PROJECT DRAFT
ECE 645 : COMPUTER ARITHMETIC : IMPLEMENTATION IN HARDWARE AND SOFTWARE.

ITERATIVE MULTIPLICATION OF LARGE NUMBERS USING KARATSUBA ALGORITHM—RTL IMPLEMENTATION FOR FPGA/90NM TSMC STANDARD-CELL BASED ASIC.

Submitted by,
GURUNATH RAMAGIRI
GRAMAGIR@GMU.EDU
**INTRODUCTION:**

Multiplying two polynomials efficiently is an important issue in a variety of applications, including signal processing, cryptography and coding theory. This project provides a generalization and detailed analysis of the algorithm by Karatsuba to multiply two polynomials which was introduced in 1962. The Karatsuba Algorithm saves coefficient multiplications at the cost of extra additions compared to the paper and pencil multiplication method.

This project aims at reducing the area of multiplier circuit by using this algorithm. The Multiplier and Multiplicand are iteratively segmented and multiplied sequentially. The partial products are stored in registers and added using carry-save adder network.

The above mentioned algorithms are mentioned below.

Paper and pencil Algorithm:

\[ P = A \cdot B = (AH^{2n} + AL) \cdot (BH^{2n} + BL) \]
\[ = AH \cdot BH \cdot 2^{2n} + (AH \cdot BL + AL \cdot BH)2^n + AL \cdot BL. \]

Karatsuba Algorithm:

\[ P = A \cdot B = (AH^{2n} + AL) \cdot (BH^{2n} + BL) \]
\[ = AH \cdot BH \cdot 2^{2n} + (AH - AL)(BL - BH) + (AH \cdot BH) + (AL \cdot BL) \cdot 2^n + AL \cdot BL. \]
ARCHITECTURES FOR THE MULTIPLIER CIRCUIT:

As shown in the multiplier block diagram, it consists of a Long-operand splitting which segments the input operands recursively to the size of the multiplier chosen. In this project, a 32-bit array multiplier is used. So a 128-bit input operands will be split into 9 separate segments each and stored in the memory. Each segment from the multiplier and multiplicand are read from memory sequentially and multiplied. The partial product is saved in registers which vary in size at each level. For example, at the lowest level, the registers are 64 bits each. A level above they are 128-bit wide. There are a total of 3 registers at each level to save the partial products. The scheduler controls the registers individually. At each level, there is a corresponding CSA/CPA adder which adds the 3 partial products and passes it to them level above. Hence the final stage result will be in a 256-bit register. A detailed architecture is shown below.

Fig 2: Detailed architecture of the multiplier
APPLICATIONS OF KARATSUBA MULTIPLIER:
1. Signal processing,
2. Cryptography and
3. Coding theory.

INPUT/OUTPUT SIGNALS FOR MULTIPLIER:

<table>
<thead>
<tr>
<th>Signal</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset</td>
<td>In</td>
<td>Reset signal applied to all registers in the multiplier.</td>
</tr>
<tr>
<td>Clock</td>
<td>In</td>
<td>To clock all register/memory in design</td>
</tr>
<tr>
<td>Data_in</td>
<td>In</td>
<td>Data arrival at the input side</td>
</tr>
<tr>
<td>In_A[31:0]</td>
<td>In</td>
<td>32-bit input A.</td>
</tr>
<tr>
<td>In_B [31:0]</td>
<td>In</td>
<td>32-bit input B.</td>
</tr>
<tr>
<td>Data_out</td>
<td>In</td>
<td>Product available.</td>
</tr>
<tr>
<td>Product[64:0]</td>
<td>Out</td>
<td>64-bit product. (256-bit result will be shifted out sequentially)</td>
</tr>
</tbody>
</table>

LANGUAGE, PLATFORM AND TOOLS:

The multiplier circuit will be implemented in VHDL and synthesized for Xilinx virtex3 family of FPGAs and 90nm TSMC library based standard-cell ASICs.

Tools:
1. Active-HDL 7.1(Functional simulation),
2. Synplicity Synplify Pro(synthesis),
3. Xilinx ISE 8.x (Implementation, placement and routing) and
4. Synopsys Design Analyzer (ASIC synthesis, implementation, placement and routing).
REFERENCES:

