Asian Journal of Information Technology 15 (5): 900-907, 2016
ISSN: 1682-3915


# Reconfigurable Processing Element Array Architectures for an Area Efficient Multiplier Architecture

[1]T. Suresh and [2]Z. Brijet
[1]Department of Electronics and Communication Engineering, RMK Engineering College,
Anna University, 601206 Kavaraipettai, Tamil Nadu, India
[2]Department of Electronics and Instrumentation Engineering,
Velammal Engineering College, Anna University, Surapet, 600066 Chennai, Tamil Nadu, India

**Abstract:** Dynamically, reconfigurable architectures have emerged as high performance programmable hardware to execute highly parallel, computationally intensive signal processing functions efficiently. Multipliers are basic functional units in today's digital signal processing and digital image processing algorithms. Multipliers have large area, long latency and consume more power. Many researchers have been trying to design multiplier architectures which offer the design targets such as high speed, low power consumption and less area. Several multiplier architectures and their performance characteristics have been analyzed and compared in this study. Then, a reconfigurable architecture which consists of an array of two different processing elements with suitable interconnects has been proposed for an efficient implementation of signal processing algorithms. From the results, it has been identified that the proposed reconfigurable architecture provides an area efficiency of 37% more than the conventional reconfigurable multiplier architecture.

**Key words:** Adders, digital signal processors, multipliers, reconfigurable architectures, intensive signal

## INTRODUCTION

There is an extensive ongoing research work on reconfigurable architectures. The number of Lookup Tables (LUTs) per cluster, the number of clusters per configurable logic block, the most efficient routing structures and many other parameters are being explored in order to find the best tradeoff between design area, performance and power consumption. These architectures also include special-purpose blocks such as embedded multipliers and Digital Signal Processing (DSP) blocks, which are used to accelerate arithmetic-intensive applications. Their design corresponds to that of an Application-Specific Integrated Circuit (ASIC) as they are specialized for some chosen arithmetic functions and optimized to achieve the highest performance (Jertic and Carreras, 2010). The signal processing algorithms used in wireless communication and image processing applications usually, require a large amount of arithmetic operations with very large operands. Multiplications are fundamental arithmetic operational units which are widely used for practical hardware and thus the optimization is essential in order to achieve high-performance system designs (Miyamoto *et al*., 2011).

Designing various architectures for different applications in the same mobile equipment would result in an intolerable increase in physical size, weight and latency. Therefore, area and latency become another two challenges in the design of mobile equipments. During the last decade, reconfigurable architectures became the mainstream implementation technology for custom computation and embedded system products in such fields as wireless communication, image processing, video processing, multimedia, etc (Xydis *et al*., 2011). These reconfigurable architectures tackle mainly DSP/multimedia issues. Therefore, the design of efficient reconfigurable modules is critical for realizing modern applications. The detailed architecture design of a reconfigurable multiplier was introduced by Koutroumpezis *et al*. (2002). It resolves the design conflict between versatility, area and computation speed and makes it possible to build a feasible and highly flexible processor with multiple multipliers for data intensive applications. Gao *et al*. (2009), the researchers proposed a design for a new reconfigurable flexible multiplier considerably faster than existing reconfigurable multipliers. The focus of research by Haynes *et al*. (1999) was to realize large size signed multipliers using DSP blocks with multigranular embedded

**Corresponding Author:** T. Suresh, Department of Electronics and Communication Engineering, RMK Engineering College, Anna University, 601206 Kavaraipettai, Tamil Nadu, India

signed multipliers in reconfigurable architectures. Serial multipliers are popular for their low area and power (Pekmestzi *et al*., 2001; Almiladi *et al*., 2007) and are more suitable for bit-serial signal processing applications with I/O constraints and on-chip serial-link bus architectures. A new method for computing serial-serial multiplication was introduced in (Doroz *et al*., 2014) by using low complexity asynchronous counters. This counter-based multiplier outperformed many serial-serial and serial-parallel multipliers in speed but its hybrid architecture does carry an area overhead.

The increase in integration density of the on-chip modules causes these modules to become highly congested. To overcome this problem, new techniques are needed to have on-chip high speed multipliers which utilize less number of resources. Compared to the previous standard and the modified architectures of multipliers, a new approach has been proposed in reconfigurable multiplier architecture which has resulted in considerable improvements in terms of area saving by reducing resources required.

## MATERIALS AND METHODS

**Adder block:** Addition is a very crucial operation because it usually involves a carry ripple step which must propagate a carry signal from each bit to its higher bit position. This results in a substantial circuit delay. The adder, therefore which lies in the critical delay path, effectively determines the system's overall speed. On the other hand, the option of reducing area and power consumption of the designed adder, which for many years has been a narrow specialty has recently been gaining prominence. This can be attributed to the emergence and increasing popularity of smaller and more durable mobile computing and communication systems. Three different adder structures will be used for reconfigurable multiplier architectures throughout this research. They are the Ripple Carry Adder (RCA), the Carry Select Adder (CSA) and Carry Look-ahead Adder (CLA).

