpour

PREPARED BY

Mohsen Karbalaei Amini, 98242128

Mohammad Mehdi Parchami, 400243084

April 17, 2023,

Serialized output multiplier

Implementation (Part a)

We constructed a generic **(extra point part)** n*n bit multiplier, that outputs the result synchronous to the *clk* signal in a serialized manner. We also embedded a *start* signal which enables us to do multiple multiplications in one test bench.

Here you can see the code:

```
module  shift_add_multiply #(parameter n=32)(multiplier,multiplicand,start,clk,out);

   input [n - 1:0]  multiplier, multiplicand;
   input         start,clk;
   output        out;

   reg [7:0] bit = 0;
   reg                  out;
   reg [n:0]    product;
        reg [n - 1:0] m;

   always @(posedge clk )begin
     if(start || bit >= n + n - 1) begin
        bit = 0;
        product[n:0] = 0;
                  out = 0;
                  m = multiplier;
     end
          else begin
                        if(bit < n) begin
                                if (m[0])
                                        product = product + {1'b0 , multiplicand} ;
     end
          m = m >> 1;
     out = product[0];
     product = product >> 1;
     bit = bit  + 1;
          end
   end
endmodule
```

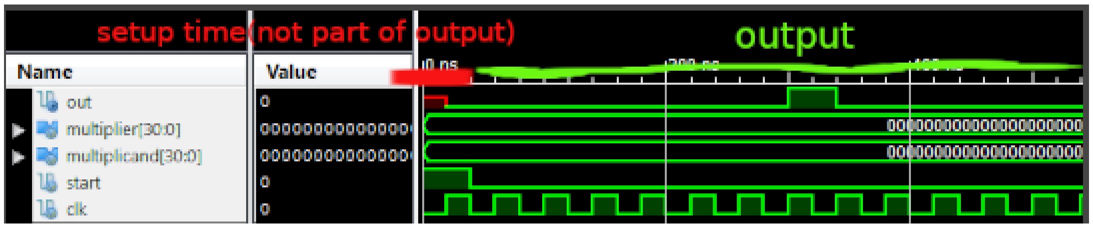
On each positive edge of the *clk* , *out* signal represents the last bit of the product (which won't be changed in the reminding sums), and then shifts the product to the right to make room for the new operations.

Test Bench (part b)

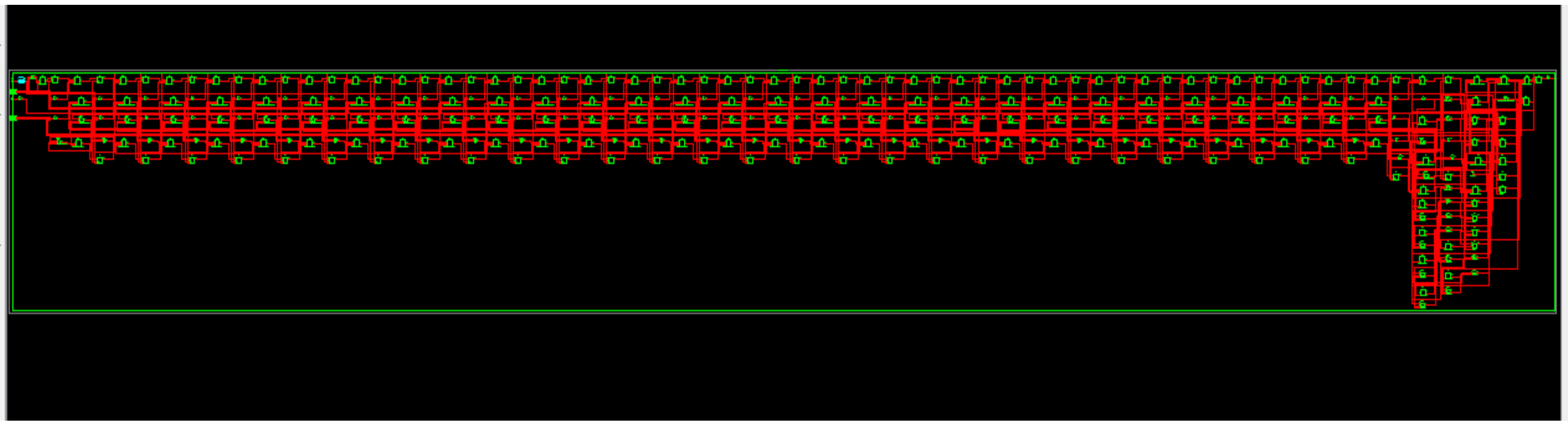
Code and simulation:

```
module shift_add_mul_tb;

	// Inputs
	reg [30:0] multiplier;
	reg [30:0] multiplicand;
	reg start;
	reg clk;

	// Outputs
	wire out;

	// Instantiate the Unit Under Test (UUT)
	shift_add_multiply uut (
		.multiplier(multiplier),
		.multiplicand(multiplicand),
		.start(start),
		.clk(clk),
		.out(out)
	);

	/*always
	begin
		clk <= !clk;
		#50;
	end
	*/
	initial begin
		// Initialize Inputs
		multiplier = 31'd8;
		multiplicand = 31'd8;
		start = 1;
		clk = 0;

		// Wait 100 ns for global reset to finish
		#10;
		#10;
		clk = 1;

		#10;
		#10;
		clk = 0;
				start = 0;
		forever #(20) clk = ~clk;

	end

endmodule
```

