Digital Fixed Point Calculator

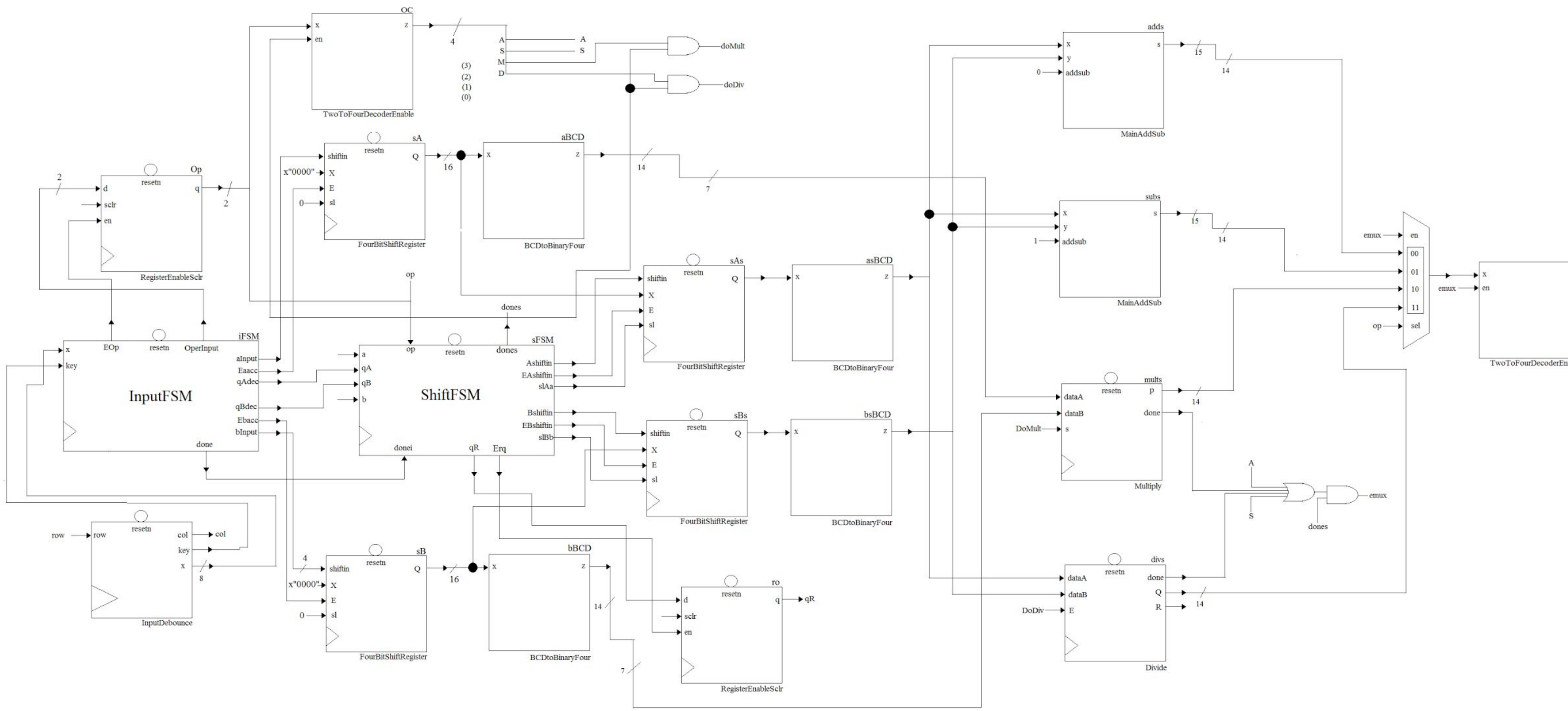
Robert Kozubiak, Muris Zecevic, Cameron Renny

