Four-Function Calculator with 8-bit Output

James Skelly, David Flores

How It Works

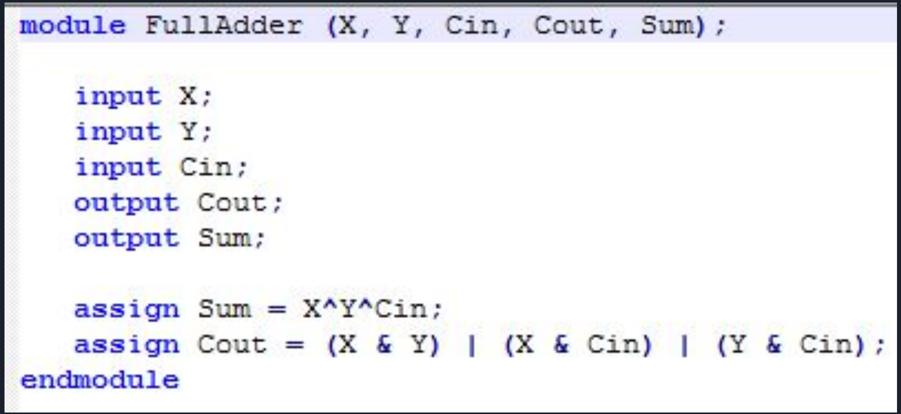
- The calculator takes in two 4-bit binary inputs as operands.
- The output of the calculator, depending on which button is pressed, is either the sum, difference, product, or quotient of the operands.
- The four functions were programmed using Verilog HDL.
- Submodules were used to code 1-bit functions, and these submodules were then used in higher abstraction levels to program the four-bit functions.

Addition

DE2 Operator: Key[0]

The Full Adder submodule performs 1-bit addition. The Addition module (not pictured here) calls the full adder submodule, and implements a 4-bit Ripple Carry Adder by feeding the Cout of the previous stage into the Cin of the next stage for four stages.

Submodule:



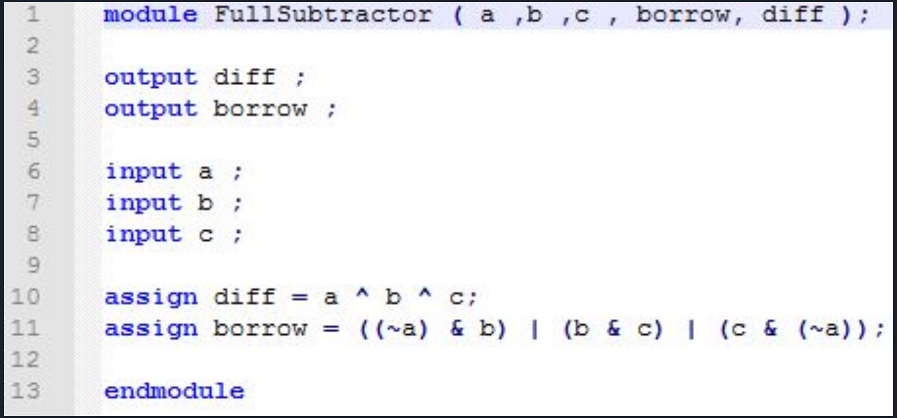
```
module FullAdder (X, Y, Cin, Cout, Sum);

   input X;
   input Y;
   input Cin;
   output Cout;
   output Sum;

   assign Sum = X^Y^Cin;
   assign Cout = (X & Y) | (X & Cin) | (Y & Cin);
endmodule
```

Subtraction

DE2 Operator: Key[1]

The Full Subtractor module shown below performs 1-bit subtraction, where a borrow input is used in the case that the minuend (number being subtracted from) is less than the subtrahend (number being subtracted). The Subtractor module calls the full subtractor submodule in order to perform 4-bit subtraction.

Submodule:



```
module FullSubtractor ( a ,b ,c , borrow, diff );

output diff ;
output borrow ;

input a ;
input b ;
input c ;

assign diff = a ^ b ^ c;
assign borrow = ((~a) & b) | (b & c) | (c & (~a));

endmodule
```

Multiplication

DE2 Operator: Key[2]

The multiplier code was too large to fit in this slide.

The multiplier module was implemented by calling the full adder function used for the addition module. A “multiple add multiplier” was implemented, with the multiplicand being the base number, while multiplier is the number of times that the base number is added to itself. This multiplier was used in the multiplier lab.