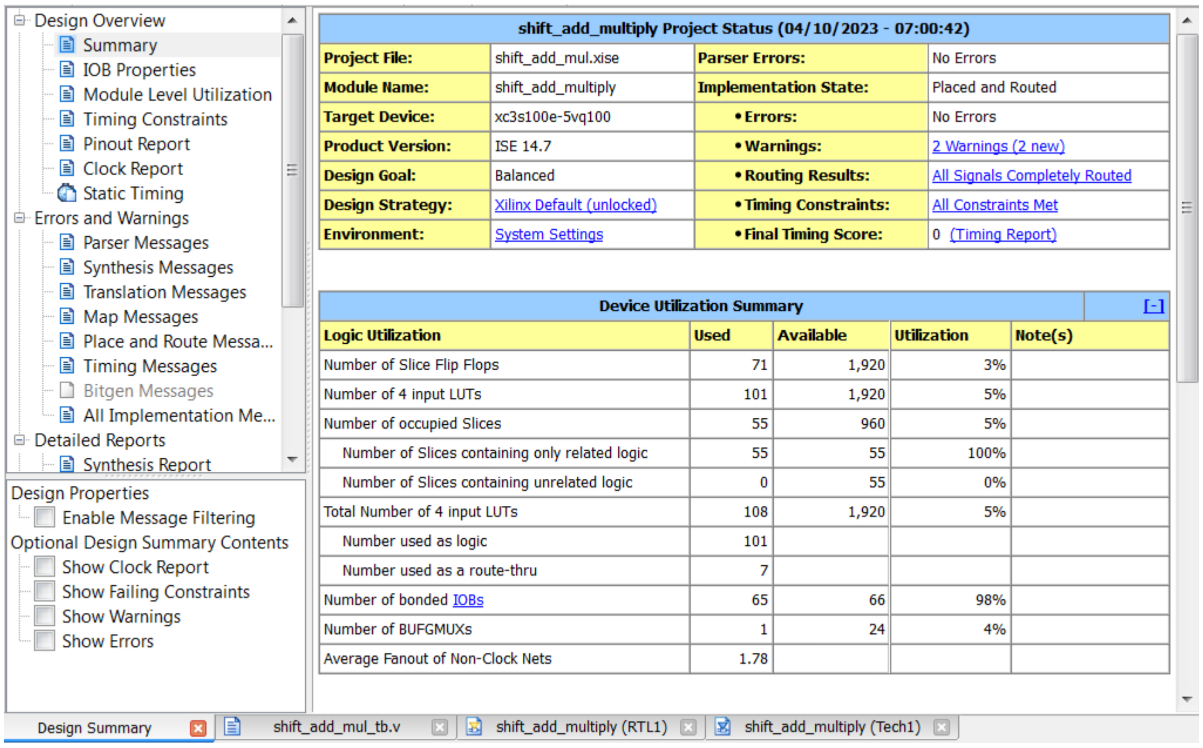
The multiplication of 8*8=64=(7’b100_0000)



Synthesize



Design Summary



shift_add_multiply Project Status (04/10/2023 - 07:00:42)			
Project File:	shift_add_mul.xise	**Parser Errors:**	No Errors
Module Name:	shift_add_multiply	**Implementation State:**	Placed and Routed
Target Device:	xc3s100e-5vq100	**• Errors:**	No Errors
Product Version:	ISE 14.7	**• Warnings:**	2 Warnings (2 new)
Design Goal:	Balanced	**• Routing Results:**	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	**• Timing Constraints:**	All Constraints Met
Environment:	System Settings	**• Final Timing Score:**	0 (Timing Report)

Device Utilization Summary [-]

Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	71	1,920	3%	
Number of 4 input LUTs	101	1,920	5%	
Number of occupied Slices	55	960	5%	
Number of Slices containing only related logic	55	55	100%	
Number of Slices containing unrelated logic	0	55	0%	
Total Number of 4 input LUTs	108	1,920	5%	
Number used as logic	101			
Number used as a route-thru	7			
Number of bonded IOBs	65	66	98%	
Number of BUFGMUXs	1	24	4%	
Average Fanout of Non-Clock Nets	1.78			