Calculator Overview

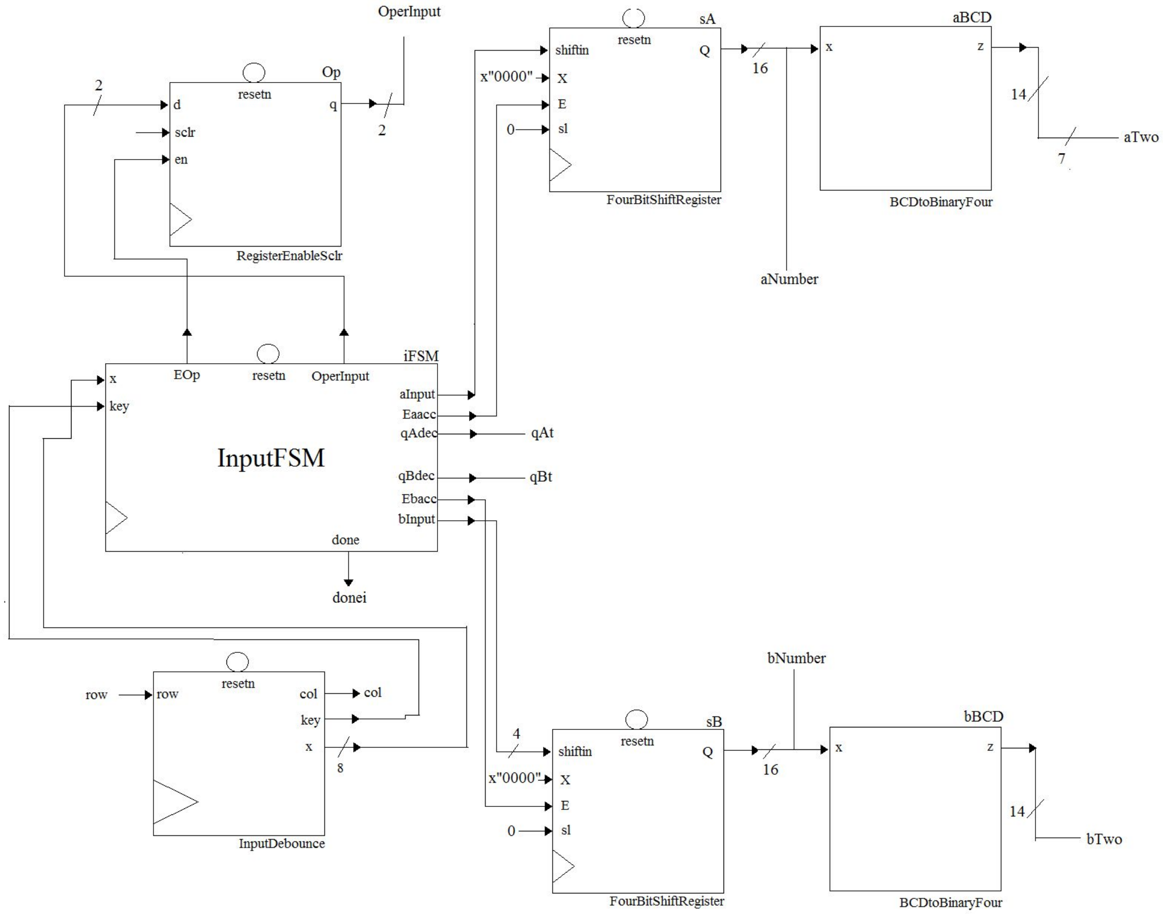
- Add, Subtract, Divide, or Multiply two 2-digit numbers, including decimal numbers
  - i.e) 2.2 + 3.3 = 5.5
- Input
  - 4x4 keypad matrix - 8 wires, 4 inputs (rows), 4 outputs (columns)
    - Turn on one column, if a button is pressed in that column, send a signal
- Output
  - LCD
    - Tells user to enter A and B, then gives result
  - LEDS and Seven Segments for testing

Overview

- Four portions
  - Input
  - Shifting
  - Arithmetic
  - Output
- Overall
  - Over 30 Unique Files - Not Counting Constraints and Testbenches
  - Comparison - Last Semester - Digital Stopwatch - 11 Unique Files
- Top File
  - 25 Components


