ECE 465 Final Project

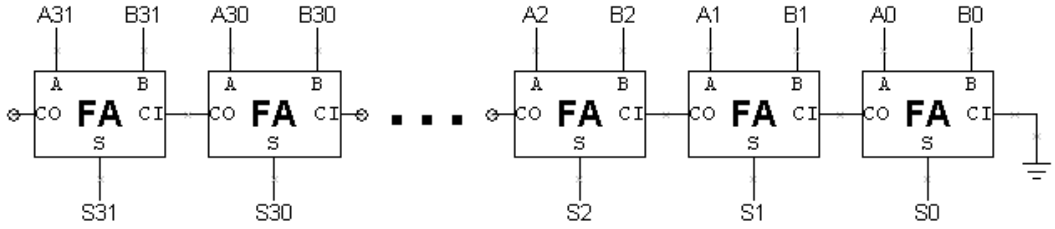
Spring 2019

Name: Karim Eltahawy

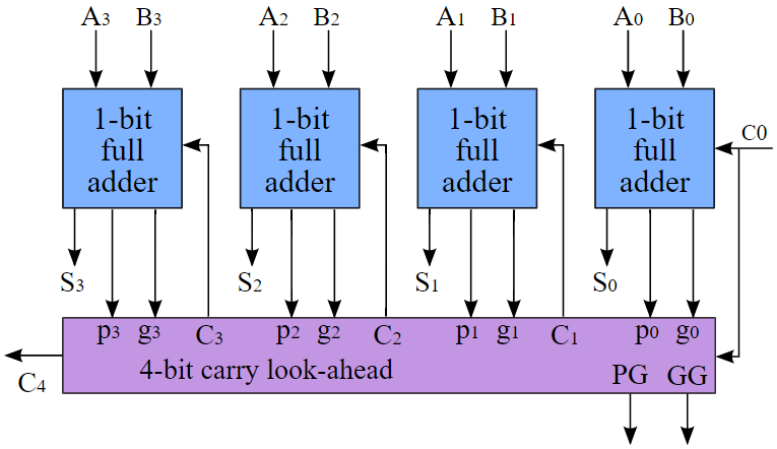
UIN: 651537986

Structures

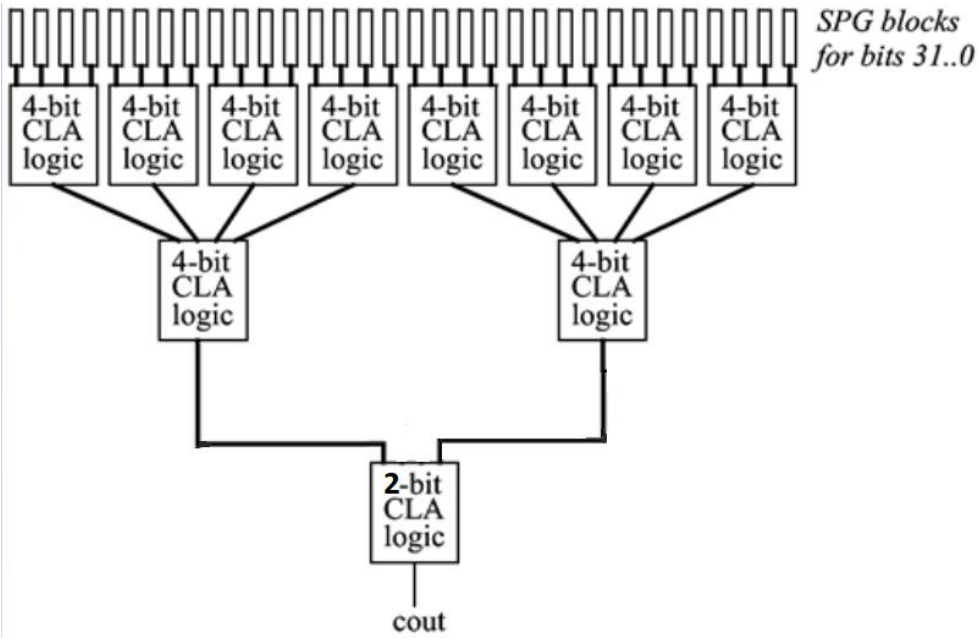
- 32-bit Ripple Carry Adder (RCA)



