Signed Fixed-Point Calculator

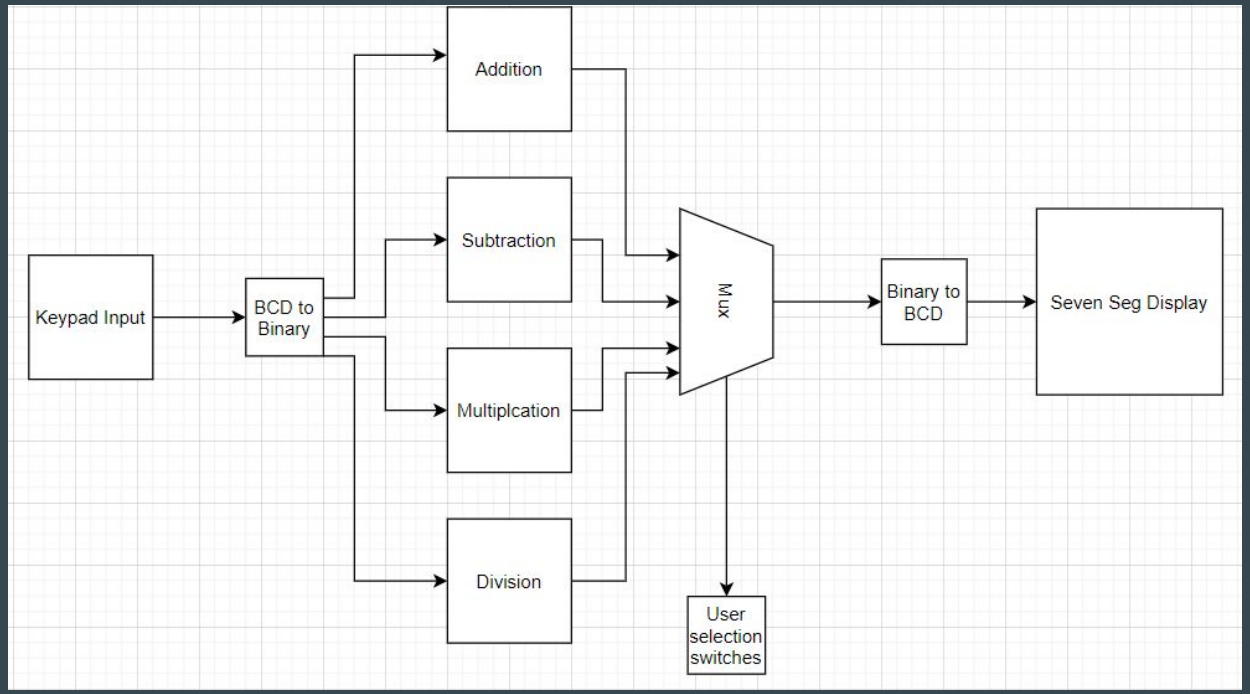
Cory Bledsoe
Rachel Pilarowski
Melvin Pulianthuruthil

Introduction

- Calculators are small complex devices
- Globally used everyday
- Simple arithmetic/complex calculations
- Several components
- Fixed point calculations
  - FX Format: [20 8]