OC
TwoToFourDecoderEnable
doMult
doDiv
sA
FourBitShiftRegister
aBCD
BCDtoBinaryFour
Op
RegisterEnableSclr
adds
MainAddSub
subs
MainAddSub
sAs
FourBitShiftRegister
asBCD
BCDtoBinaryFour
iFSM
InputFSM
sFSM
ShiftFSM
sBs
FourBitShiftRegister
bsBCD
BCDtoBinaryFour
mults
Multiply
InputDebounce
sB
FourBitShiftRegister
bBCD
BCDtoBinaryFour
ro
RegisterEnableSclr
divs
Divide
emux
TwoToFourDecoderEn

Input



OperInput
Op
resetn
d
q
sclr
en
2
2
RegisterEnableSclr
sA
resetn
shiftin
Q
x"0000"
X
E
0
sl
16
FourBitShiftRegister
aBCD
x
z
14
7
aTwo
BCDtoBinaryFour
aNumber
iFSM
EOp
resetn
OperInput
x
key
aInput
Eaacc
qAdec
qAt
InputFSM
qBdec
qBt
Ebacc
bInput
done
donei
bNumber
resetn
row
row
col
col
key
x
8
InputDebounce
sB
resetn
4
shiftin
Q
x"0000"
X
E
0
sl
16
FourBitShiftRegister
bBCD
x
z
14
bTwo
BCDtoBinaryFour

Input

- First, input Number A
  - Two digits per Number
  - Enter a digit, 4-bits go to shift register
    - ex) Input 9, ‘1001’ to shift register
  - Once you enter two digits, you have to enter an operation
  - A = Add, B = Subtract, C = Multiply, D = Divide
  - If you press #, then you turn on the decimal counter
- After you enter the operation, input Number B
  - Once you enter two digits, you have to press * to move to shifting

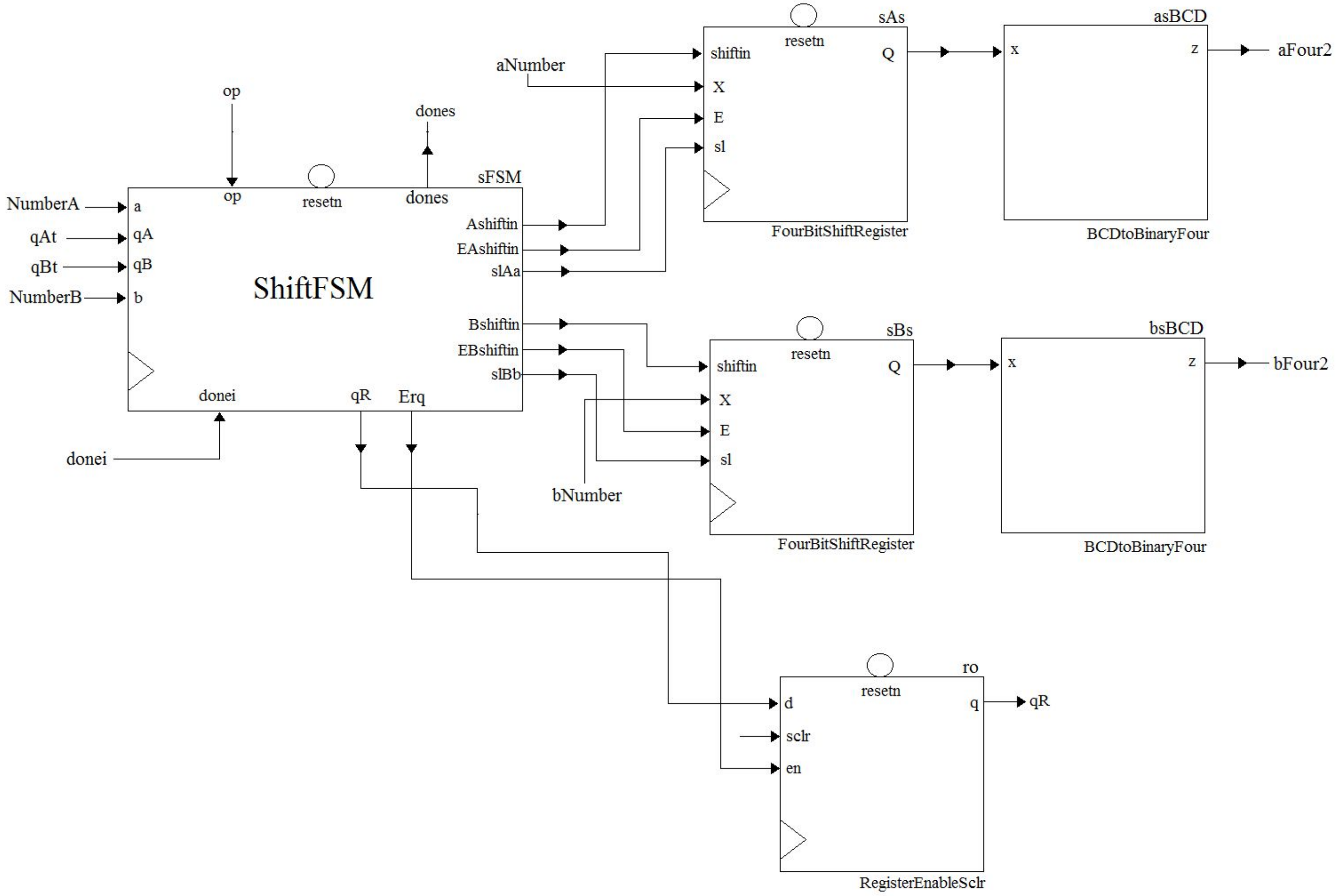
Input

- End result
  - aNumber - 16-bit BCD Signal (i.e. 0099 = x”0099”)
    - aTwo - 7-bit Binary Signal (i.e. 0099 = 1100011)
      - 99 = 7-bits
  - qA - decimal counter
  - bNumber
    - bTwo - 7-bit binary
  - qB
  - Operation
    - A= Add = 00, B = Sub = 01, C = Mult = 10, D = Div = 11

Why Two Digits?

- 7-bit binary signals - Unshifted, 14-bit binary signals - Shifted
- Easy to account for, easy to code, easy to test
- Code should be pretty easy to expand to Four digits, etc
- Multiply and Divide Modules
    - Bigger Numbers = Longer to Complete = More Effort to Complete

