Wallace Multiplier with Booth Encoding

Sharzy

2021.10.11

What is a multiplier

A circuit that multiply two numbers.

Input:

$$a = a_{n-1}a_{n-2}\cdots a_0 = \sum_{i=0}^{n-1} 2^i a_i$$

$$b = b_{n-1}b_{n-2}\cdots b_0 = \sum_{i=0}^{n-1} 2^i b_i$$

Output:

$$\begin{aligned} z &= z_{2n-1}z_{2n-1}\cdots z_0 \\ &= a \times b \end{aligned}$$

Wallace Multiplier

Directly expand the product:

$$a \times b = \sum_i \sum_j 2^{i+j} a_i b_j \tag{1}$$

Now we have n^2 partial product $2^{i+j} a_i b_j$, $a_i b_j$ is 0 or 1.
Define 2^{i+j} be the "weight" of such a partial product.

Naïve method: sum up all partial products in a binary tree.

Since n-digit addition takes $O(\log n)$ time, this method takes $O(\log^2 n)$ time.

Wallace Multiplier

Optimization: group partial products by their weight (each group called a column). Only add up partial products in the same group.

$$2^i a_1 + 2^i a_2 + 2^i a_3 = 2^{i+1} b_1 + 2^i b_2 \quad \text{(Full adder)}$$

Transform 3 partial products to 2 (3-2 compressor)

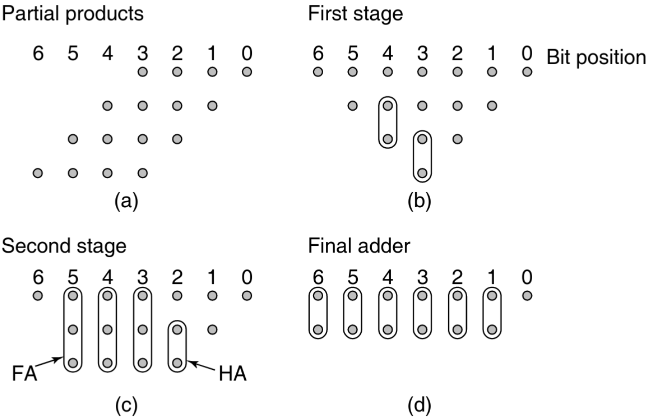
$$2^i a_1 + 2^i a_2 = 2^{i+1} b_1 + 2^i b_2 \quad \text{(Full adder)}$$

Transform 2 partial products to 2 (2-2 compressor)

In each stage we compress each column as much as possible. Until there are at most two items in each column.

It remains to execute an $2n$-bit addition.

Wallace Multiplier: Example



Notice that in each stage we reduce the number of partial products by roughly $1/3$, there are roughly $\log_{3/2}(n)$ stages.

---

[0] Figure from `http://www.cse.psu.edu/~kxc104/class/cmpen411/16s/lec/C411L20Multiplier.pdf`

Optimization Techniques

Fast enough. But may I make it even faster?

A wallace multiplier consists of three parts:

- **Partial product generation**: generate less partial products
- **Compression**: compress more in one stage (use 4-3, 5-3 compressor)
- **Addition**: faster adder (not covered in this topic)

Last two optimizations are easy. The first is tricky.

Booth Recoding

Define $a_{-1} = 0$, $a_n = a_{n-1}$ (sign extension).

$$
\begin{aligned}
a \times b = \sum_{i=0}^{n-1} 2^i a_i b &= \sum_{i=0}^{n-1} (2^{i+1} - 2^i) a_i b \\
&= \sum_{i=0}^{n-1} 2^i (-a_{i+1} + a_i) b
\end{aligned}
$$

Example:

$$
1100111 = 2^7 - 2^5 + 2^3 - 2^0
$$

Now we have

$$
1100111 \times b = 2^7 b + 2^5(-b) + 2^3 b + 2^0(-b) \tag{2}
$$

Not faster in worst case.

Radix-4 Booth Recoding

$$
\begin{aligned}
a &= \sum_{i=0}^{n-1} 2^i a_i \\
&= \sum_{i=0}^{(n-1)/2} 4^i(-2a_{2i+1} + a_{2i} + a_{2i-1})
\end{aligned}
$$

Look like this:

$$
\begin{array}{rcccccccl}
 & & & & (-2a_1 & +a_0 & +a_{-1} &)4^0 \\
 & & (-2a_3 & +a_2 & +a_1 & & &)4^1 \\
 & (-2a_5 + a_4 + a_3 & & & & & &)4^2 \\
= & & \cdots + 8a_3 & +4a_2 & +2a_1 & +a_0 & + \cdots &
\end{array}
$$

b and $2b$ can be calculated immediately.

$-b = \overline{b} + 1$ (invert all bits, then plus 1).

Radix-4 Booth Recoding: Example

5-bit multplication: $(-10) \times (-14) \Longrightarrow 10110_{(2)} \times 10010_{(2)}$.

Booth-recode: $10010_{(2)} = (-1) \times 4^2 + 1 \times 4^1 + (-2) \times 4^0$

$$
\begin{array}{cccccccccc l}
\color{blue}{0} & \color{blue}{0} & \color{blue}{0} & \color{blue}{0} & 0 & 1 & 0 & 0 & 1 & 0 & + \overline{10110}_{(2)} \times 2 \times 4^0 \\
\color{blue}{1} & \color{blue}{1} & \color{blue}{1} & 1 & 0 & 1 & 1 & 0 & \color{red}{1} & \color{red}{0} & + 10110_{(2)} \times 1 \times 4^1 \color{red}{+4^0 \times 2} \\
\color{blue}{0} & 0 & 1 & 0 & 0 & 1 & \color{red}{0} & \color{red}{0} & & & + \overline{10110}_{(2)} \times 1 \times 4^2 \\
 & & & & \color{red}{0} & \color{red}{1} & & & & & + \color{red}{4^2} \\
\hline
0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & = 140
\end{array}
$$

Reduce the number of partial products by nearly $1/4$.

Radix-8 Booth Recoding

Even more?

$$
\begin{aligned}
a \times b &= \sum_{i=0}^{n-1} 2^1 a_i b \\
&= \sum_{i=0}^{(n-1)/3} 8^i z_i b \quad \text{(corner case ignored)}
\end{aligned}
$$

Where
$z_i = -4a_{3i+2} + 2a_{3i+1} + a_{3i} + a_{3i-1} \in \{\pm 4, \pm 3, \pm 2, \pm 1, 0\}$.

In this case we need to calculate $3b$, which is not cheap.

Tradeoff between the cost of two stages.

Chisel Implementation

Work In Progress

`https://github.com/sequencer/arithemic/blob/multiplier/arithmetic/src/multiplier/WallaceMultiplier.scala`

\Thanks
\end{presentation}