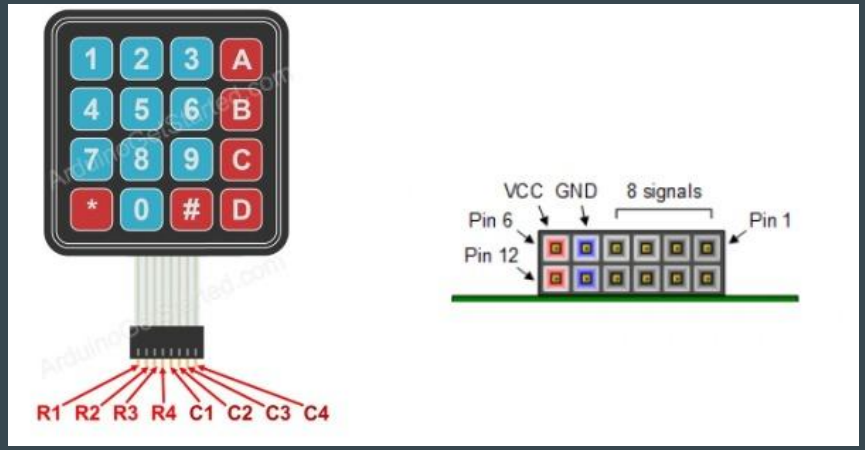
Components

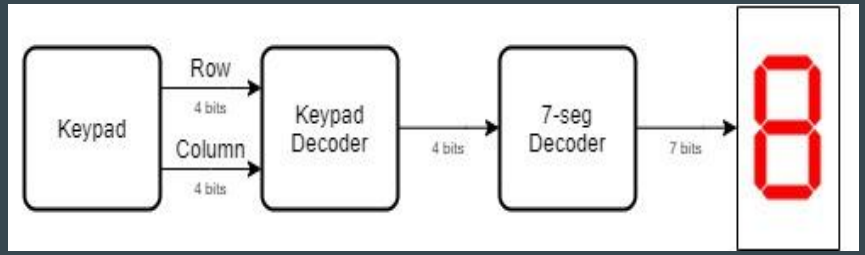
- NEXYS Board
- Keypad
- BCD/Binary Conversions
- 4 circuits (addition, subtraction, multiplication, division)
- Seven Segment Displays
- Switches



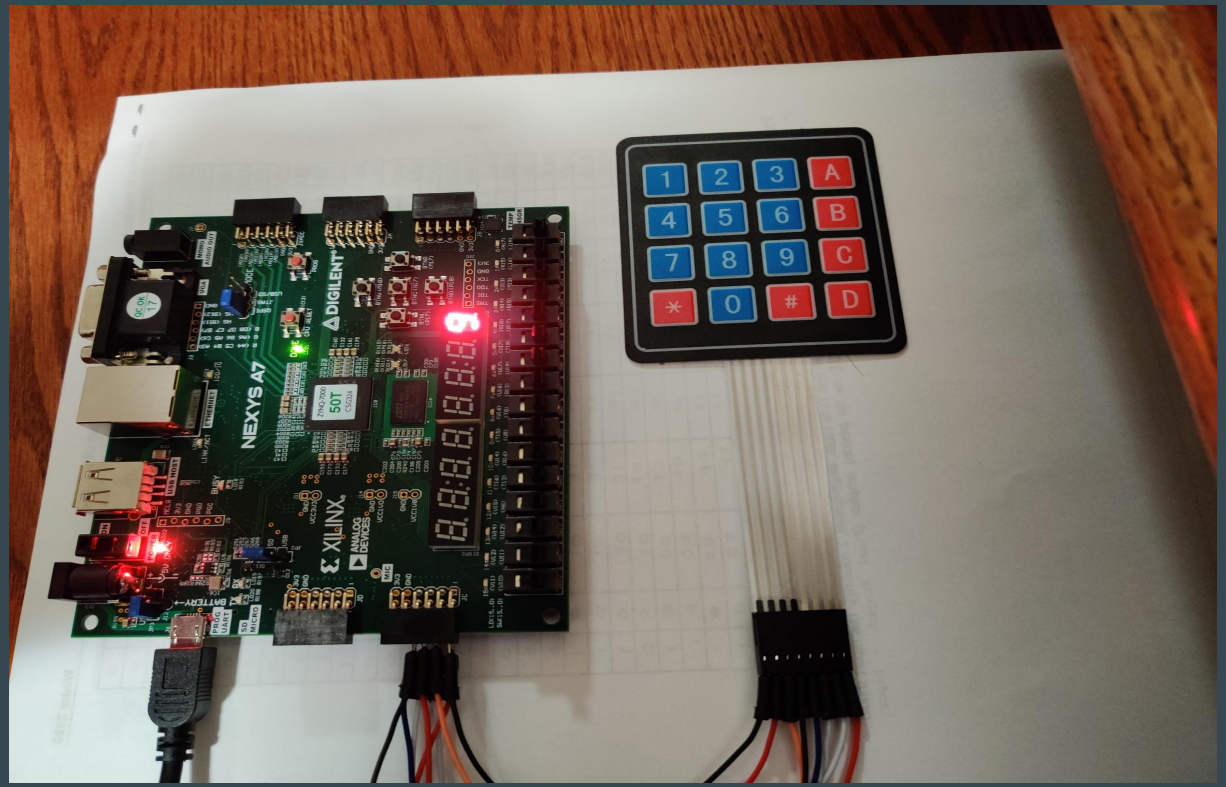
Keypad

- PMOD port JC:
  - Pin 1-4 = C1-C4
  - Pin 7-10 = R1-R4
- Keep one column bit low and rest high. Row bits are kept high. Then scan each row for a low bit within that column.
- One column bit and one row bit low = a button press.
  - Value displayed to the 7-segment
- If no row bits are low, move to the next column. Repeat process