Shifting



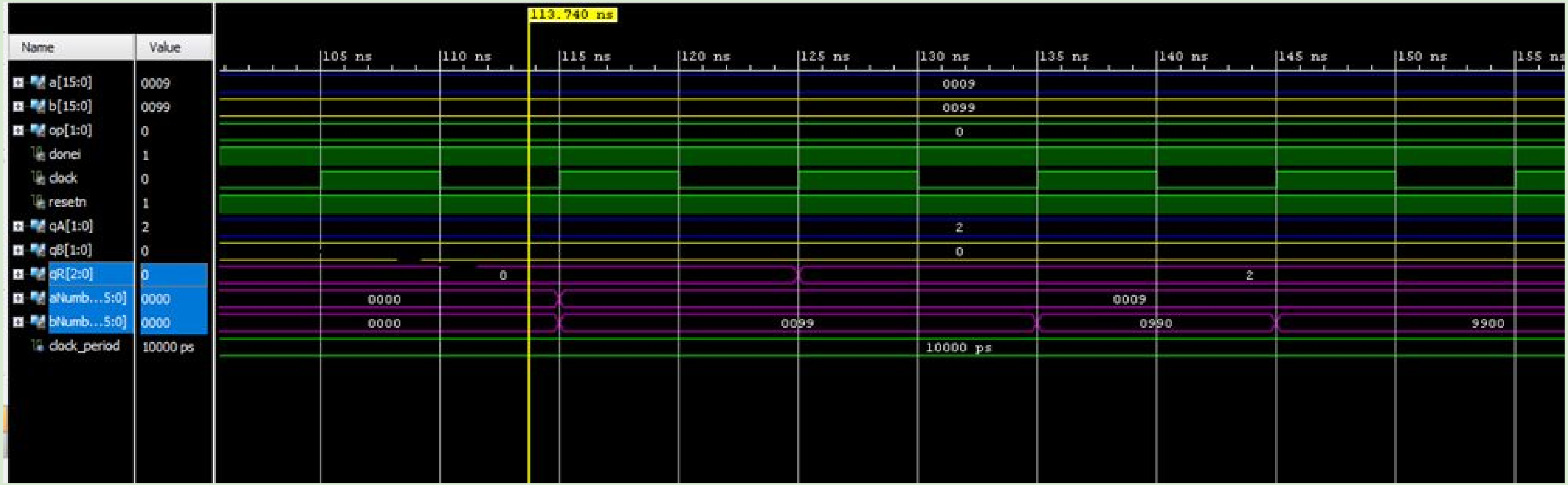
sAs
asBCD
resetn
shiftin
Q
x
z
aFour2
aNumber
X
E
sl
op
dones
FourBitShiftRegister
BCDtoBinaryFour
sFSM
NumberA
a
op
resetn
dones
Ashiftin
qAt
qA
EAshiftin
qBt
qB
slAa
NumberB
b
ShiftFSM
Bshiftin
sBs
bsBCD
EBshiftin
resetn
slBb
shiftin
Q
x
z
bFour2
donei
qR
Erq
X
E
donei
sl
bNumber
FourBitShiftRegister
BCDtoBinaryFour
ro
resetn
d
q
qR
sclr
en
RegisterEnableSclr

Shifting

- Load aNumber, bNumber onto new shift registers (Shifted A, Shifted B)
- Check the operation, and decimal counters qA, qB
    - Multiply
        - Decimal result, qR = qA+qB, No shifting
    - Add/Sub
        - If qA > qB, qR = qA, shift (qA-qB) 0’s into Shifted B
        - If qB > qA, qR = qB, shift (qB-qA) 0’s into Shifted A
        - If qB = qA, qR = qB = qA, no shifting
    - Division
        - Works same as Add/Sub, but qR=0