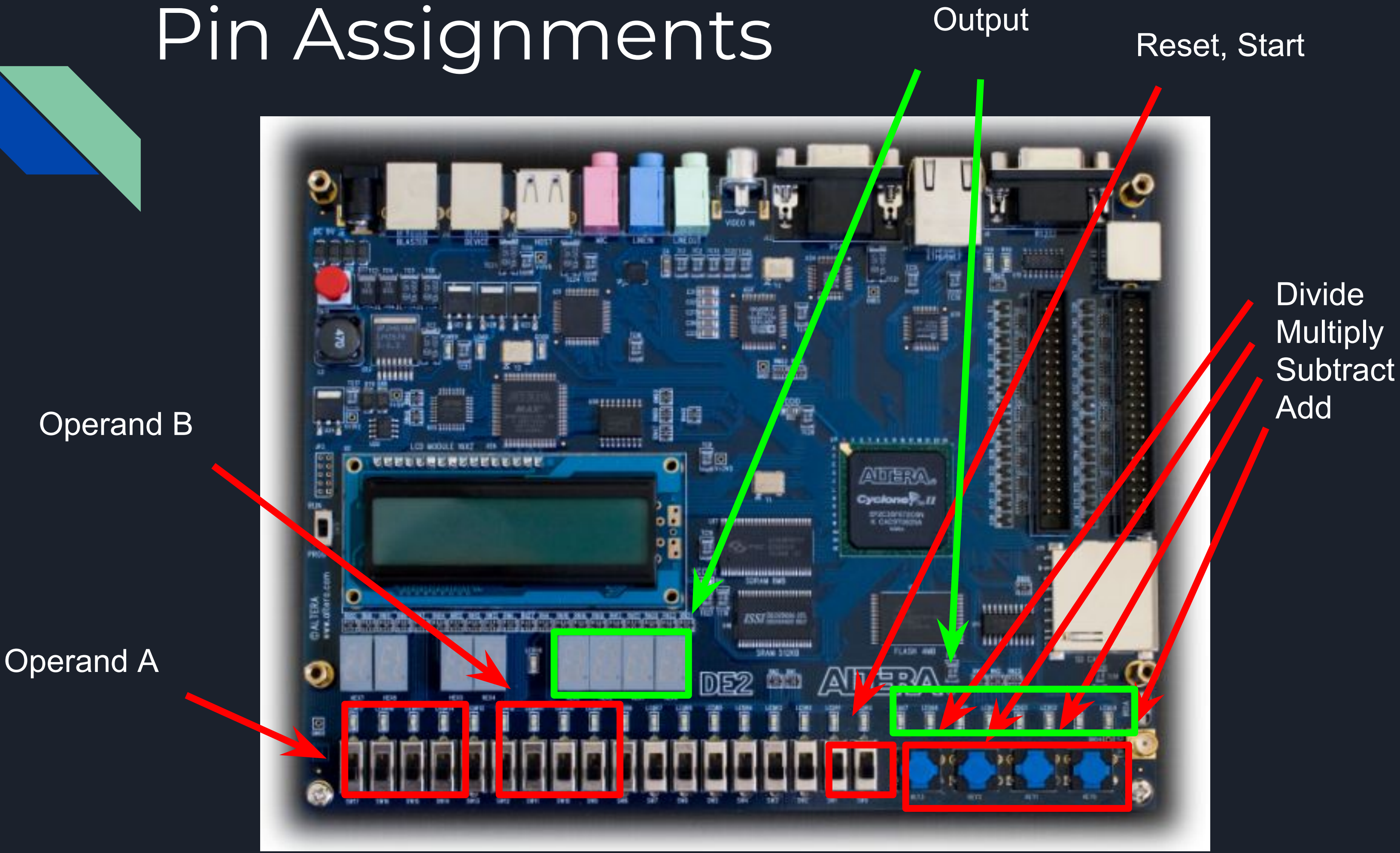
Division

DE2 Operator: Key[3]

The divider code was too long to fit in this slide.

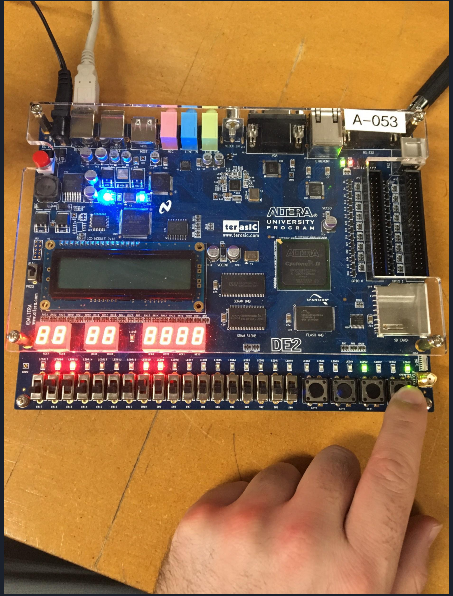
The divider was implemented using repeated subtraction. The divider takes in a 4-bit numerator and a 4-bit denominator, and using if statements, repeatedly subtracts the the denominator from the numerator. The denominator is subtracted from the decrementing numerator until the numerator is less than the denominator, and the result is output. This divider does not account for remainders.