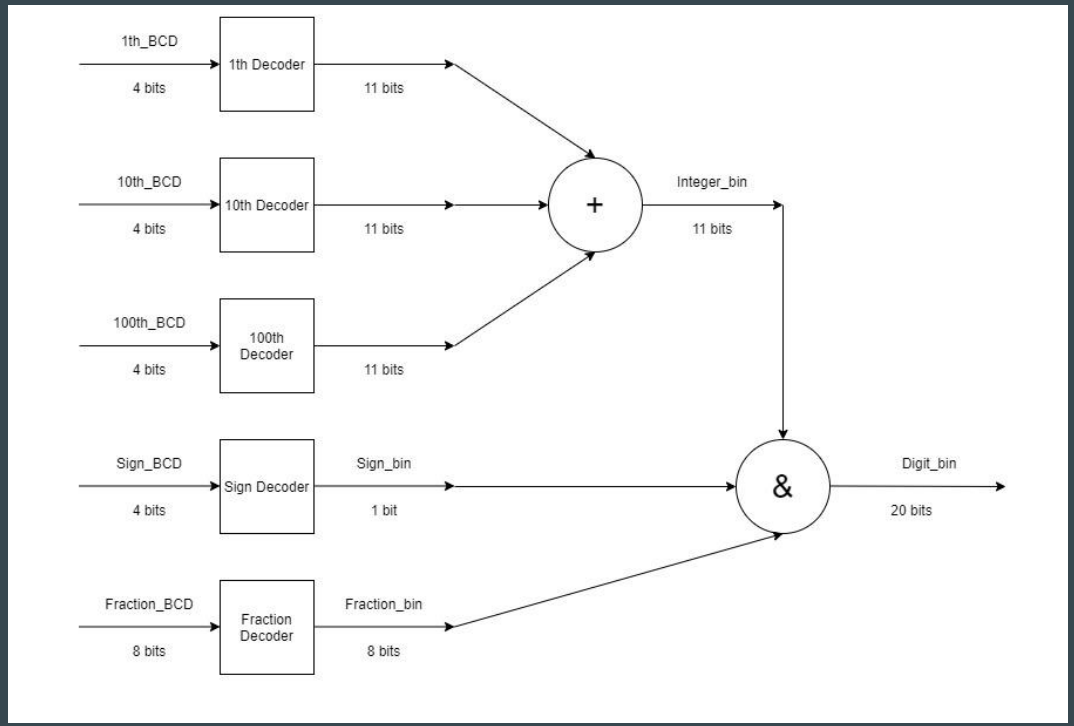


Keypad



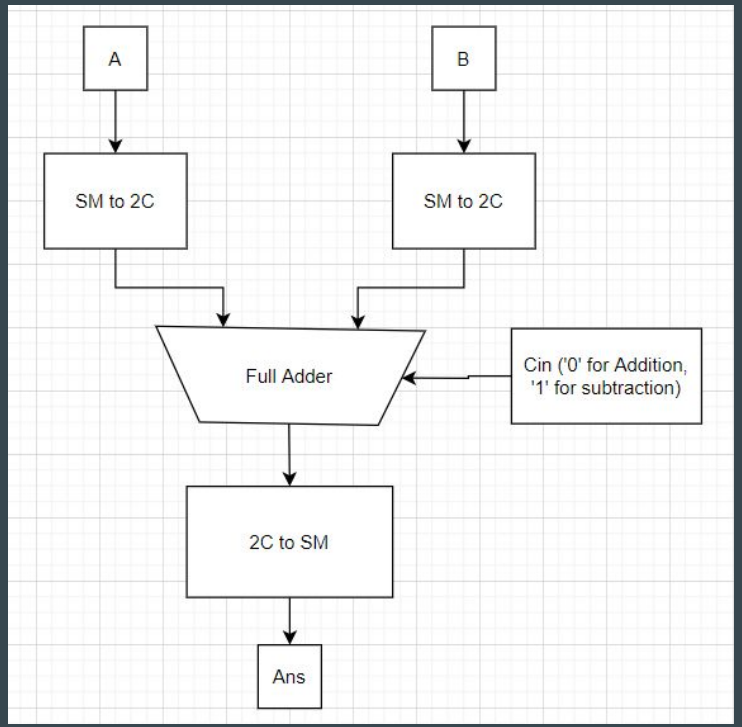
BCD to Binary

- Sign Decoder
  - Press B => ‘1’
  - Press A => ‘0’
- Integer Decoder
  - 1^{th} place digit BCD => 11 bits
  - 10^{th} place digit BCD => 11 bits
  - 100^{th} place digit BCD => 11 bits
  - Add the 3 results
- Fraction Decoder
  - Like LUTs
  - Fraction_BCD(7 downto 4) chooses which Mux
  - Then select the value from the list
- Result = Sign & Integer & Fraction



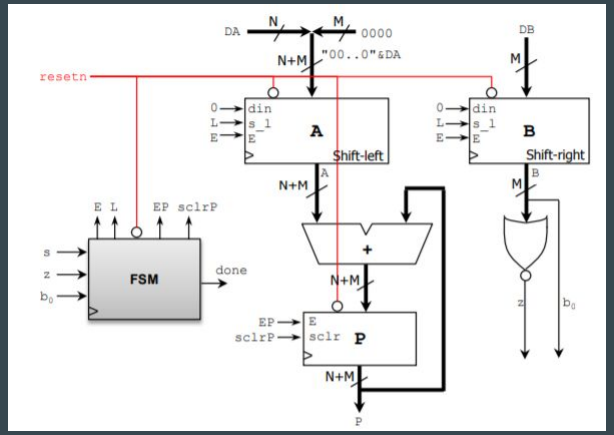
Addition/Subtraction

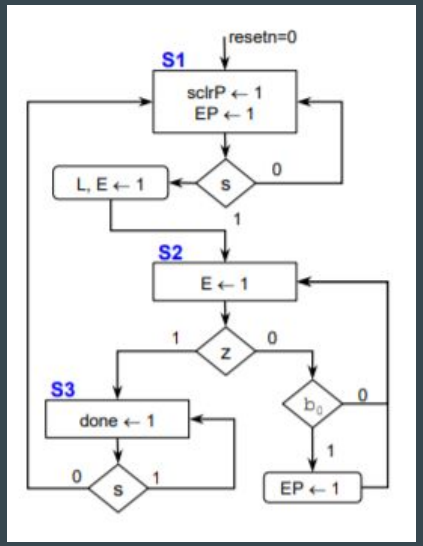
- Uses simple full adder
  - Cin is selected to be ‘0’ for addition and ‘1’ for subtraction
- Full adder expects inputs in 2’s Complement
- SM to 2C
  - If sign is ‘0’, output does not change
  - If sign is ‘1’, append a ‘0’ to the MSB and compute 2C
- 2C to SM
  - If MSB is ‘0’, output does not change
  - If MSB is ‘1’, convert and append ‘1’ to MSB



Multiplication

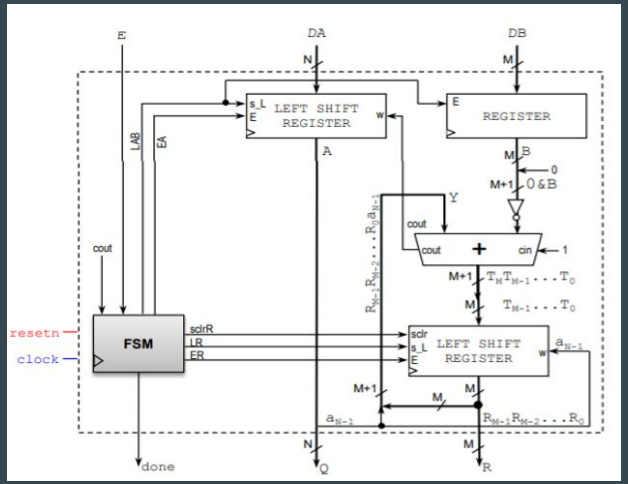
- Inputs are in Sign and Magnitude
  - Just need an XOR gate for sign
- Append 0's to MSB of second input
- Use typical Iterative Multiplier
  - If b0 = '1', P = P + A
  - Shift A left
  - Shift B right
- Answer has FX Format [38 16]
  - Truncate bottom 8 bits for 8 decimal places and top bits for 11 integer bits