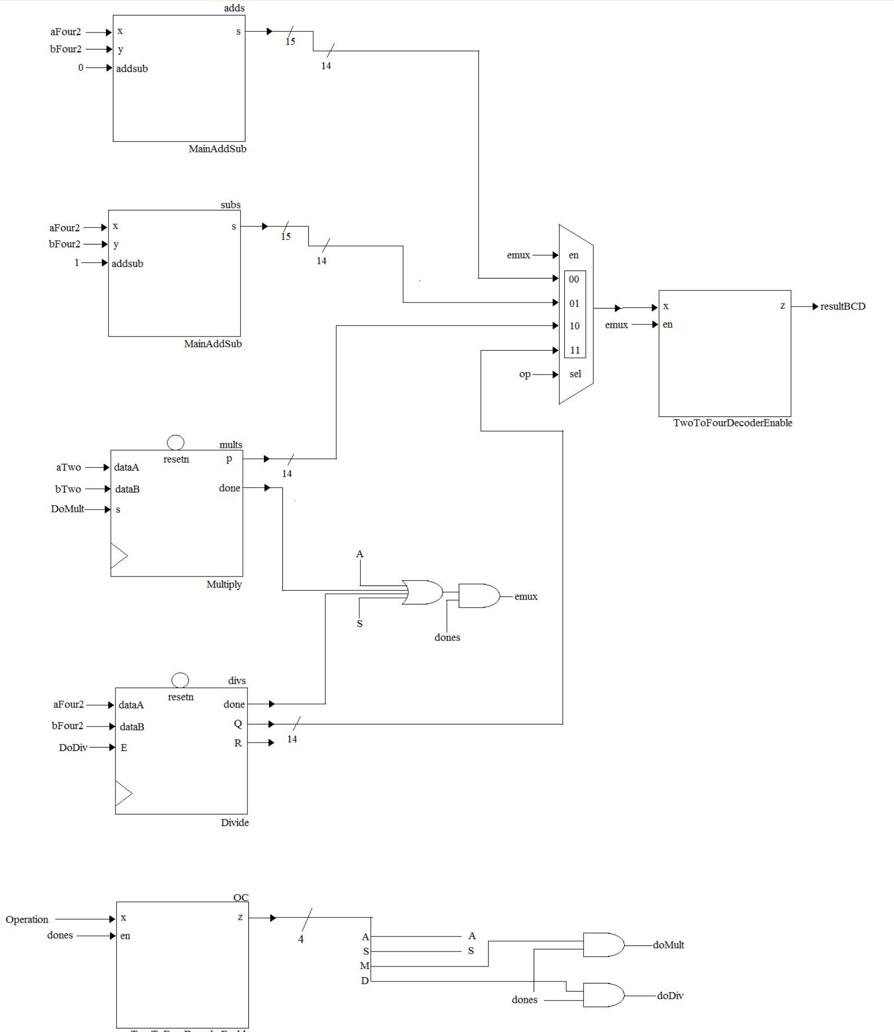
Shifting Example

- A = 3.3   B=0.44   Adding
- Nonshifted A = 0033,   qA = 1
- Nonshifted B = 0044,   qB = 2
- qB>qA
- qR = 2
- Shifted A = 0330
- Shifted B = 0044
- A = 3.30   B = 0.44
- R = 3.74   qR = 2



113.740 ns
Name
Value
105 ns
110 ns
115 ns
120 ns
125 ns
130 ns
135 ns
140 ns
145 ns
150 ns
a[15:0]
0009
b[15:0]
0099
op[1:0]
0
donei
1
clock
0
resetn
1
qA[1:0]
2
qB[1:0]
0
qR[2:0]
0
aNumb...5:0]
0000
bNumb...5:0]
0000
clock_period
10000 ps
0009
0099
0
2
0
0
2
0000
0009
0000
0099
0990
9900
10000 ps

