Matt Winslow
02/17/15
DL EECE6013
Homework 4a

Simple Processor

This project was to implement a simple processor that can add, or, and 2-bit inputs. KEY0 is used store the values in the switches in a register. KEY1 is used to execute the instruction.

The instructions are:
Output A, output not A, output A+B, output A and B.

Inputs used: CLOCK_50, SW, KEY
Outputs used: LED

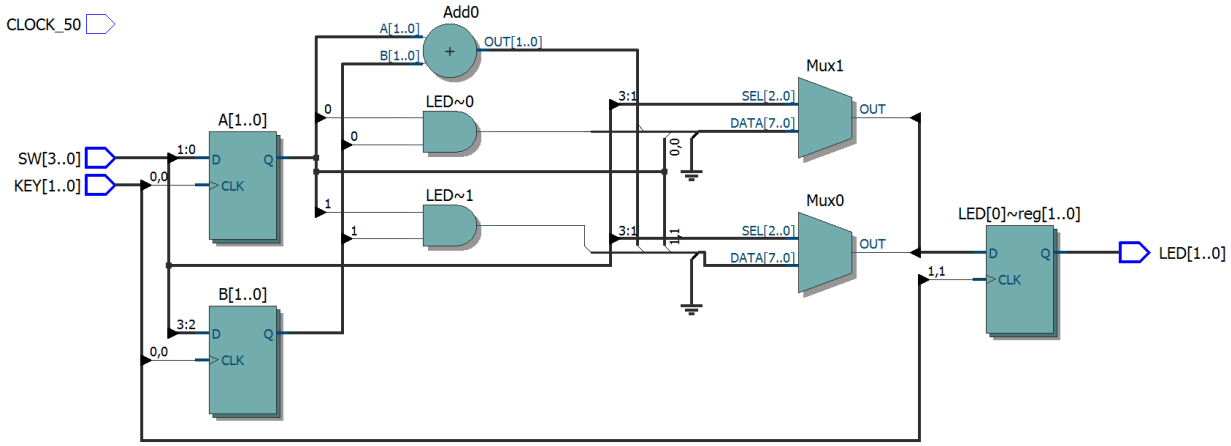
Compilation Report:

Flow Summary	
Flow Status	Successful - Tue Feb 17 20:23:56 2015
Quartus II 64-Bit Version	14.1.0 Build 186 12/03/2014 SJ Web Edition
Revision Name	proce
Top-level Entity Name	proce
Family	Cyclone IV E
Device	EP4CE22F17C6
Timing Models	Final
Total logic elements	9 / 22,320 (< 1 %)
Total combinational functions	5 / 22,320 (< 1 %)
Dedicated logic registers	6 / 22,320 (< 1 %)
Total registers	6
Total pins	9 / 154 (6 %)
Total virtual pins	0
Total memory bits	0 / 608,256 (0 %)
Embedded Multiplier 9-bit elements	0 / 132 (0 %)
Total PLLs	0 / 4 (0 %)

PowerPlay Power Analyzer:

PowerPlay Power Analyzer Summary	
PowerPlay Power Analyzer Status	Successful - Tue Feb 17 21:36:43 2015
Quartus II 64-Bit Version	14.1.0 Build 186 12/03/2014 SJ Web Edition
Revision Name	proce
Top-level Entity Name	proce
Family	Cyclone IV E
Device	EP4CE22F17C6
Power Models	Final
Total Thermal Power Dissipation	98.79 mW
Core Dynamic Thermal Power Dissipation	0.00 mW
Core Static Thermal Power Dissipation	77.41 mW
I/O Thermal Power Dissipation	21.38 mW
Power Estimation Confidence	Low: user provided insufficient toggle rate data

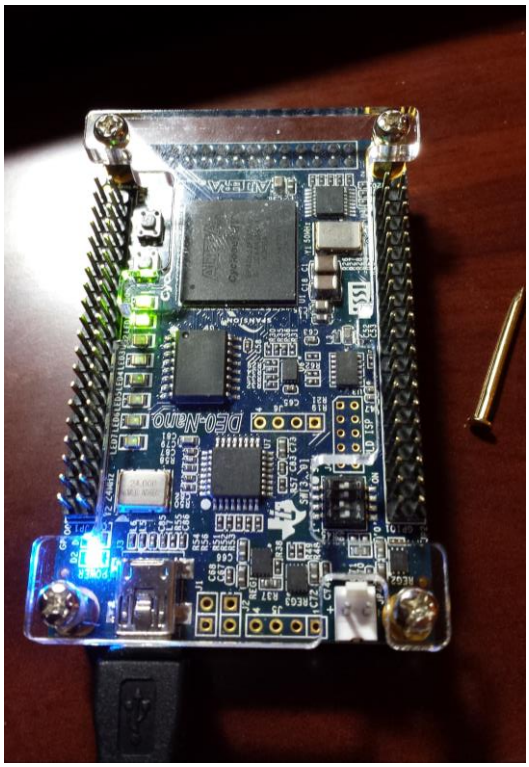
RTL Viewer:



The output is in 2s-compliment (00 = 0, 01 = 1, 10 = -2, 11 = -1) and displays on 2 LEDs.

Here is a quick example of how this program works:

Flip switches 1 and 4.
Press KEY0 to store values, flip switches back.
Flip switch 3 (output A+B).
Press KEY1 to execute.
LED0 and LED1 light up.



Appendix A (proce.v)

```
module proce(
input CLOCK_50,
input [3:0] SW,
input [1:0] KEY,
output reg [1:0] LED
);

reg signed [1:0] A = 0;
reg signed [1:0] B = 0;

// load A and B when KEY[0] is pressed
always @(posedge KEY[0]) begin
A <= SW[1:0];
B <= SW[3:2];
end

// execute the instruction specified by the SW when KEY[1] is pressed
always @(posedge KEY[1]) begin
case (SW)
1: LED <= A;
2: LED <= ~A;
4: LED <= A + B;
8: LED <= A & B;
default: LED <= 2'bxx; // don't care if switch config is invalid
endcase
end

endmodule
```