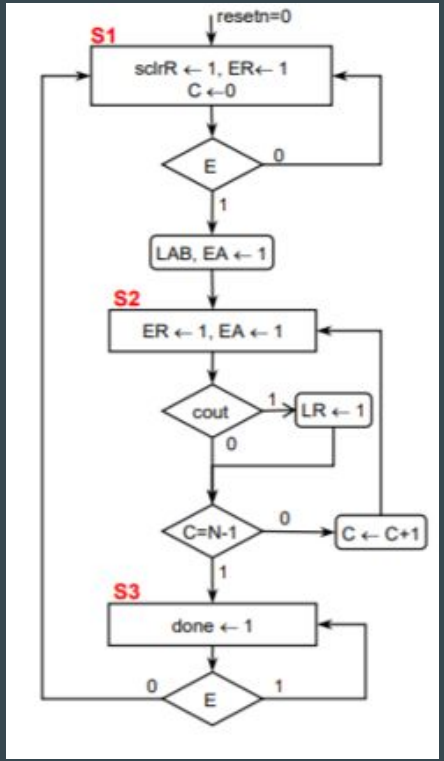




Division

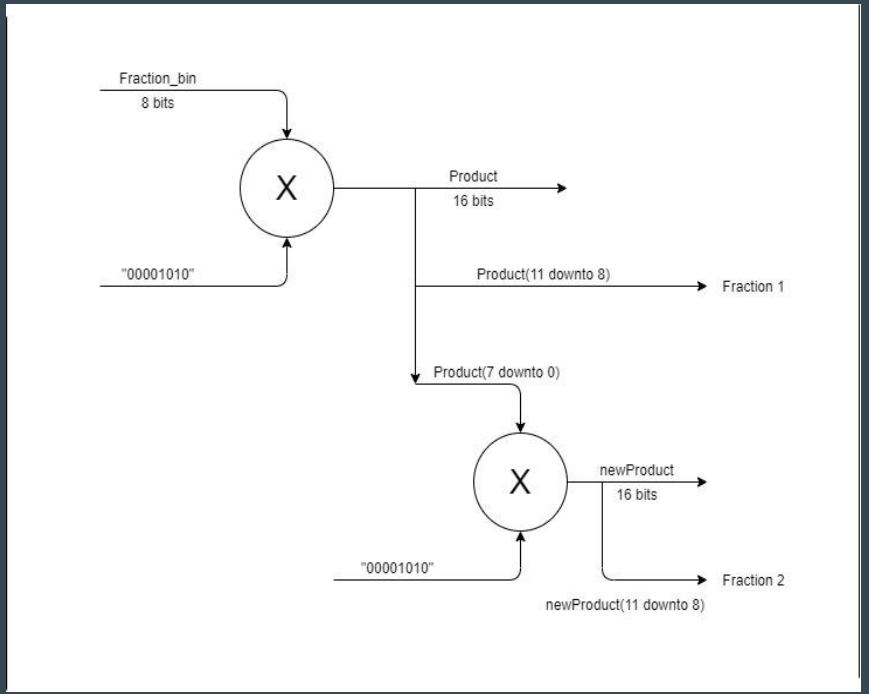
- Inputs are in Sign and Magnitude
  - Just need an XOR gate for sign
- Append 8 bits to LSB
  - For 8 bit precision
- Use normal Iterative Divider
  - Bits of A are compared to the divisor
  - If higher or equal to B, then B is subtracted
- Last 8 bits of Q are truncated





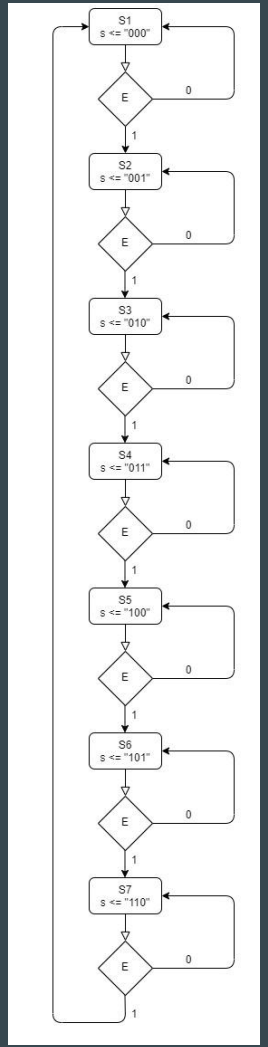
Binary to BCD

- 20 bit Result => 6 BCDs
- Result(19) = sign
- Result(18 downto 8)
  - Double Dabble
    - Left Shift
    - Add 3 if BCD > 4
  - Get 3 BCD values
- Result(7 downto 0)
  - Multiply by 10
  - Product(11 downto 8) = Fraction 1 BCD
  - Multiply 10 by Product(7 downto 0)
  - newProduct(11 downto 8) = Fraction 2 BCD