Pin Assignments



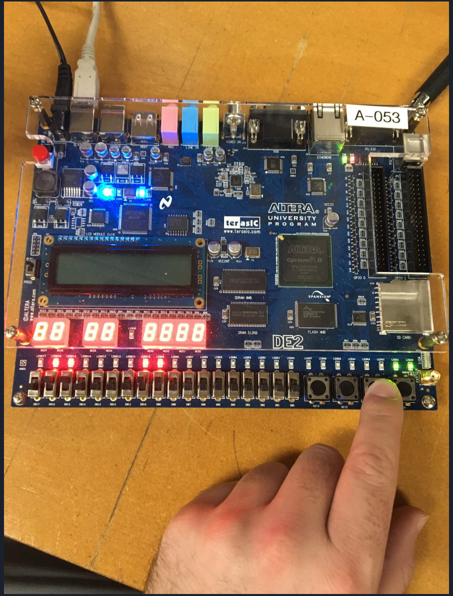
Operands: A = 6, B = 3

Addition



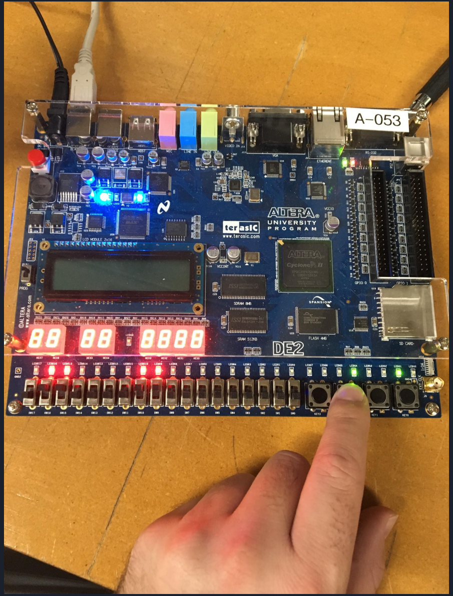
6 + 3 = 9

Subtraction



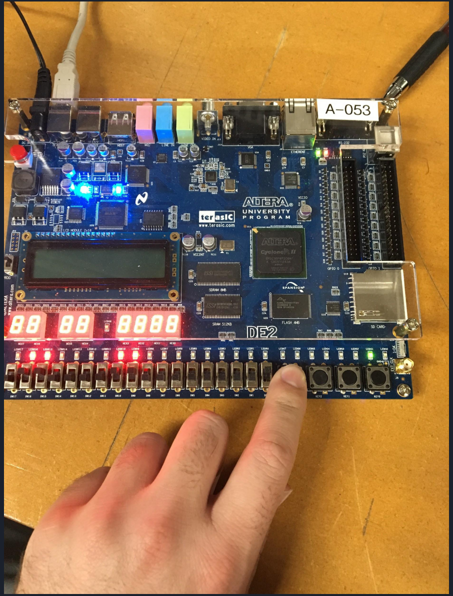
6 - 3 = 3

Multiplication



6 * 3 = 18

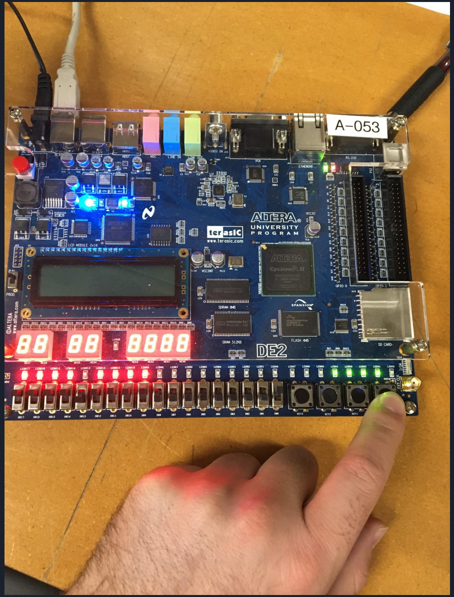
Division



6 / 3 = 2

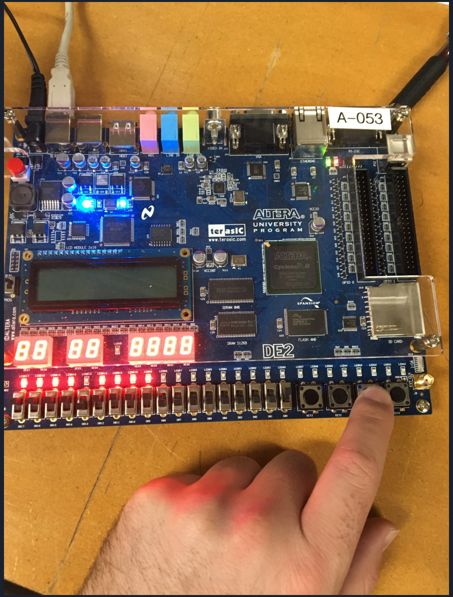
Operands: A = 15, B = 15

Addition