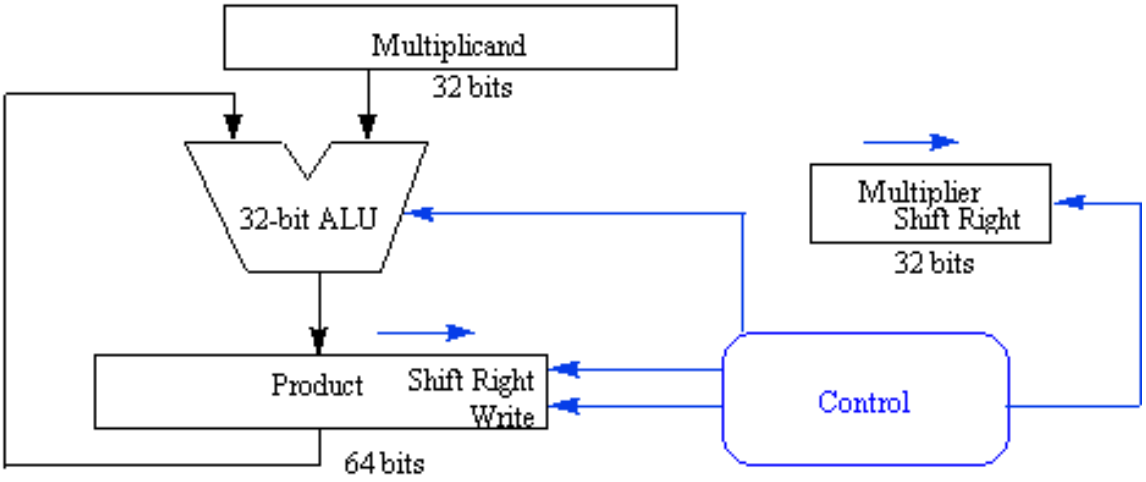
- 4-bit Carry Look Ahead Adder (CLA)



- 32-bit Carry Look Ahead Adder (CLA)