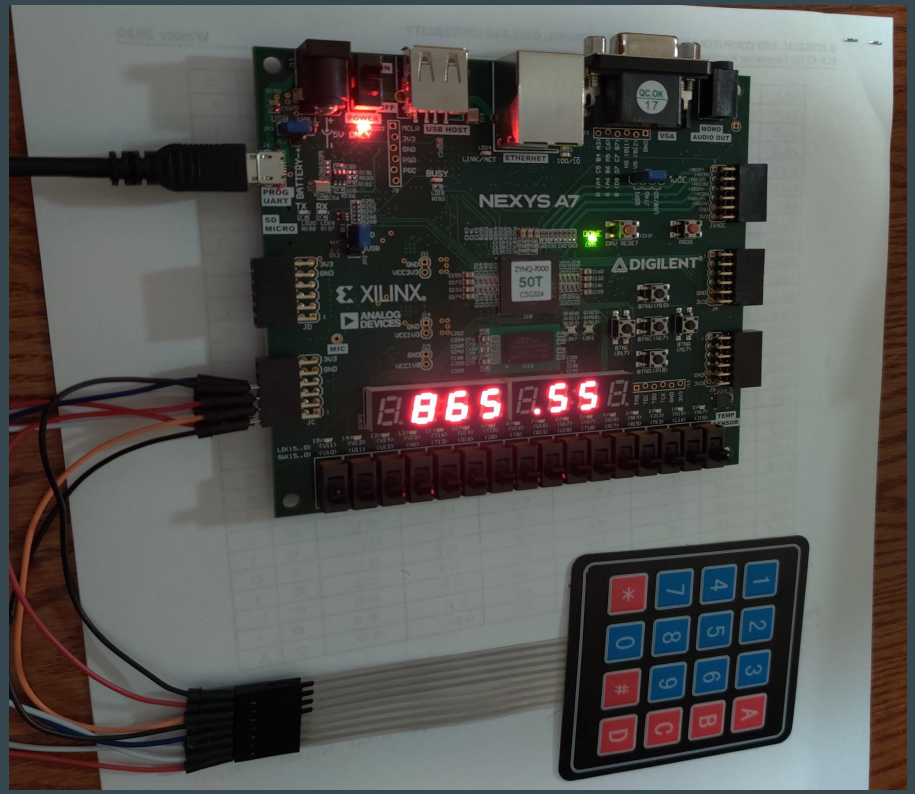
Display Result

- Pulse generator outputs ‘1’ every 1ms
- FSM: 7 states to enable each 7-segments
  - E = ‘1’ every 1ms
  - s <= “000” enables first 7-segment
  - Last 7-segment not used
- 1st 7-segment for sign
  - Positive Number => no display
  - Negative Number => display “-”
- 2nd - 4th 7-segment for integer
- 5th 7-segment for decimal point
- 6th and 7th for fraction



Final Implementation

- Turn on switches to grab Keypad value
  - Switches 1 for sign of Number A
  - Switches 2 for 100th digit of Number A
  - Switches 3 for 10th digit of Number A
  - Switches 4 for 1th digit of Number A
  - Switches 5 for 10th fraction of Number A
  - Switches 6 for 100th fraction of Number A
  - Switches 7-12 for Number B
- BCD to Binary
- Choose arithmetic using Switch 15 and 16
  - “00” for add
  - “01” for sub
  - “10” for multiply
  - “11” for divide
- Arithmetic
- Binary to BCD
- Display Result



Improvements

- Get digits without using switches
- Display Number A and B after user inputs
- Use LCD to display
  - Bigger range for the fixed point format
- A faster and efficient way to convert BCD fractions
  - Not a good method if it was a bigger fraction (Ex: 0.9999)

Thank You