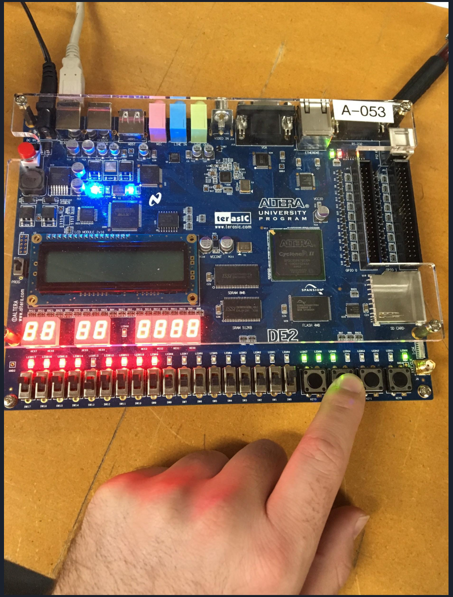
15 + 15 = 30

Subtraction



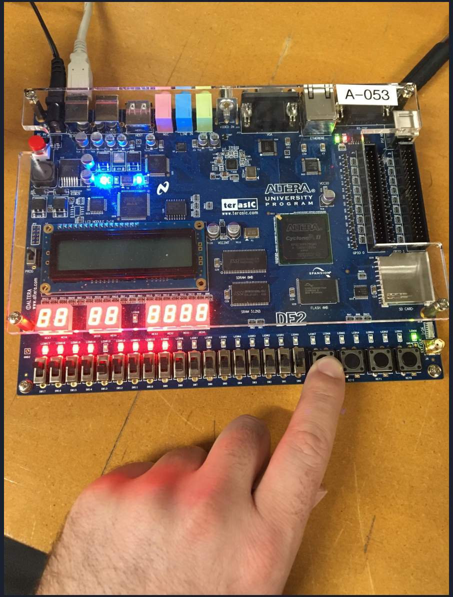
15 - 15 = 0

Multiplication



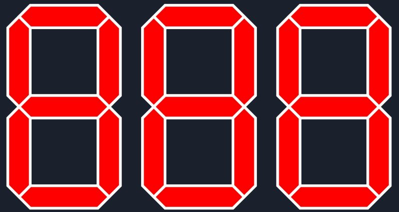
15 * 15 = 225

Division



15 / 15 = 1

Problems & Improvements



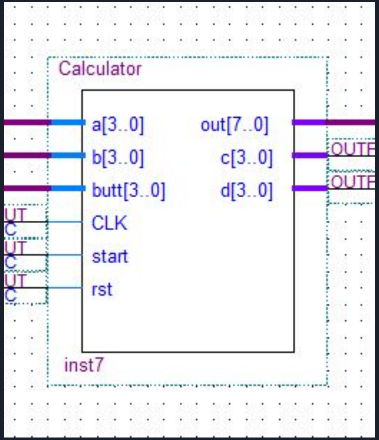
Double Dabble Algorithm

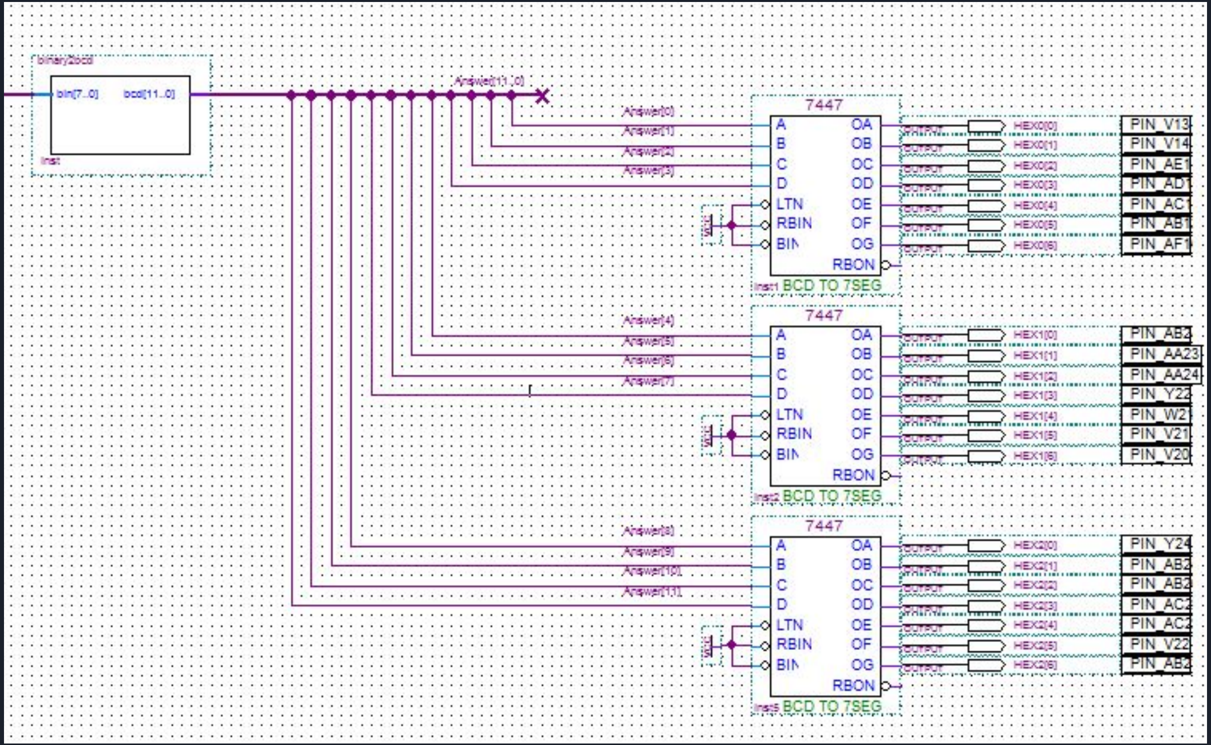
Binary to BCD converters