**Ripple carry adder:** The RCA is the simplest adder circuit. It consists of N full adders with the carry signal propagating from one full adder stage to the next from LSB to MSB. It has many advantages which include low power, low area and a simple layout. The drawback of the ripple carry adder, though, is its slow speed. The delay of the adder is linearly dependent on the bit-width of the adder. The critical path of the ripple carry adder consists of the carry chain from the first full adder to the last.

Table 1: Area requirements of full adder

| Gate requirements for full adder | Area ($\mu m^2$) | Quantity | Total area ($\mu m^2$) |
|---|---|---|---|
| XOR | 9.02 | 2 | 18.04 |
| AND | 8.82 | 3 | 26.46 |
| OR | 8.84 | 2 | 17.68 |
| Net Area ($\mu m^2$) | -- | -- | 62.18 |

Table 2: Area requirements of 8-bit RCA

| Name of the adder | No. of full adder | Total area ($\mu m^2$) |
|---|---|---|
| RCA-8 bit | 8 | 497.44 |

Table 3: Area requirements of 8-Bit CSA

| Hardware requirements | Area ($\mu m^2$) | Quantity | Total area ($\mu m^2$) |
|---|---|---|---|
| Full adder | 62.18 | 16 | 994.88 |
| Multiplexer | 9.81 | 16 | 156.96 |
| Net area ($\mu m^2$) | -- | -- | 1151.84 |

Area requirement of 8 bit RCA is calculated as given in Table 1 and 2. From Table 1 it is observed that the area occupied by full adder is 62.18 $\mu m^2$. So the total area occupied by 8-bit RCA is 497.44 $\mu m^2$ since, 8 full adders are required to design 8-bit RCA. The total delay is 0.96 ns which is due to 8 full adders since the circuit delay of each gate is 0.017 ns.

**Carry select adder:** The CSA employs an intelligent technique to reduce the carry propagation delay (Ramkumar and Kittur, 2012 ; Dorrigiv and Jaberipur, 2014). As the carry signal takes on a value of either a 1 or a 0, if the sum is calculated for both the cases in advance, the carry propagation chain is reduced to just the selection of the correct outputs at each stage using multiplexers. The critical path will now just consist of multiplexers at the output of each bit. The speed of CSA is achieved at the cost of doubling the area because two adders per bit are required, one adder to calculate the sum with a carry-in of 0 and another adder to calculate the sum with a carry-in of 1. In addition, multiplexer is needed for every bit to choose the result based on the actual carry value. As a consequence of this duplication of logic, this design also consumes more power.

Area requirement of 8-bit CSA is calculated as given in Table 3. From Table 3, it is observed that the area occupied by 8-bit CSA is 1151.84 $\mu m^2$. The total delay is 0.624 ns which is due to 8 multiplexers since the circuit delay of each multiplexers is 0.078 ns.

**Carry look-ahead adder:** Carry-ripple delay grows linearly with the size of the input operand for the RCA but these delays can be shortened by generating the carries of each stage in parallel. The CLA adder is theoretically one of the fastest methods for addition. This adder is based on carry generation and propagation logic. The delay time of the CLA architecture exhibits logarithmic dependency on

Table 4: Area requirements of 8-Bit CLA

| Hardware requirements | Area ($\mu m^2$) | Quantities | Total area ($\mu m^2$) |
|---|---|---|---|
| Full adder | 62.18 | 8 | 497.44 |
| AND gate | 8.82 | 8 | 70.56 |
| XOR gate | 9.02 | 8 | 72.16 |
| Net area ($\mu m^2$) | -- | -- | 640.16 |

Table 5: Area and timing results of all synthesized adders

| Adder design | Area in ($\mu m^2$) | Min. clock delay (ns) |
|---|---|---|
| RCA 8-bit | 524.62 | 1.33 |
| CSA 8-bit | 1251.84 | 1.07 |
| CLA 8-bit | 857.02 | 1.09 |

the size of the adder which allows the propagation delay of the carry signal to be minimized. Therefore, the CLA comes in handy for better delay-reduction performance. However, the CLA consumes more area and power because of its large number of logic gates.

The list of hardware and the area allocated by each hardware element of 8-bit CLA is given in Table 4. From Table 4, it is observed that the area occupied by 8-bit CLA is 640.16 $\mu m^2$. The delay time of 8-bit CLA is 0.903ns, which is the logarithmic dependency on the size of the adder.

The RCA, CSA and CLA adder designs have been created in Verilog and synthesized in a 0.18 micron standard CMOS cell library using leonardo spectrum of mentor graphics corp. in order to validate the theoretical results. Area and timing results of synthesized adders are listed in Table 5. These results have included the area and delay due to interconnect lines between hardware elements. As shown by the results given in Table 5 and Fig. 1, the CLA is the fastest scheme for implementing addition, whereas the RCA is the smallest. CSA is the fastest area-efficient adder, as it is slightly smaller than a CLA but significantly faster than an RCA. Since, the goal is to target area efficiency and low power consumption for the proposed architecture, the simple and area efficient RCA structure has been chosen for addition.