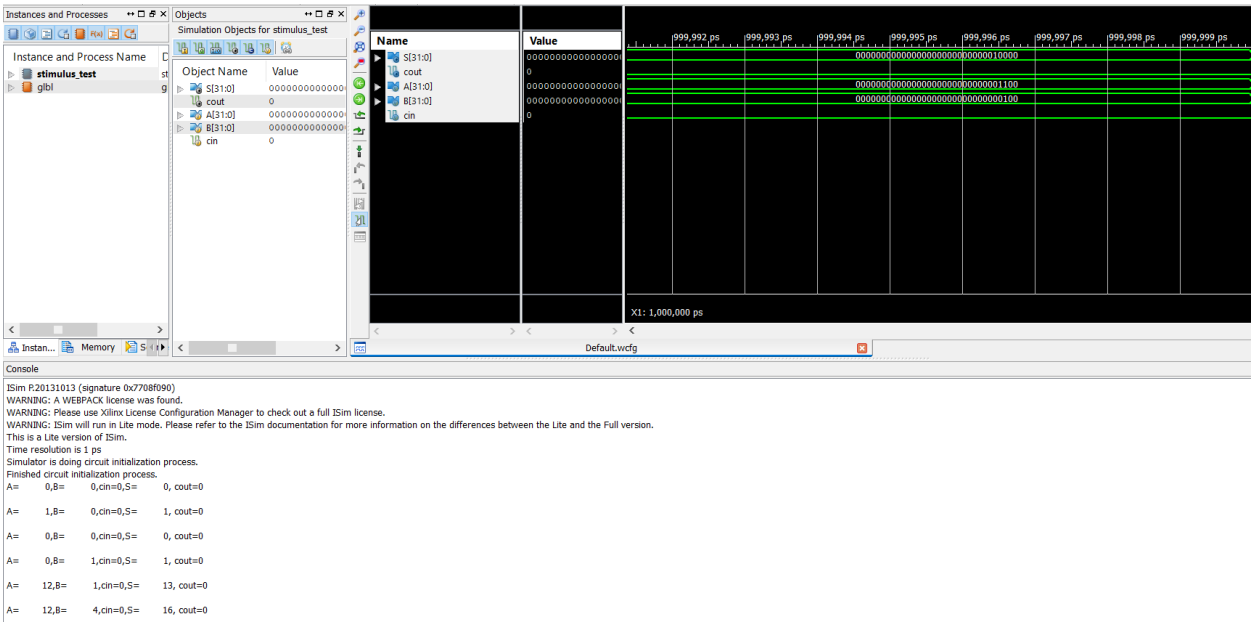


- 32-bit Multiplier

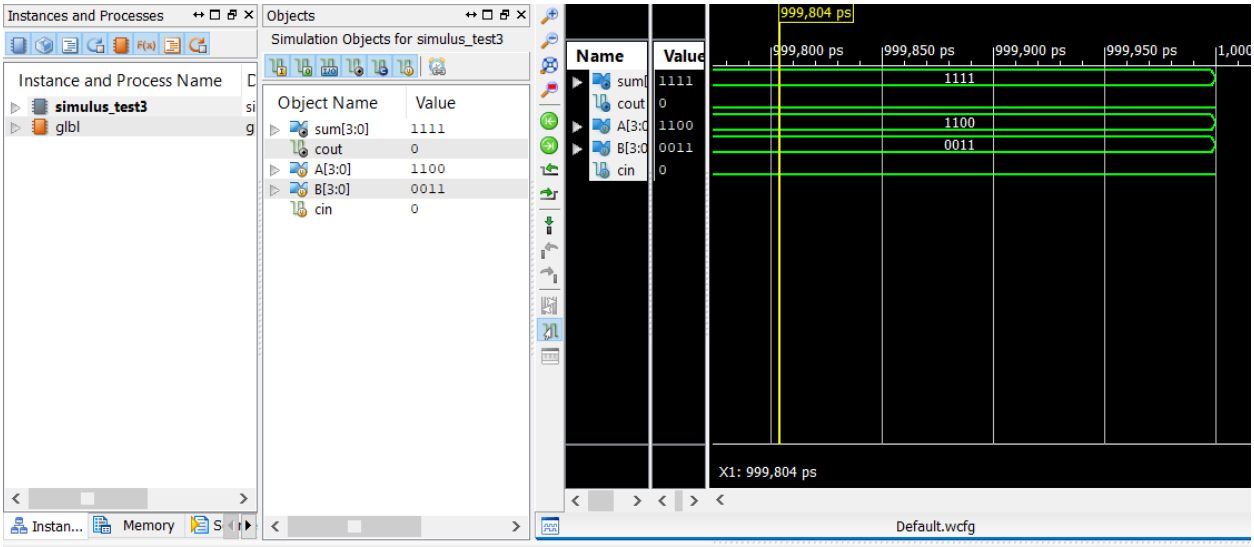


Simulation Results

- 32-bit Ripply Carry Adder (RCA)

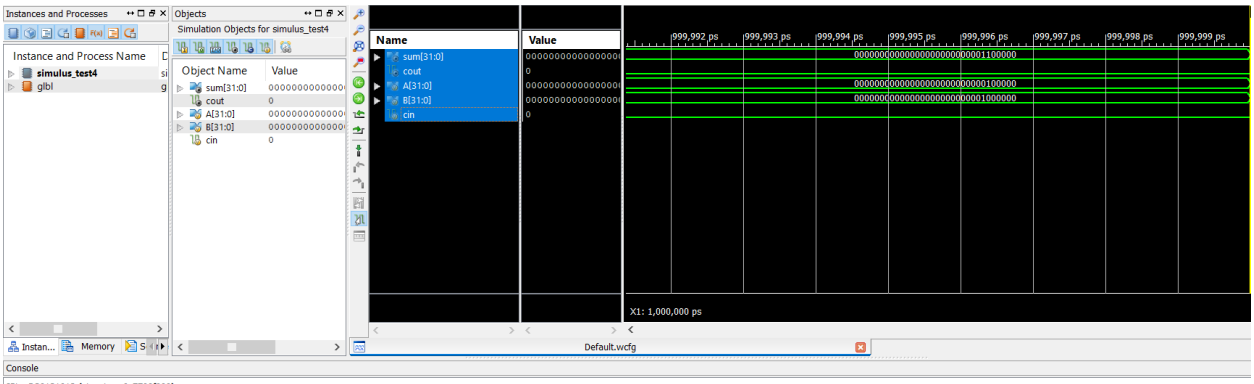


- 4-bit Carry Look Ahead Adder (CLA) (Needed for 32-bit CLA)



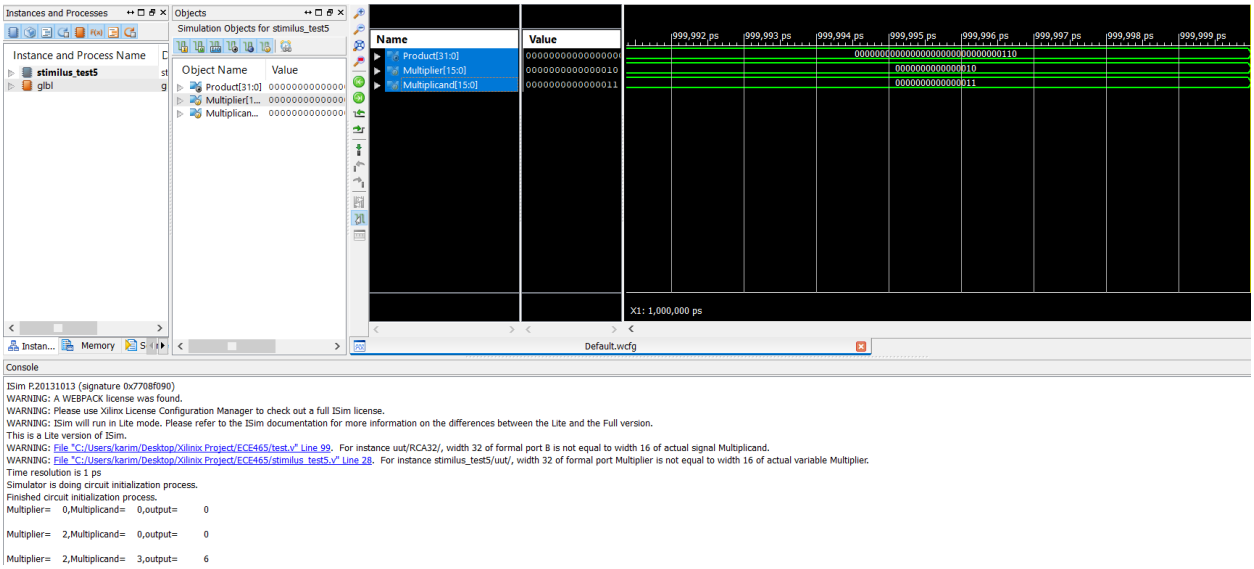

```
Console
ISim P.20131013 (signature 0x7708f090)
WARNING: A WEBPACK license was found.
WARNING: Please use Xilinx License Configuration Manager to check out a full ISim license.
WARNING: ISim will run in Lite mode. Please refer to the ISim documentation for more information on the differences between the Lite and the Full version.
This is a Lite version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
A= 0,B= 0,cin=0,sum= 0, cout=0

A= 1,B= 0,cin=0,sum= 1, cout=0

A= 0,B= 0,cin=0,sum= 0, cout=0

A= 0,B= 1,cin=0,sum= 1, cout=0

A=12,B= 1,cin=0,sum=13, cout=0

A=12,B= 3,cin=0,sum=15, cout=0
```