BCD to 7-segment converters

- Got the modules working separately, could not integrate 7-Segment Display schematic with calculator verilog top level code.
- Would add negative output feature in future design for improved operation and range of outputs.

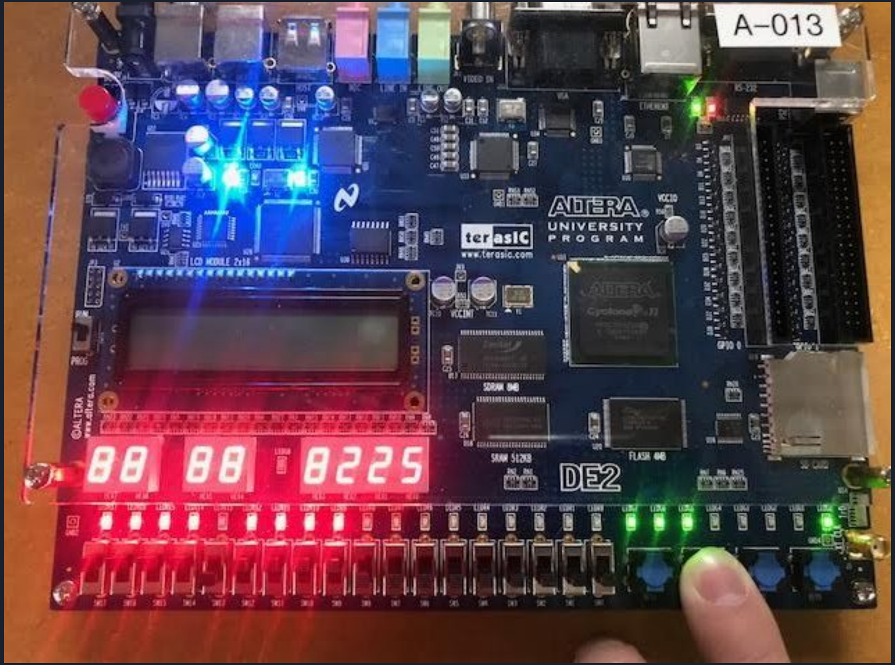
Final Improvements





Operands A = 15, B = 15 with 7-Seg. Output

15 * 15 = 225



Thank you!