FPU Generator For Design Space Exploration

Sameh Galal, Ofer Shacham, John S. Brunhaver II, Jing Pu, Artem Vassiliev and Mark Horowitz
Stanford University
An FPU generator in 2013?!

- **4800 articles in IEEE Xplore**
  - When you search for floating point

- **After all these papers and 44 years of ARITH,**
  - It is still pretty hard to design an FPU

- **One size doesn’t fit all.**
  - CPUs: 10s of FPUs, Double Precision, Low Latency
  - GPUs: 1000s of FPUs, Single Precision, High Throughput
  - Mobile: 100s of FPUs, Half-Precision, Low power
Converting the problem to a solution

- Procedurally encode this huge body of existing tricks

- Incorporate this knowledge into parameterized generators
  - These are organized hierarchically
  - Use optimization to find best solution for your target

- Generate an FPU for every different requirement
  - Precision
  - Energy Efficiency
  - Latency/Throughput
But how to build a generator?

- There is no magic bullet.
- Need to optimize at all design levels
- Used our Genesis2 generator language
Capturing the designer knowledge: Abstraction

- This is a multiplier
- The problem is that many tricks exist, at arch, uarch, circuit and layout levels
- A generator enabled us to encode them all
- The results surprised us!
Q1: Booth encoding: Which is better for energy efficiency?

- Booth 2
  - 13 Partial Products
  - Slow 3x multiple Generation

- Booth 3
  - Smaller Tree
A1: It’s about the area

Booth 2

- $A_{Booth2\_mux} = 0.625 \ A_{CSA}$
- $A_{total} \approx \frac{1}{2} (1+0.625) n^2 A_{CSA}$
  $= 0.8125 \ n^2 A_{CSA}$

Booth 3

- $A_{Booth3\_mux} = A_{CSA}$
- $A_{total} \approx \frac{1}{3} (1+1) n^2 A_{CSA}$
  $= 0.6667 \ n^2 A_{CSA}$
A1: It’s about the area

**Booth 2**

- \( A_{\text{Booth2\_mux}} = 0.5 \, A_{\text{CSA}} \)
- \( A_{\text{total}} \approx \frac{1}{2} \left( 1 + 0.5 \right) n^2 \, A_{\text{CSA}} \)
- \( = 0.75 \, n^2 \, A_{\text{CSA}} \)

**Booth 3**

- \( A_{\text{Booth3\_mux}} = A_{\text{CSA}} \)
- \( A_{\text{total}} \approx \frac{1}{3} \left( 1 + 1 \right) n^2 \, A_{\text{CSA}} \)
- \( = 0.6667 \, n^2 \, A_{\text{CSA}} \)
Q2: Partial products reduction: What are the most important factors?

- The combining element
  - 3:2 Counter
  - 4:2 Compressors
  - 7:3 counter

- Number of Levels of CSA
  - Wallace (shortest)
  - Trees: Overturned Staircase, ZM

- Routing and wire tracks of the tree
A2: It matters how counters are connected!

- This is a Double Precision Wallace Booth 2 multiplier tree

- Designer insight:
  - ‘Sum’ output takes 1 unit delay and ‘Carry’ output takes 0.5 unit delay
  - Balancing the interconnection of sum and carries
    - Lets 7 levels of CSA take only 5.5 CSA delays
### A2: CSA simplest and most powerful

<table>
<thead>
<tr>
<th>Circuit</th>
<th>3:2 Counter</th>
<th>4:2 Compressor</th>
<th>7:3 Counter [12]</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Circuit Diagram" /></td>
<td><img src="image" alt="Circuit Diagram" /></td>
<td><img src="image" alt="Circuit Diagram" /></td>
<td><img src="image" alt="Circuit Diagram" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Element Delay</th>
<th>1</th>
<th>1.5</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tree Delay</td>
<td>$0.75 \frac{\log_3 n}{\log_{1.72} n} = 1.5 \frac{\log_2 n}{\log_{1.59} n}$</td>
<td>$2 \log_{7/3} n = \log_{1.53} n$</td>
<td></td>
</tr>
<tr>
<td>Element Area</td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Elements</td>
<td>$n - 2$</td>
<td>$\frac{n - 2}{2}$</td>
<td>$\frac{n - 3}{4}$</td>
</tr>
<tr>
<td>Tree Area</td>
<td>$n - 2$</td>
<td>$n - 2$</td>
<td>$n - 3$</td>
</tr>
</tbody>
</table>
A2: Layout matters!

- Adding layout hints at the generator level
  - Where the knowledge is
- Pass them through to layout
- Enables quick exploration of many layout scenarios
Putting it all together

Resulting Efficient Designs for Double precision
A2: Wires matter!

For quad precision, long wires result in OS trees becoming more energy efficient.
Q3: What is the best FPU architecture?

Cascade (CMA)

- Exponent Difference
- Leading Zero Anticipator (106 bits)
- Normalizer (106 bits)
- Accumulation Bypass
- CLOSED PATH
- Subtract Exp Diff ≤1
- Multiply-Add Bypass
- Rounder (53 bits)
- Significand Result

Fused (FMA)

- MULTIPLIER
- EXPONENT DIFFERENCE ≥ 1
- LEADING ZERO ANTICIPATOR
- ADDER
- 3:2 CSA
- 2:1 MUX
- 2:1 MUX
- Aligner (159 bits)
- EAC Adder (106 bits)
- Shift 1 (53 bits)
- FAR PATH
- Multiply-Add Bypass
- Rounder (53 bits)
- Significand Result
A3: It depends..

For Latency

For Throughput
Q4: What about my special feature?

1. Can we reuse a double-precision multiplier as two single precision?

2. How hard would it be to add this “smart” to FPGen?
   - Turns out that in a generator, it is hard, but not that hard…

\[
\begin{align*}
\Sigma PP[i] &= A \times B \\
\Sigma PP'[i] &= \{b \times d, a \times c\}
\end{align*}
\]
A4: Yes, added multiple precision support

- Overhead can be quite small

OS1 Tree

Wallace Tree
**Conclusion**

- **Details matter!**
  - Booth mux area
  - Wiring of CSAs…

- **Built a generator that incorporates this knowledge**
  - Used it to explore optimal design
  - Results are better than SoA FP IP blocks
  - (multiple precision, pipelining, ..etc.)

- **You can try your ideas too.**
THANK YOU!

Visit

http://vlsiweb.stanford.edu/fpgen/