FPGA Implementation of a Resource Efficient Vedic Multiplier using SPST Adders

K. V. Gowreesrinivas
Department of ECE, Anil Neerukonda Institute of Technology & Sciences (ANITS), India
srinivas.ece@anits.edu.in (corresponding author)

Srinivas Sabbavarapu
Department of ECE, Anil Neerukonda Institute of Technology & Sciences (ANITS), India
ssrinivas.ece@anits.edu.in

Punniakodi Samundiswary
Department of Electronics Engineering, Pondicherry University, India
samundiswary_pdy@yahoo.com

Received: 19 February 2023 | Revised: 27 March 2023 | Accepted: 29 March 2023
Licensed under a CC-BY 4.0 license | Copyright (c) by the authors | DOI: https://doi.org/10.48084/etasr.5797

ABSTRACT
Nowadays, the requirement for very high-speed operations in processors constantly increases. Multiplication is a crucial operation in high power-consuming processes such as image and signal processing. The main characteristics of a multiplier are good accuracy, speed, reduction in area, and little power consumption. Speed plays a major role in multiplication operations, and an increase in speed can be obtained by reducing the number of steps involved in the computation process. Since a multiplier has the largest delay among the basic blocks in a digital system, the critical path is determined by it. Furthermore, the multiplier consumes more area and dissipates more power. Hence, designing multipliers that offer high speed, lower power consumption, less area, or a combination of them is of prime concern. Hence, an attempt is made in this paper to achieve the above design metrics using a Spurious Power Suppression Technique (SPST) adder. A resource-efficient SPST-based Vedic multiplier is developed and implemented using Artix 7 FPGA and is finally compared with the ripple carry adder-based Vedic multiplier.

Keywords-vedic multiplier; Urdhva Triyakbhyam; Spurious Power Suppression Technique (SPST); Verilog; Artix7

I. INTRODUCTION
The operation of multiplication plays a pivotal role in many operations and it is important to be computed fast with great efficiency in Digital Signal Processing (DSP) and Image Processing, where functions like Discrete Fourier Transform (DFT) and Fast Fourier Transform (FFT) are used. The research to obtain improved multiplier versions and establish new method types for efficacious multiplier architecture is still ongoing. Each multiplier is different in its own way. High performance with a less number of partial products is preferred in a multiplier. Much research has been conducted regarding the ancient Vedic mathematics, which bought many efficacious multipliers, dividers, squaring, and cube architectures. During the last few years, various decimal algorithms have been extended to the binary number system based on the Vedic mathematics [1-4]. Reducing the number of stages in a Vedic multiplication process leads to a great reduction in the propagation delay. The multiplication process is based on Urdhva Tiryagbhyan algorithm/sutra, which belongs to the 16 sutras of Vedic mathematics [5-7]. Generally, most multipliers consume area, power, and work with little accuracy. Therefore, the design of multipliers which offer high speed, low power consumption, less area, or combinations of all these, is of immense research interest [8-15].

II. VEDIC MULTIPLIER
Indian mathematics has a unique technique of arithmetic computation based on 16 sutras or formulas, which is applicable in many fields. One of the fastest developing fields is DSP. So, there is a need to implement as fast as possible multiplication operation for the digital processing of signals. The Vedic multiplier is mainly based on the vertical and crosswise algorithm of Urdhva Triyagbyham generating all partial products and sums in a single step. The UT Sutra has been conventionally utilized for the multiplication of decimal numbers in comparatively less time. We can use this to multiply two numbers. The bits on the two extremes of the number are multiplied and the result is added to the previous carry. When there are more digits in each number, all the results are added to the previous carry. The process of this scripture is discussed below with the help of an example.
Let us consider two decimal numbers: 234 and 159.

- Initially, the least significant digits of both numbers are multiplied and the result is 36.
- Next, we have cross-wise multiplication of the first digit of the first number with the second digit of the second number. Another cross-wise multiplication of the first number’s second digit with the second number’s first digit. Finally, we add those two products.
- Cross-wise multiplication of the first number’s least significant digit with the second number’s most significant digit and vice-versa. Multiplication of the middle digits gives a total of three products which are finally added.
- Without considering the least significant digits of the two numbers, we perform cross-wise multiplication and addition.
- Vertical multiplication of most significant digits of the two numbers.

For all the steps, with the exception of the first step, each compartment needs to have only one digit. If not, then the carry is forwarded to the initial digits in the previous compartment.

III. SPST ADDER

An adder, either a Full Adder (FA) or a Half Adder (HA), is a combinational circuit used to calculate the sum of three or two inputs. In a parallel adder, FAs are connected in series to produce an n-bit adder. In each section, the sum and carry are not computed until the input carry arises, slowing down the summation process. To avoid delay issues, the Carry Look Ahead (CLA) adder has been suggested. To improve the speed of CLA, Spurious Power Suppression Technique (SPST) is applied. This can reduce the dissipated power of combinational VLSI for DSP purposes. The SPST consists of two parts, the most significant part and the least significant part. It disables the MSP when it doesn’t modify the results saving power [16-20]. The SPST adder/subtractor is classified into two parts, MSP and LSP. The MSP of the first adder/subtractor is modified to include:

- detection unit,
- data controlling unit,
- sign extension unit, and
- circuits for calculating Cin and Cout signals.

In SPST, the detection unit is used to detect if a transition in data bits of the result will occur in the circuits (adders or multipliers) or not. The detection logic unit works as follows:

- When the detection unit disables the MSP, then the output of the MSP are re-compensated so that the time saved for skipping computations will cancel out the delay of the detection logic circuit.
- When it enables the MSP, then the MSP will wait for acknowledgement of the detection unit to enable the data latches. So, the delay of the detection unit will contribute to the entire circuit’s delay.
- When the detection unit stays in its decision irrespective enabling or disabling of the MSP, then the delay of detection unit is insignificant.

There is a data asserting control unit realized by using the registers to further remove any unwanted spurious signals of the arithmetic unit whenever the latched portion has been turned on. This asserting control obviously brings power reduction [16-20].

The 16-bit addition using the SPST process is interpreted in 5 cases. The first case is the transient state, in which spurious transitions of carry signals occur in MSP although the result of the MSP is not altered. The second and third cases illustrate the situation of a negative operand adding to a positive operand without and with carry from LSP, respectively. The fourth and fifth cases demonstrate the condition of two negative operand additions without and with carry-in from LSP, respectively. In all the above cases, the results of the MSP are predictable, so the computations in the MSP aren’t required and can be neglected. Removing those spurious computations will not only save the power consumed by the SPST adder or subtractor but also reduce the noises which are going to affect the subsequent arithmetic circuits.
IV. THE PROPOSED MULTIPLIER

Our main goal is to differentiate the performances of the Vedic multiplier with and without using the SPST adder by comparing both models. To improve the Vedic multiplier’s efficiency, the SPST adder is introduced at intermediate stages to add the sub module outputs. Finally, performance analysis is conducted with respect to area and power.

To develop 16-bit Vedic multiplication, four 8×8 multipliers are required. In the second stage, 16-bit SPST adders are used to add intermediate partial products.

Both multipliers models were simulated and synthesized using Xilinx Vivado tool. Two 16-bit Vedic multipliers, one with RCA and one with SPST, were programmed using HDL and their performance was compared with respect to area and power.

V. SIMULATION RESULTS

All the blocks were programmed and simulated using Xilinx Vivado tool. After verification, the proposed multiplier was implemented on Nexsys 4 DDR Artix 7 FPGA. The synthesis and simulation results of VM using SPST with respect to main modules and sub modules are shown in Figures 3 to 10.

<table>
<thead>
<tr>
<th>Vedic multiplier Appearance</th>
<th>Slice LUTs</th>
<th>Slice registers</th>
<th>Power (W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using RCA</td>
<td>402</td>
<td>120</td>
<td>33.54</td>
</tr>
<tr>
<td>Using SPST</td>
<td>361</td>
<td>0</td>
<td>25.49</td>
</tr>
</tbody>
</table>

From the synthesized results it is concluded that the Vedic multiplier using SPST logic (Figure 7) requires 361 slice LUTs.
(Figure 8) and 25.49W on-chip power which includes 24.708W dynamic power as shown in Figure 9. The result comparison can be seen in Table I. It can be seen that the SPST-based Vedic multiplier provides better performance in terms of LUTs and delay.

![Vedic multiplier using SPST simulation report.](image1)

**Fig. 7.** Vedic multiplier using SPST simulation report.

<table>
<thead>
<tr>
<th>Site Type</th>
<th>Used</th>
<th>Available</th>
<th>Util%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slice LUTs*</td>
<td>561</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>LUT as Logic</td>
<td>361</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>LUT as Memory</td>
<td>0</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>Slice Register</td>
<td>0</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>Register as Flip Flop</td>
<td>0</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>Register as Latch</td>
<td>0</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>FF Max</td>
<td>0</td>
<td>0</td>
<td>0%</td>
</tr>
</tbody>
</table>

![Vedic multiplier using SPST implementation report.](image2)

**Fig. 8.** Vedic multiplier using SPST implementation report.

![Vedic multiplier using SPST power report.](image3)

**Fig. 9.** Vedic multiplier using SPST power report.

VI. HARDWARE IMPLEMENTATION

The external hardware used for implementation consists of:

- ADC kit
- Slide switches and LEDs
- Jumper wires, 3.3V fixed voltage source (from RPS)
- Nexys 4 DDR

After simulation, the proposed Vedic multiplier using the SPST model was dumped into the FPGA to verify the response. In this work, the proposed model was implemented on Nexys 4 DDR Artix7 FPGA with the help of Xilinx Vivado. For the implementation, the first task is to map the original input and output ports with board I/Os by generating a constraint file. Next, the program can be dumped to the FPGA kit to verify the simulation results practically.

![Implementation of the Vedic multiplier with SPST in Artix7.](image4)

**Fig. 10.** Implementation of the Vedic multiplier with SPST in Artix7.

VII. CONCLUSION

In this paper, a 16-bit Vedic multiplier was designed using SPST adders and was programmed in Verilog. The proposed Vedic multiplier was synthesized in the Artix7 FPGA family using the Xilinx tool and its response in terms of area and delay was observed. This work is mainly focused on developing efficient Vedic multipliers using SPST adders and its performance was compared with the one of the ripple carry adder-based Vedic multiplier. From the implementation results, it is observed that the performance of the Vedic multiplier using SPST is better by 10% in respect of area, while power dissipation is improved by 24% compared to that of Vedic using the RCA adder. After the validation, the SPST-based Vedic multiplier was successfully implemented on Nexyxx 4 DDR Artix FPGA.

REFERENCES


