

Fixed point division.

Firstly I have calculated division using Newton Raphson's method for 15/23 in python using the formula as discussed in class. The guess value is calculated using the formula I have derived as part of the homework (stated below). Next, I converted 15 and 23, initial guess values to fixed point format Q(6,16). Then, I used that binary result as input in the Verilog, calculated the division, and compared the output with the python.

Next, I have done the same but for an array of values. I took 9 values in array. I.e total 9 divisions to be computed. Like in the 2nd assignment, I first defined the numerator and denominator, calculated guess values, and calculated its division using Newton raphson method and stored values for future reference. I have converted the inputs, numerator, denominator, guess value arrays to fixed format Q(6, 10) and stored the results in the text file. Later I loaded the text file values into verilog memory and used the division module defined for the single division exercise and computed the division of numerator and denominator arrays using the initial guess array. Later I stored the division outputs in another text file and observed the errors between precomputed values in python and verilog outputs.

$$f(x+h) = f(x) + f'(x)h + \frac{f''(x)h^2}{2!} + \frac{f'''(x)h^3}{3!} + \dots$$

$$x+h = x_{i+1} \quad h = (x_{i+1} - x_i)$$

$$x = x_i$$

$$f(x_{i+1}) = f(x_i) + f'(x_i)(x_{i+1} - x_i) + \dots$$

$$0 = f(x_i) + f'(x_i)(x_{i+1} - x_i)$$

For any function

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \quad \leftarrow \text{way to find the optimal sol}^n \text{ iteratively}$$

$$x_{i+1} = x_i + \frac{f(x_i)}{\left(\frac{1}{x_i^2}\right)}$$

$$= x_i + \left(\frac{1}{x_i} - D\right)x_i^2$$

$$= x_i + x_i - Dx_i^2$$

$x_{i+1} = x_i (2 - Dx_i)$

$Q = \frac{F}{D}$
 $= K \cdot x$
 $K = \frac{1}{D}$
 $f(x) = \frac{1}{x} - D$
 $f'(x) = -\frac{1}{x^2}$

For Division

need to guess x_0 properly
derivation is in the next page

$$e_n = x_n - \frac{1}{D}$$

Error Analysis

$$x_n = e_n + \frac{1}{D}$$

$$e_{n+1} = x_{n+1} - \frac{1}{D}$$

$$e_{n+1} = x_n(2 - Dx_n) - \frac{1}{D}$$

Derivation of Initial guess formula

→ we know optimization formula for $\frac{1}{D}$ is

$$x_{n+1} = x_n (2 - dx \cdot x_n)$$

Initial guess could be close to $\frac{1}{dx}$

dx can be represented as $dx = M 2^E$

$M \rightarrow$ mantissa $[0.5 \leq M \leq 1]$

$E \rightarrow$ exponent (integer)

$$\frac{1}{dx} = \frac{1}{M 2^E} = \frac{1}{M} 2^{-E}$$

$$0.5 \leq M \leq 1$$

$$1 \leq \frac{1}{M} \leq 2$$

so simple approx for x_0 will be

$$x_0 \approx 2^{-E}$$

$$\boxed{\log_2(dx)}$$

choosing best approximations

$$dx = M \cdot 2^E$$

$$\log_2(dx) = \log_2(M) + E \quad \text{as } M < 1 \text{ we can approx to}$$

$$[\log_2(dx)] \approx E$$

$$\text{So, } x_0 = 2^{-[\log_2(dx)]}$$

This approx works for small & large denominators.

```
def newton_raphson_iterative(nr, dr, init_guess, eps=1e-5):
    x_init = init_guess

    for i in range(100):
        x_ipp = x_init*(2-dr*x_init)
        x_init = x_ipp

    x_optim = x_init
    return nr*x_optim

newton_raphson_iterative([15, 23], 0.08)

0.6521739130434783
```

Python output for 15/23 using Newton Raphson method.

```
1065 division_res = 660 = 0294 = 0000001010010100
```

Verilog output for 15/23. 660 is the decimal equivalent scaled value when divided by 2^{10} (1024) gives 0.64453, which closely matches with the Python result.

Division of arrays

```
nr = np.array([15, 2, 15, 29, 9, 19, 2, 18, 19])
dr = np.array([23, 11, 13, 17, 19, 29, 7, 2, 3])
initial_guess = 2 ** (-np.ceil(np.log2(dr)))
```

Numerator and denominator arrays I have taken and computed initial guesses as per my notes.

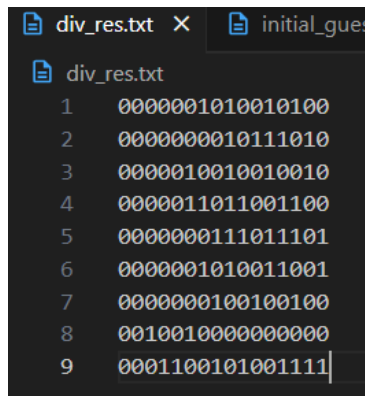
```
newton raphson output = [0.65217391 0.18181818 1.15384615 1.70588235 0.47368421 0.65517241
0.28571429 9. 6.33333333]
```

nr/dr array result from python.

Calculated fixed point Q(6, 10) for nr, dr, and initial guess, and the binary strings were stored and given as inputs to the verilog memory

```
initial_guess.txt dr.txt nr.txt div_res.txt
nr.txt
1 0011110000000000
2 0000100000000000
3 0011110000000000
4 0111010000000000
5 0010010000000000
6 0100110000000000
7 0000100000000000
8 0100100000000000
9 0100110000000000
```

Verilog division module output is stored as a txt file to view in Python error calculation.



```
div_res.txt X initial_gues
div_res.txt
1 0000001010010100
2 0000000010111010
3 0000010010010010
4 0000011011001100
5 0000000111011101
6 0000001010011001
7 0000000100100100
8 0010010000000000
9 0001100101001111
```

Python outputs vs verilog division module outputs.

```
error bw python div and fpga div = [0.00764266 0.00017756 0.01126803 0.0066636 0.0078639 0.00575835
0.00055804 0.0061849 ]
mean absolute error between python output and verilog output = 0.005124114724003176
```