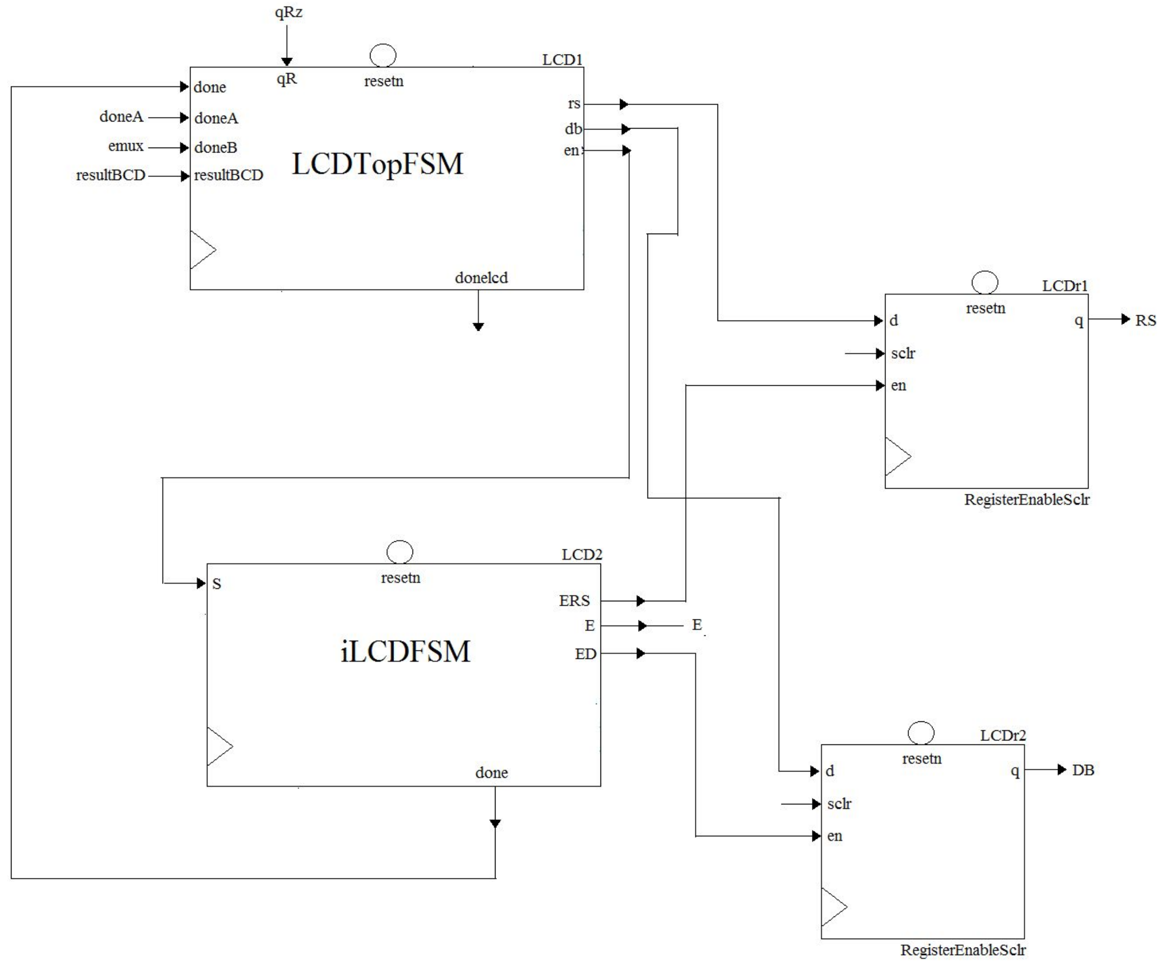
Arithmetic


adds
aFour2
bFour2
0
x
y
addsub
s
15
14
MainAddSub
subs
aFour2
bFour2
1
x
y
addsub
s
15
14
MainAddSub
emux
en
00
01
10
11
op
sel
x
emux
en
z
resultBCD
TwoToFourDecoderEnable
mults
resetn
aTwo
bTwo
DoMult
dataA
dataB
s
p
done
14
Multiply
A
S
emux
dones
divs
resetn
aFour2
bFour2
DoDiv
dataA
dataB
E
done
Q
R
14
Divide
OC
Operation
dones
x
en
z
4
A
S
M
D
A
S
doMult
dones
doDiv

Arithmetic

- Two Signals for A
  - Shifted - 14 bits
  - Nonshifted - 7 bits
- Two Signals for B
  - Shifted - 14 bits
  - Nonshifted - 7 bits
- What signals to use for which operation?
  - Addition - Shifted A + Shifted B - 14 + 14 = 15 -----> 14 (<=9999)
  - Subtraction - Shifted A - Shifted B - 14 + 14 = 15 -----> 14
  - Multiplication - Nonshifted A * Nonshifted B - 7 * 7 = 14 -----> 14
  - Division - Shifted A / Shifted B - 14 / 14 = 14 -----> 14

Output


qRz
LCD1
qR
resetn
done
doneA
doneA
emux
doneB
resultBCD
resultBCD
LCDTopFSM
rs
db
en
donelcd
LCDr1
resetn
d
q
RS
sclr
en
RegisterEnableSclr
LCD2
resetn
S
ERS
E
E
ED
iLCDFSM
done
LCDr2
resetn
d
q
DB
sclr
en
RegisterEnableSclr

Output

- LCD
    - First - Initialize the LCD
        - Set Mode, Clear Display, Turn on Display
    - Then output characters to LCD
        - Input Number A: “ENTER A”
        - Input Number B : “ENTER B”
        - Output Result
            - Uses resultBCD, qR
                - Uses qR to determine when to output the decimal
                - I.e. resultBCD = “9999”   qR= “100”
                    - Output: .9999