FPGA IMPLEMENTATION OF NIST P-384 MODULAR MULTIPLIER

Yuqiu Jiang, Yangpingqing Hu
Advisor: Weizhong Wang
Electrical Engineering

OBJECTIVE

NIST p-384 modular multiplier
- Implemented with Xilinx FPGA
- Tailored structure enhances computation speed
- High performance per watts
- Maximizes the utilization of modern FPGA features

APPRAACH

- The proposed structure consists of 3 pipeline stages: Multiplication, addends arrangement and addition.
- Multiplication stage:
  - Using each DSP slice independently, multiplication processed in parallel.
  - Generate three 768-bit results, which are added in addition stage.
- Addends arrangement stage:
  - Rearrange the multiplication results based on the NIST p-386 standard [1].
  - Utilizing FPGA block memory to shift and store the long integers.
- Addition stage:
  - Using DSP to speed up the addition process.
  - Proposing a new structure by introducing a storage bit between DSPs to reduce the dependency of each addition process.

METHODOLOGY

Multiplication Stage

Multiplication Stage takes in two 384-bit input integers and computes the multiplication process with 24 DSP slices. The multiplication strategy is shown in the lower right table. On the left side is a comparison to the traditional multiplication strategy.

<table>
<thead>
<tr>
<th>Stage</th>
<th>DSP1</th>
<th>DSP2</th>
<th>DSP3</th>
<th>...</th>
<th>DSP24</th>
<th>DSP25</th>
<th>DSP26</th>
<th>...</th>
<th>DSP30</th>
<th>DSP31</th>
</tr>
</thead>
<tbody>
<tr>
<td>stage1</td>
<td>32768</td>
<td>32768</td>
<td>32768</td>
<td>...</td>
<td>32768</td>
<td>32768</td>
<td>32768</td>
<td>...</td>
<td>32768</td>
<td>32768</td>
</tr>
<tr>
<td>stage2</td>
<td>32768</td>
<td>32768</td>
<td>32768</td>
<td>...</td>
<td>32768</td>
<td>32768</td>
<td>32768</td>
<td>...</td>
<td>32768</td>
<td>32768</td>
</tr>
<tr>
<td>stage3</td>
<td>32768</td>
<td>32768</td>
<td>32768</td>
<td>...</td>
<td>32768</td>
<td>32768</td>
<td>32768</td>
<td>...</td>
<td>32768</td>
<td>32768</td>
</tr>
</tbody>
</table>

As shown in the table, each column is a DSP slice and each row is the task assigned to each DSP. There are two different types of tasks in the table. The white cell multiplies two 32-bit values, while the shaded cell multiplies and clears the value stored in the accumulators of each DSP slice.

The registered values are further concatenated into the shown structure. First and second long integers are the lower 32-bits of each registered accumulator value. The third long integer is the carry portion of the accumulated value. Since this value is registered separately, it would enable the proposed structure to be calculated in parallel.

Addends Sorting Stage

- Each of the integer will be rearrange according to NIST recommended elliptic curve algorithm on modular multiplication on p-384. The rearranged values are stored in block memory. We categorized these addends into two bases on the length. The results will be further passed to addition stage.

Addition Stage

The task of this part is basically described by following equation:

\[ B = (a + 2s_1 + s_2 + s_3 + s_4 + s_5 + s_6 + s_7 - (d_1 + d_2 + d_3)) \mod p \]

Two special details in this stage are described as follows.

- a. 48-bit DSP adder cell
  - 48-bit DSP adder cell is the sub-module of the 384-bit adder and 192-bit adder. One 48-bit DSP adder cell includes one 48-bit DSP and some peripheral parts called Storage-bit. Traditionally, DSPs for higher bits have to wait the lower ones generating the carry-out, which makes the whole process cost at least 9 clock cycles for 384-bit numbers. To speed up, between two DSPs, we set a Storage-bit, which would take the 48θ bit of every 48-bits. The Storage-bit would temporarily store the addition result of every 48-th bit. Because the Storage-bit is the computation result of only single one bit, it can be done quickly. So the higher DSP can immediately take the result as the Carry-In.

- b. Modular reduction
  - The final step is to do the modular reduction, that is to take \( n + p \) off from temporary product \( S \). Also at this step, we will take the Storage-bit off, so we will finally get a normal number as the result. The task of this step is:

  \[ \text{FinalResult} = S - \text{Storage} - n + p \]

RESULT

Our design has relatively low frequency, since our the datathrough is wide as either 192-bit or 384-bit. Finishing single one prime field arithmetic computing costs about 2.44μs. With our pipeline design, computing multiple times, latency for each computation would be reduced to 0.92μs.

When compared with other designs, the MULT in our architecture is the best, only requiring 32 clock cycles to perform one operation, while the one in [3] needs 70 clock cycles, and the one in [4] needs 178 clock cycles. This is the biggest advancement in this design. The ADD/SUB is at same level with the one in [4].

CONCLUSION

An architecture of the NIST prime field arithmetic unit is proposed. By inserting the Storage-bit and incorporating properly the DSP48 built in Xilinx FPGA board, the speed of addition/subtraction operation is dramatically improved. According to our test, this design can increase the efficiency of the prime field multiply and the modular reduction, which are crucial part in elliptic curve cryptography. This unit can be injected into other ECC and increase the general performance. Besides, a special consideration of the power consumption is made, as we expect such design would be widely used in mobile devices.

Future work would be a complete design of ECC, which would take advantage of the NIST prime field arithmetic unit, further increasing the general performance. Another potential improvement lands on the support for more NIST prime numbers.

REFERENCES