- 32-bit Carry Look Ahead Adder (CLA)




```
Console
ISim P.20131013 (signature 0x7708f090)
WARNING: A WEBPACK license was found.
WARNING: Please use Xilinx License Configuration Manager to check out a full ISim license.
WARNING: ISim will run in Lite mode. Please refer to the ISim documentation for more information on the differences between the Lite and the Full version.
This is a Lite version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
A=      0,B=      0,cin=0,sum=      0, cout=0

A=      1,B=      0,cin=0,sum=      1, cout=0

A=      0,B=      0,cin=0,sum=      0, cout=0

A=      0,B=      1,cin=0,sum=      1, cout=0

A=     32,B=      1,cin=0,sum=     33, cout=0

A=     32,B=     64,cin=0,sum=     96, cout=0
```

- 32-bit Shift-and-Add Multiplier



Resource consumption and Delay analysis

- 1-bit Full adder
  Max delay = 6.110 ns

- 32-bit Ripply Carry Adder (RCA)
  Max delay = 32.191 ns

- 4-bit Carry Look Ahead Adder (CLA)
  Max delay = 7.455 ns

- 32-bit Carry Look Ahead Adder (CLA)
  Max delay = 24.313 ns

- 32-bit Shift-and-Add Multiplier
  Max delay = 23.191 ns

Total Consumption using XPower tool:

Device	
Family	Spartan6
Part	xc6slx4
Package	csg225
Temp Grade	C-Grade
Process	Typical
Speed Grade	-2

Environment	
Ambient Temp (C)	25.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	0
Heat Sink	None
Custom TSA (C/W)	NA

On-Chip	Power (W)	Used	Available	Utilization (%)
Logic	0.000	32	2400	1
Signals	0.000	113	---	---
IOs	0.000	98	132	74
Leakage	0.014			
Total	0.014			

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	32.2	84.6	25.4

Supply Summary	Total	Dynamic	Quiescent
Source	Voltage	Current (A)	Current (A)	Current (A)
Vccint	1.200	0.004	0.000	0.004
Vccaux	2.500	0.003	0.000	0.003
Vcco25	2.500	0.001	0.000	0.001

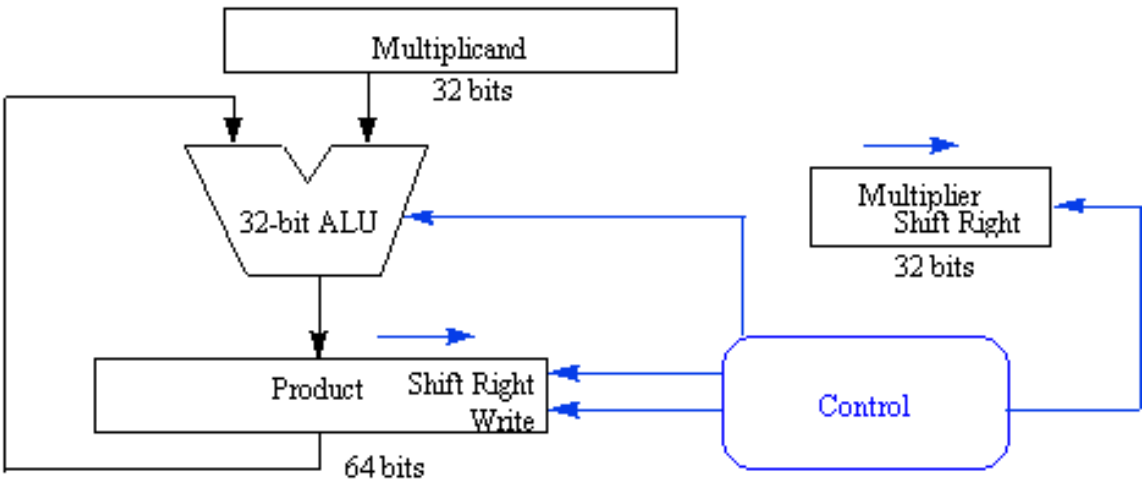
	Total	Dynamic	Quiescent
Supply Power (W)	0.014	0.000	0.014

Performance analysis of your shift-and-Add Multiplier design

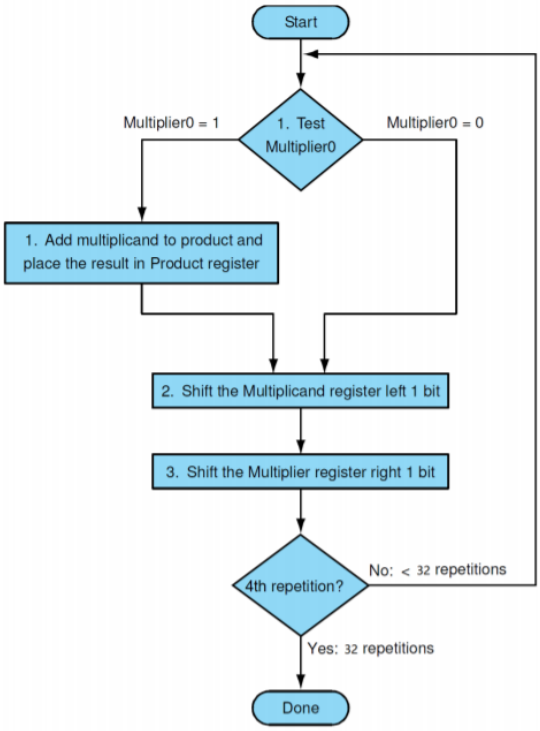
The work done in order for the multiplier to function is in the Control Unit which we focus on in Verilog:

```
module ShiftAddMultiplier32(Multiplier,Multiplicand, Product);

   input [31:0]  Multiplier;
   input [15:0]  Multiplicand;
   output[31:0]  Product;
   reg [31:0]    Product;
   reg           cr;
   integer       i; //used for for loop

   always @(*)
     begin
//initialize
        Product[31:16] = 32'd0;
        Product[15:0] = Multiplier;
        cr = 1'd0;
//add,shift algorithm
        for(i=0; i<16; i=i+1)
          begin
              if(Product[0])
                begin
                      // add using the 32 bit ripple carry adder
                  RippleCarryAdder32 RCA32 (Product[63:32], Multiplicand, 1'b0, Product[63:32], cr);
                     //shift
                  Product[31:0] = {cr,Product[31:1]};
                  cr = 0;
                end
              else
                begin
                    Product[31:0] = {cr,Product[31:1]};
                     cr = 0;
                end
      end    //end of for loop..
  end //end of always block
endmodule
```

Our main design is through this structure:



In Verilog, our main focus is on the Control Unit where we follow this flow and taking the inputs and output into account to be used in the Control Unit:



Using the for loop allows to analyze each bit of the Multiplier, checking if it's 1 or 0. If a 0 is detected then it will use the 32-bit RCA to add the current product with the multiplicand. Then it shifts the multiplicand left and the multiplier right (to get to the next bit for analyzing).

The adder used here is a 32-bit RCA because normally a 32-bit Full Adder is used. But the Ripple Adder function relies on the 1-bit Full Adder module by using 32 FA's and add them together making it a 32-bit Adder.