ELECTRONICS AND ELECTRICAL ENGINEERING
ISSN 1392 – 1215 *2011. No. 3(109)*
ELEKTRONIKA IR ELEKTROTECHNIKA

ELECTRONICS
T 170
ELEKTRONIKA

FPGA based Improved Hardware Implementation of Booth Wallace Multiplier using Handel C

S. A. Shinde, R. K. Kamat
VLSI Laboratory, Department of Electronics, Shivaji University, Kolhapur – 416 004, INDIA, e-mail: rkk_eln@unishivaji.ac.in

Introduction

Amongst the basic arithmetic operations over finite field, multiplication is the one which has received the most attention in the literature. This is obvious as it forms the foundation of most of the public-key cryptosystems, specifically RSA, Diffie-Helleman, EIGamal, the elliptic curve cryptosystems and the digital signal processing (DSP) operations [1]. In addition to the above mentioned applications, Graphics and Process control are two more domains wherein the multiplier performance plays a crucial role. The bottlenecks posed by multiplication in the above mentioned application areas are both temporal and spatial in nature. Therefore it appears that, custom VLSI implementations in the form of Application Specific Integrated Circuits (ASIC) or the DSP processors are the only viable alternatives to address the latency demands of such computationally intensive applications, that too without compromising the spatial aspects inview of the propagation delays. However, even in the FPGA paradigm, the custom multiplication hardware embedded within a reconfigurable array has shown promising results and hence is a preferred choice currently due to the cost effective rapid prototyping design cycles as well as the possibility of concurrency, distributed arithmetic and extensive specialization [2–4]. Accordingly many researchers have implemented variants of the basic 'Shift Addition' method to realize the multipliers in FPGA. Some of the preeminent methods of multiplier implementation in the FPGA paradigm are Scaling Accumulator, Serial by Parallel Booth, Ripple Carry Array, Row Adder Tree, Carry Save Array, Look-Up Table and Partial Product, Computed Partial Product, Constant Multipliers from Adders, KCM multipliers, Booth Recoding and Wallace Trees. Amongst the above mentioned multiplier implementations, the Wallace tree and Booth multipliers stimulate VLSI implementation interests as the former reduce the depth of the adder chain thereby minimizing the time complexity while the later giving improved hardware efficiency due to less number of intended partial products.

In this work we present design and implementation of Booth Wallace Multiplier on Xilinx FPGAs. Incidentally, the literature survey pertaining to the hardware multipliers reveals several reports regarding the Booth recoding combined with the Wallace tree. However the central theme of our implementation of the 'Booth Wallace' multiplier differs from the previously reported ones. We resort to the design of the Booth Wallace Multiplier in the FPGA paradigm using the higher level of abstraction using Handel C, which is a variant of C. The paper also exemplifies making the best of the algorithm by using 'C' based tools without compromising the gate count and timing aspects of the crucial arithmetic building blocks ike multipliers.

The rest of the paper is organized as follows: In the section following introduction we review the prior art and emphasize our approach. This follows by the presentation of the 'Booth Wallace' Multiplier architecture and its implementation in Handel C. The results pertaining to the device utilization and timing aspects for 8x8 multiplier core on Xilinx FPGAs are then given. Finally we draw conclusion regarding its applicability in the last section.

Prior Art

The basic Wallace tree algorithm pioneered by Wallace suffers from the performance speed bottlenecks due to the time required in carry propagation addition. The same is overcome by using the carry save adders (CSAs) to add the numbers in redundant and carry free propagate manner. Furthermore, the crossover point where the Wallace tree is faster depends on the VLSI technology used i.e. whether the design is on an application specific integrated circuit (ASIC) or a field programmable gate array (FPGA). In the ASIC paradigm, the tree structure described by Wallace suffers from irregular interconnections and poses difficulty in generating optimum floor plan and efficient layout. Thus this fastest method of summing the partial products, suffers with the extremely complex wiring.

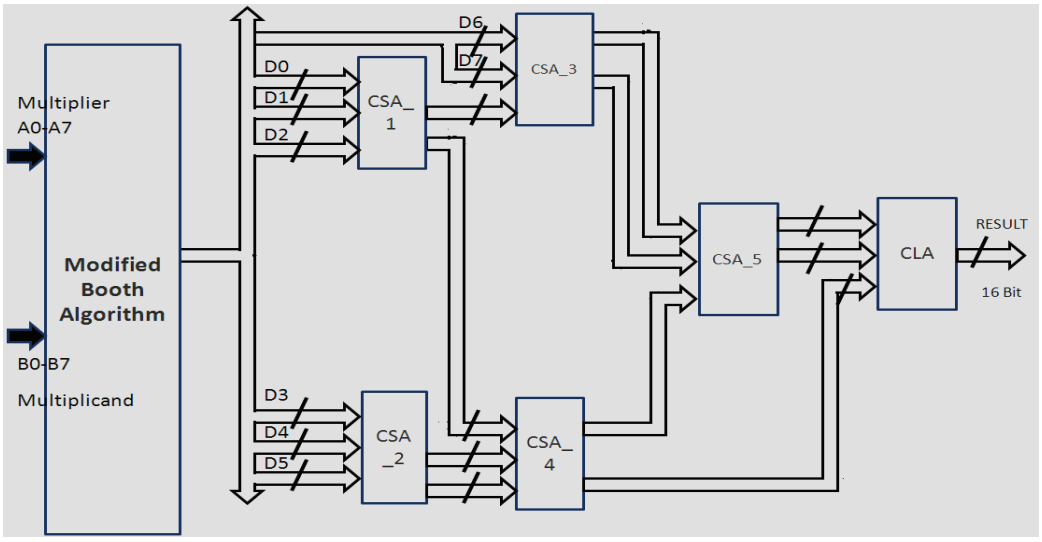

Fig. 1. Architecture of the Booth Wallace Multiplier

```
par
{ if(in1[0]==0)
      { D0=0;}
      else
            {D0=in2;}
  if(in1[1]==0)
            { D1=0;}
      else
            {D1=in2@0;}
if(in1[2]==0)
            {D2=0;}
      else
            {D2=in2@0;}
if(in1[3]==0)
            {D3=0;}
      else
            {D3=in2@0;}
if(in1[4]==0)
            {D4=0;}
      else
            {D4=in2@0;}
if(in1[5]==0)
            {D5=0;}
      else
            {D5=in2@0;}
if(in1[6]==0)
             {D6=0;}
      else
            {D6=in2@0;}
if(in1[7]==0)
            {D7=0;}
      else
             {D7=in2@0;}
```

Fig. 2. Booth Recoding in Handel C

```
par
{sum1=0@D0^0@D1^0@D2;
carry=(0@D0&0@D1)^(0@D0&0@D2)^(0@D1&0
@D2);  }

 par
{carry1=carry[8:0]@0;  sum2=0@D3^0@D4^0@D5;
car=(0@D3&0@D4)^(0@D4&0@D5)^(0@D5&0@
D}

 par

{carry2=car[11:0]@0;sum3=0@D6^0@D7^0@sum1
;
carr=(0@D6&0@D7)^(0@D6&0@sum1)^(0@D7&
0@sum1);}

 par
 {carry3=carr[13:0]@0;     sum4     =(0@sum2)     ^
(0@carry1) ^(0@carry2);
A=(0@sum2&0@carry1)^(0@carry2&0@sum2)     ^
(0@ carry1&0@carry2);}

par
 {carry4=A[11:0]@0;
sum5=0@sum3^0@sum4^0@carry3;
B=(0@sum3&0@carry3)^(0@sum3&0@sum4)^(0@
sum4&0@carry3);} carry5=B[13:0]@0;

par
 {sum6=0@sum5 ^ 0@carry5 ^ 0@carry4;
C=(0@sum5  &  0@carry5)^(0@sum5  &  0@carry4)
^(0@carry4 & 0@carry5); }
```

Fig. 3. Booth recoding with wallace tree in handel C lisitng shows parallelism inculcated using the‘par‘ statement.

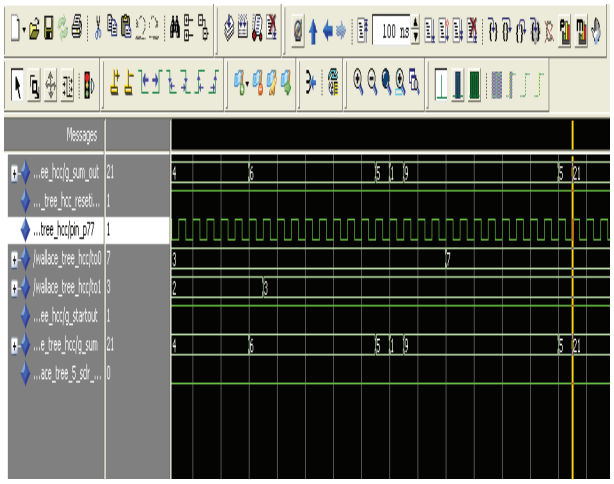

Fig. 4. Simulation of the wallace tree multiplier using ModelSim PE

The lengths of these wires can affect the performance, and the wires themselves take up valuable layout area and therefore the regular binary tree type multipliers seems to be preferred. However the above mentioned pitfalls of the ASIC domain may be overcome by taking advantage of the parallel architecture of the state of art FPGAs. Literature review reveals good amount of work centered on the above theme. Technology mappers that maps RTL networks onto LUT based FPGA's without expanding them for obtaining better results have been reported by many research groups. However, there are components like multipliers, decoders, RAMS etc. which cannot be handled efficiently by general purpose, mappers and require specialized tools called module generators. Many research groups have also reported such module generators for multiplier designs.

However, our approach differs from all those reported in the literature. The main theme of our work is a platform-based approach for obtaining portable FPGA source code to implement the Booth Wallace Multiplier. The platform used for the design is Handel C a basic variant of 'C' language facilitating designing at highest level of abstraction whilst targeting low-level hardware. Our approach treats a high-level system model specified in Handel C and inculcates parallelism in the source code itself which is eventually implemented on the FPGA.

Architecture of the Booth Wallace Multiplier

The architecture of the Booth Wallace Tree Multiplier is shown in figure 1. It comprises of the Booth recoder block cascaded to the Wallace Tree and finally summed by the Carry Look Ahead Adder (CLA). As shown in the figure, recoding of the bits is done at the first stage in parallel at all the bit positions thereby minimizing latency. By using the carry save adders (CSA) the speed is further improved as the carries are saved instead of propagated. The partial sum and carry are then given to the CLA to obtain the multiplication result. The architecture shown in figure 1 is implemented using Handel C. The very rationale behind the choice of the Handel C is its inherent parallel synchronous programming platform where the notion of time is fundamental to its specification. Further it facilitates encoding of the algorithm at behavioral level of abstraction and thus aids in exploiting the advantages of the algorithm at the fullest extent. Further it is packaged as part of a design suite called as DK1 which incorporates a modified version of the GNU preprocessor available in standard Windows development environment with dockable windows and customizable tool bars. The pseudo code (very close to the Handel C version) for the implementation of the Booth recoding is shown in figure 2, while the same cascaded with the Wallace tree is shown in figure 3. Notable feature of this code is instantiation of the intermediate variable in parallel manner so as to facilitate the execution in a single clock cycle.

Table 1. Comparison of device utilization on different FPGAs

Device	Number of Slice Flip Flops	Number of 4 input LUTs	Number of occupied Slices	Net Skew(ns)	Max Delay (ns)
xc3s500e-4fg320	318	213	177	0.083	0.201
xcv100-4fg256 (Vertex)	327	213	182	0.077	0.493
xc5vlx50-3ff324 (Vertex5)	350	222	114	0.114	1.481

Experimental Results

The Booth Wallace Multiplier described above was simulated using ModelSim PE after conversion the Handel C program into VHDL. The simulation reveals correct operation for various 8 bit multiplicand and multiplier combinations. A snapshot of the simulation window is shown in figure 4. In order to compare the spatial aspects of the implementation, VHDL equivalent of the Handel C listing was synthesized using the Xilinx Webpack Version 9.2. Table 1 shows the relative sizes, number of occupied slices and maximum delay on different Xilinx FPGA devices.

We compared our results with the previously reported ones which ranges from the early 1994, paper by Canik and Swartzlander upto the latest implementations of the Booth Wallace Multipliers using different routes. Our solution has shown significant advantages over the above mentioned previous reports in terms of the spatial and temporal specifications.

Conclusions

Significant research is underway to establish the competitive advantage of the Handel C over the conventional hardware descriptor languages in the arena of the real time implementation of multipliers. Our results evidenced that Handel C confirms to be one of the best alternatives not only in taking the best of the algorithmic implementation, but also provides competitive area and time efficiency. The reported implementation has considerable potential in implementing higher order multipliers by its instantiation as a soft IP core.

References

1. **Pierre D. J., Kalach K., Tittley N.** Hardware Complexity of Modular Multiplication and Exponentiation // IEEE Transactions on Computers, 2007. – No. 56(10). – Vol. 1308-319.
2. **Petersen R. J., Hutchings B. L.** An Assessment of the Suitability of FPGA based Systems for use in Digital Signal Processing // Field-Programmable Logic and Applications. - Springer, 1995.
3. **Bencheva N., Kostadinov N., Ruseva Y.** On Teaching Hardware/Software Co-design using FPGA // Electronics and Electrical Engineering. – Kaunas: Technologija, 2010. – No. 6(102). – P. 91–94.
4. **NirmalaDevi M., Mohankumar N., Arumugam S.** Modeling and analysis of Neuro–Genetic Hybrid System on FPGA // Electronics and Electrical Engineering. – Kaunas: Technologija, 2009. – No. 8(96). – P. 69–74.

Received 2010 12 29

S. A. Shinde, R. K. Kamat. FPGA based Improved Hardware Implementation of Booth Wallace Multiplier using Handel C // Electronics and Electrical Engineering. – Kaunas: Technologija, 2011. – No. 3(109). – P. 71–74.

Applications requiring intensive arithmetic operations such as multiplication are exponentially increasing than ever before. The state of art FPGAs are the preferred implementation platforms for implementation of multipliers inspite of the speed and area issues. In this paper we present implementation of Booth Wallace Multiplier on Xilinx FPGAs. Our approach employs design at the higher level of abstraction using Handel C which also inculcates parallelism at the algorithmic level. Ill. 4, bibl. 4, tabl. 1 (in English; abstracts in English and Lithuanian).

S. A. Shinde, R. K. Kamat. Wallace daugiklio, aprašyto Handel C programavimo kalba, diegimas atnaujintose FPGA struktūrose // Elektronika ir elektrotechnika. – Kaunas: Technologija, 2011. – Nr. 3(109). – P. 71–74.

Skaitmeninėje technikoje vis plačiau taikomas daugiafunkcinis režimas. Tai susiję su didėjančiu matematinių operacijų poreikiu. Šiuo metu siūloma tobulinti jau sukurtą pačią struktūrą. Wallace daugiklis, aprašytas Handel C programavimo kalba, diegiamas atnaujintose Xilinx FPGA struktūrose. Pateikiami tokios struktūros pranašumai. Il. 4, bibl. 4, lent. 1 (anglų kalba; santraukos anglų ir lietuvių k.).