Study on Digital Multiplier Architecture Using Square Law

群馬大学大学院 理工学府
電子情報・数理教育プログラム
孫 逸菲 (Sun Yifei, ソン・イッヒ)
OUTLINE

- Research Background
- Digital Multiplier Algorithm
- Design of Multiply Circuit and Simulation Verification
- Circuit Design Using Squaring Calculation Logic and Simulation Verification
- Compared with Each methods
- Future Work
- Conclusion
OUTLINE

- Research Background
  - Digital Multiplier Algorithm
  - Design of Multiply Circuit and Simulation Verification
  - Circuit Design Using Squaring Calculation Logic and Simulation Verification
  - Compared with Each methods
- Future Work
- Conclusion
Research Background

Digital arithmetic devices
- Adder
- Subtractor
- Multiplier
- Divider

DSPs, µ Processors use several digital multipliers on a chip.

Requirements
- Small scale
- Low power
- High speed

- Digital multiplier hardware implementation algorithm has been a research topic for 50 years.
- Decrease of the multiplier scale is still a research topic.
How Digital Multiplier Works

**Decimal**

```
25  multiplicand
x 39  multiplicator
```

```
45
18
15
+ 6
975  product
```

**Binary**

```
011001 : 25_{10}  multiplicand
x100111 : 39_{10}  multiplicator
```

```
011001
011001
011001
000000
000000
+ 011001
```

```
001111001111 : 975_{10}  product
```

Calculation of the sum of partial products increases
Purpose of Study

Composition of array digital multiplier

Multiplier (Using 2D array of full adders)
- Circuit size
- Power
- Computation time

BIG

Ex: In 6bit × 6bit situation
6 × 6 = 64 full adders are needed

Reduce

circuit size • power • computation time
OUTLINE

- Research Background
- Digital Multiplier Algorithm
- Design of Multiply Circuit and Simulation Verification
- Circuit Design Using Squaring Calculation Logic and Simulation Verification
- Compared with Each methods
- Future Work
- Conclusion
Investigated Multiplier Algorithm

Based on square law

\[ AB = \frac{1}{2} [(A + B)^2 - (A^2 + B^2)] \]

- Squaring 3 times
- Addition twice
- Subtraction once
- \( \frac{1}{2} \) operation can be realized with a 1-bit left shift or just interconnection change

Realization circuit

General Multiplier Algorithm

\[ AB = A + A + \cdots + A \]

Multiplier A×B
Number of additions : B times
What is Look Up Table (LUT)

$A \rightarrow A^2$

**LUT Memory (ROM, RAM)**

<table>
<thead>
<tr>
<th>Address</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>100</td>
<td>10000</td>
</tr>
</tbody>
</table>

No calculation

Data

<table>
<thead>
<tr>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10000</td>
</tr>
</tbody>
</table>

Using LUT  ➔ Memory reference processing

Disadvantage

Efficient

LUT processing ⇒ handled large number of bits ⇒ Large circuit size
OUTLINE

- Research Background
- Digital Multiplier Algorithm
- Design of Multiply Circuit and Simulation Verification
- Circuit Design Using Squaring Calculation Logic and Simulation Verification
- Compared with Each methods
- Future Work
- Conclusion
Improvement Plan of Implementation Circuit

Come up with Divide & Conquer method

$$AB = \frac{1}{2}[(A + B)^2 - (A^2 + B^2)]$$

A, B, A+B divide into upper bits and lower bits for calculation scale reduction

In 8 bit case: divide into upper 4 bits and lower 4 bits

LUT size becoming smaller
Number of Bits Handled by LUT

Reduce required number of bits using Divide & Conquer

The number of input bits is reduced by half → LUT size will be significantly reduced
Divide & Conquer Method Analysis

In 8 bit case \( (A = 11001001 : 201_{10}) \)

8bit x 8bit divide 4bit \([A_H], [A_L]\]
Calculated by each \([A_H], [A_L]\)

Divided input, output values up and down

\[ A = 11001001 \]
\[ A_H = 1100 : 12_{10} \]
\[ A_L = 1001 : 9_{10} \]

\[ A_H^2 = 10010000 : 144_{10} \]
\[ A_L^2 = 1010001 : 81_{10} \]

\[ A_H A_L = 1101100 : 108_{10} \]
Divide & Conquer Method Analysis

\[ A^2 = A_H^2(\text{Nbit left shift}) + A_HA_L\left(\frac{N}{2} + 1\right)\text{bit left shift} + A_L^2 \]

\[ N=8 \text{ bit situation} \]

\[ A^2 = A_H^2(8\text{bit left shift}) + A_HA_L(5\text{bit left shift}) + A_L^2 \]

\[ A_H^2 = 10010000 \]
\[ A_HA_L = 1101100 \]
\[ A_L^2 = 1010001 \]
\[ A_H^2(8\text{bit left shift}) = 10010000000000000 \]
\[ A_HA_L(5\text{bit left shift}) = 110110000000 \]
\[ A_L^2 = 1010001 \]

\[ A^2 = 10011101110100001 : 40401_{10} \]

\( (A^2 = 201 \times 201 = 40401) \)
Divide & Conquer Method Circuit

\[ A(N\text{bit}) \xrightarrow{X \text{LUT} X^2} A^2(2N\text{bit}) \]

\[ A^2 = A_H^2(N\text{bit left shift}) + A_H A_L \left( \frac{N}{2} + 1 \right) \text{bit left shift} + A_L^2 \]

Using Divide & Conquer with \(X\) times, LUT size will decrease \(2^X\) times
Divide & Conquer Method Circuit (8 bit case)

\[ AB = \frac{1}{2} [(A + B)^2 - (A^2 + B^2)] \]

By dividing, the bit of LUT becoming smaller

Need LUT for 16 bit

Need LUT for 8 bit
RTL Simulation (8 bit × 8 bit)

Input values A, B are changed every 100 ns and 200 ns.
A, B: input
G : output.

