Parametric, Floating Point Arithmetic Logic Unit

by Nathan Sketch, Saher Al-Khrayyef, Raymon Benjamin
Objectives

- Design Parametric Floating Point Arithmetic Logic Unit
  - Addition
  - Subtraction
  - Multiplication
- Interface the FPU to a ARM processor via an AXI4 lite bus.
IEEE single and double precision floating point representation

This figure shows the IEEE single and double precision floating point format.

- Where s is the sign bit.
- Exp is the exponent, it depends on the binary number size.
- Mantissa, also it depends on the binary number size.
- Bias single is 127 and double is 1023
**IEEE-754 Standard Representation**

- The representation is as follows:

\[ X = \pm 1 \cdot f \times 2^e \]

Significand: \( f \) is the mantissa. We should add 1 to the beginning of the mantissa before start addition/subtraction.

Significand range should be \([1, 2 - 2^{-p}] = [1, 2)\)

Biased Exponent:

- \( E \) is the number of the bits.
- Bias = 127 or 1023
- \( e = \text{exp} - \text{bias} \)
Floating Point Addition/Subtraction Equations

\[ b_1 = \pm s_1 \times 2^e_1, \quad s_1 = 1.f_1 \]

\[ b_2 = \pm s_2 \times 2^e_2, \quad s_2 = 1.f_2 \]

- \[ b_1 + b_2 = \pm s_1 \times 2^e_1 \pm s_2 \times 2^e_2 \]
Floating Point Adder/Subtractor Sign

● An add-subtract operation has three sign inputs, a_sign, b_sign, and op.

● We can transform the normal equation into one that better serves the output floating-point format.

\[
\begin{align*}
-a+(-\ b) & \quad a-(+b) \\
-a-b & \quad a-b \\
-(a+b) & \quad a-b \\
\end{align*}
\]
DESIGN OF FLOATING POINT ADDITION/SUBTRACTION

- Flow Chart Diagram

1. e1 and e2 input
   - Yes: e1 >= e2
   - NO: ep = e2, no swap here for f.

   - sm = 0
     - ep = e1, e1 subtracts i to get the final e
     - Add the sign number and find the 2^c for s1 and s2
     - Normalized to get the final f

   - sm = 1
     - ep = e2, e2 subtracts i to get the final e

2. swap f_x = f2, f_y = f1, subtract e1 from e2, and shift s_x to the right to normalize s_x.
Normalized left shift

- The normalized left shift in the post-normalization step removes leading zeros.
DESIGN OF FLOATING POINT ADDITION/SUBTRACTION .... Cont.
Addition Simulation Part 1
Addition Simulation Part 2

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Name</th>
<th>Value</th>
<th>Name</th>
<th>Value</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>expWidth</td>
<td>11</td>
<td>fracWidth</td>
<td>52</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>expWidth</td>
<td></td>
<td>fracWidth</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Subtraction Simulation Part 1

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>0.000000 ns</th>
<th>0.000010 ns</th>
<th>0.000020 ns</th>
<th>0.000030 ns</th>
<th>0.000040 ns</th>
<th>0.000050 ns</th>
<th>0.000060 ns</th>
<th>0.000070 ns</th>
<th>0.000080 ns</th>
</tr>
</thead>
<tbody>
<tr>
<td>num1[63:0]</td>
<td>2.5</td>
<td>0.0</td>
<td>9.25</td>
<td>0.0</td>
<td>3.5</td>
<td>0.0</td>
<td>3.5</td>
<td>2.5</td>
<td>1.5</td>
<td>1.5</td>
</tr>
<tr>
<td>num2[63:0]</td>
<td>3.5</td>
<td>0.0</td>
<td>9.25</td>
<td>2.5</td>
<td>3.5</td>
<td>0.0</td>
<td>3.5</td>
<td>2.5</td>
<td>1.5</td>
<td>1.5</td>
</tr>
<tr>
<td>num3[63:0]</td>
<td>-1.0</td>
<td>0.0</td>
<td>-9.25</td>
<td>2.5</td>
<td>-1.0</td>
<td>0.0</td>
<td>-1.0</td>
<td>2.5</td>
<td>-1.0</td>
<td>1.5</td>
</tr>
<tr>
<td>num4[63:0]</td>
<td>1.0</td>
<td>0.0</td>
<td>9.25</td>
<td>-9.25</td>
<td>1.0</td>
<td>0.0</td>
<td>-1.0</td>
<td>1.0</td>
<td>-1.0</td>
<td>-1.0</td>
</tr>
</tbody>
</table>
Subtraction Simulation Part 2
DESIGN OF FLOATING POINT MULTIPLICATION EQUATIONS

- $b_1 = \pm s_1 \times 2^{e_1}$
  $b_2 = \pm s_2 \times 2^{e_2}$
- $b_1 \times b_2 = (\mp s_1 \times 2^{e_1}) \times (\pm s_2 \times 2^{e_2}) = \pm (s_1 \times s_2) \times 2^{(e_1 + e_2)}$
- $s = (s_1 \times s_2) \in [1, 4)$. 
DESIGN OF FLOATING POINT MULTIPLICATION

- Sign Bit Calculation
- Multiplying two number's result is a negative sign if one of the multiplied numbers is of a negative value. By the aid of a truth table we find that this can be obtained by XORing the sign of two inputs.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Y = A ⊕ B</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
DESIGN OF FLOATING POINT MULTIPLICATION .... Cont.

Flow chart Diagram

Multiply

Is num1 = 0?

Yes

Product = 0

No

Is num2 = 0?

Yes

Subtract Bias

Multiply Significands

Normalize

Final Product

Add Exponents

No

No
Multiplication Simulation
Floating Point Top Circuit
Top Floating Point Arithmetic Circuit Simulation
Addition/Subtraction/Multiplication AXI4 lite Interface

- slv_0 (63,32) → in1
- slv_1 (31,0) → in1
- slv_2 (63,32) → in2
- slv_3 (31,0) → in2
- in1 → answer
- in2 → Mult/Add/Sub
- answer → slv_5 (63,32)
- answer → slv_6 (31,0)
- slv_4 (4),(0) → Mult/Add/Sub

<table>
<thead>
<tr>
<th>Bit1</th>
<th>Bit2</th>
<th>Op</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Add</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Sub</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Mult</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Mult</td>
</tr>
</tbody>
</table>
Challenges/Improvements

- Challenges VHDL
- Improvements Testbench and Divider
Any Questions?

Thank You