G = A × B
OUTLINE

- Research Background
- Digital Multiplier Algorithm
- Design of Multiply Circuit and Simulation Verification
- Circuit Design Using Squaring Calculation Logic and Simulation Verification
- Compared with Each methods
- Future Work
- Conclusion
Do NOT use LUT to Implement Square Law

We have found that direct logic circuit Implementation of squaring can be simple.

Large size of LUT increases the overall circuit area
Direct Squaring Calculation Logic Circuit

\[ C = A + B \]

LUT part was replaced with squaring calculation circuit.
### Truth Table and Logic Expression

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>I3 I2 I1 I0</td>
<td>O7 O6 O5 O4 O3 O2 O1 O0</td>
</tr>
<tr>
<td>0 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>1 0 0 1</td>
<td>1 0 0 0 0 0 0 0 1</td>
</tr>
<tr>
<td>2 0 0 1 0</td>
<td>4 0 0 0 0 0 0 1 0 0</td>
</tr>
<tr>
<td>3 0 0 1 1</td>
<td>9 0 0 0 0 0 1 0 0 1</td>
</tr>
<tr>
<td>4 0 1 0 0</td>
<td>16 0 0 0 1 0 0 0 0 0</td>
</tr>
<tr>
<td>5 0 1 0 1</td>
<td>25 0 0 0 1 0 0 0 1 0 0 1</td>
</tr>
<tr>
<td>6 0 1 1 0</td>
<td>36 0 0 1 0 0 1 0 0 1 0 0</td>
</tr>
<tr>
<td>7 0 1 1 1</td>
<td>49 0 0 1 1 0 0 0 1</td>
</tr>
<tr>
<td>8 1 0 0 0</td>
<td>64 0 1 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>9 1 0 0 1</td>
<td>81 0 1 0 1 0 0 0 1</td>
</tr>
<tr>
<td>10 1 0 1 0</td>
<td>100 0 1 1 0 0 1 0 0 0 1</td>
</tr>
<tr>
<td>11 1 0 1 1</td>
<td>121 0 1 1 1 1 0 0 1</td>
</tr>
<tr>
<td>12 1 1 0 0</td>
<td>144 1 0 0 1 0 0 0 0 0</td>
</tr>
<tr>
<td>13 1 1 0 1</td>
<td>169 1 0 1 0 1 0 0 1</td>
</tr>
<tr>
<td>14 1 1 1 0</td>
<td>196 1 1 0 0 0 0 1 0 0</td>
</tr>
<tr>
<td>15 1 1 1 1</td>
<td>225 1 1 1 0 0 0 0 1</td>
</tr>
</tbody>
</table>

In the situation of output equal to 1, the input are 0011, 0101, 1011, 1101.

\[
O3 = \overline{I3}I2I1I0
\]

\[
O3 = \overline{I3}I2I1I0
\]

\[
O3 = \overline{I3}I2I1I0
\]

\[
O3 = \overline{I3}I2I1I0
\]

Write in a theoretical simplification:

\[
O3 = (I2 \oplus I1)I0
\]

Calculate the logic expression O0~O7.
The Layout of Direct Squaring Calculation Logic Circuit

Circuit creates individual logic expressions by the number of bits of input
Simulation Result

Input 4 bit × 4bit circuit

Using direct squaring calculation logic circuit was validated.
OUTLINE

- Research Background
- Digital Multiplier Algorithm
- Design of Multiply Circuit and Simulation Verification
- Circuit Design Using Squaring Calculation Logic and Simulation Verification
- Compared with Each methods
- Future Work
- Conclusion
Comparison of Various Algorithms

Square law method → faster computation

Using D&C reduces LUT size
OUTLINE

- Research Background
- Digital Multiplier Algorithm
- Design of Multiply Circuit and Simulation Verification
- Digital Multiplier Architecture Using Square Law and Simulation Verification
- Compared with Each methods
- Future Work
- Conclusion
Future Work

- Consideration of multipliers in other methods
  Using \( AB = \frac{1}{4} \{(A + B)^2 - (A - B)^2\} \) multiplier algorithm to realize circuit design

- Create squaring calculation logic circuit of upper bit

- Perform FPGA implementation and confirm operation
OUTLINE

- Research Background
- Digital Multiplier Algorithm
- Design of Multiply Circuit and Simulation Verification
- Circuit Design Using Squaring Calculation Logic and Simulation Verification
- Compared with Each methods
- Future Work
- Conclusion
Conclusion

- Discussion the multiplication algorithm based on square law
- Propose Divide & Conquer method to reduce the LUT size in RTL level validation by simulation
  - reduce computation
  - circuit area reduction
- Consider reduction of multiplication using squaring calculation logic in RTL level validation by simulation
  - create dedicated circuit to calculate square simple
Thanks for your listening
Q: This is the operation used in multiplication, have you considered how to make the other operation (such as sin, cos operation) simple?

A: To use LUT also can realize sin and cos operation, but the capacity of LUT also large. I will consider it in the future. (After publication, I find an paper named Application of LUT Cascades to Numerical Function Generators can let the sin, cos operation simple.)

Q: To explain the figure that comparison of various algorithms.

A: In left picture, the horizontal axis represent the number of bits that need to be multiplied, the vertical axis represent the add times. With the increase in the number of bit, the square law method has less computation adder times.

In right picture, the horizontal axis represent the number of bits that need to be multiplied, the vertical axis represent the number of required bit in LUT. With the increase in the number of bit, using Divide&Conquer method can reduce the size of LUT.