**Multiplier block:** Multiplication is an important fundamental arithmetic function currently implemented in signal processing and wireless communication applications. They usually contribute significantly to the time delay and take up a great deal of silicon area in the architecture (Karhe *et al*., 2012; Liu *et al*., 2014). Since multiplication dominates the execution time of most digital signal processing applications, using a high speed multiplier is very desirable (Sriraman and Prabakar, 2012). With an ever-increasing quest for greater computing power on battery operated mobile devices, design emphasis has shifted from optimizing conventional delay time and area size to minimizing power dissipation while still maintaining high

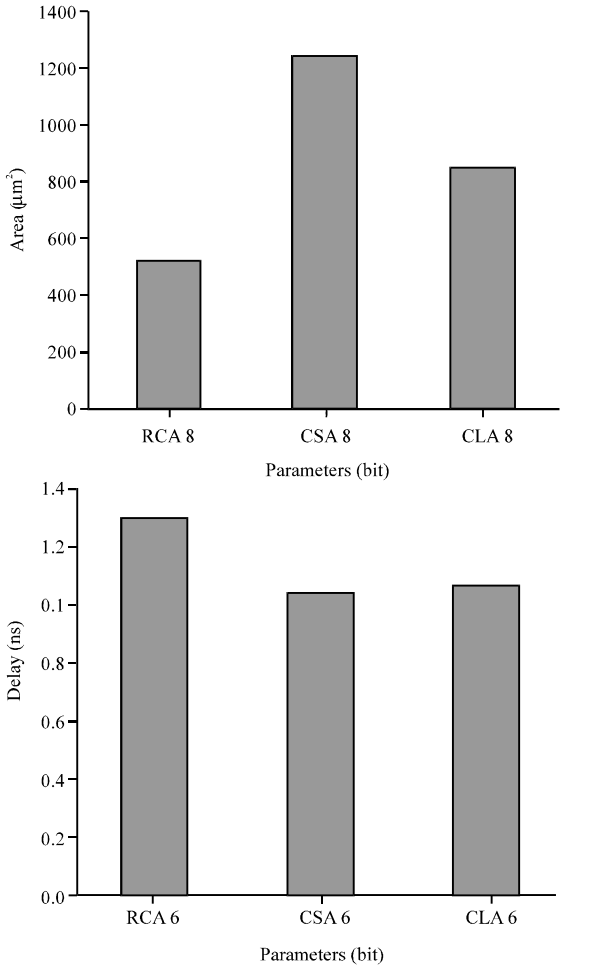


Fig. 1: Area and speed comparisons of 8-bit RCA, CSA and CLA adder designs

performance (Mariammal *et al*., 2013). Therefore, it is needed to select area efficient high speed multiplier which is to be used in the architecture.

Array multiplier is one of the simplest techniques for implementing multiplication. The idea is to add all the N partial products sequentially using N-1 adders. In order to multiply N bit values then in effect it requires N-1 of N-bit adders or N×(N-1) single adder cells (Akhter and Chaturvedi, 2014). Array multipliers are very slow as their critical path is very long. The wallace tree multiplier belongs to a family of multipliers called column compression multipliers. The underlying principle in this family of multipliers is to achieve partial product accumulation by successively reducing the number of bits of information in each column using full adders or half adders. Although, the amount of hardware required to perform this style of multiplication is large, the delay is near optimal. Dadda multipliers belong to the column

Table 6: Area requirements of 8 bit array multiplier

| Hardware requirements | Area ($\mu m^2$) | Quantity | Total area ($\mu m^2$) |
|---|---|---|---|
| Full adder | 62.18 | 49 | 3046.82 |
| Half adder | 17.84 | 7 | 124.88 |
| AND gate | 8.82 | 64 | 564.48 |
| Net area ($\mu m^2$) | -- | -- | 3736.18 |

Table 7: Area requirements of 8 bit wallace tree multiplier

| Hardware requirements | Area ($\mu m^2$) | Quantity | Total area ($\mu m^2$) |
|---|---|---|---|
| Full adder | 62.18 | 38 | 2362.84 |
| Half adder | 17.84 | 15 | 267.6 |
| AND gate | 8.82 | 64 | 564.480 |
| Net area ($\mu m^2$) | -- | -- | 3194.92 |

Table 8: Area requirements of 8 bit dadda multiplier

| Hardware requirements | Area ($\mu m^2$) | Quantity | Total area ($\mu m^2$) |
|---|---|---|---|
| Full adder | 62.18 | 35 | 2176.30 |
| Half adder | 17.84 | 7 | 124.88 |
| AND gate | 8.82 | 64 | 564.48 |
| Net area $\mu m^2$) | -- | -- | 2865.66 |

Table 9: Synthesis results of multiplier structures

| Multiplier design | Area ($\mu m$) | Min. clock delay (ns) |
|---|---|---|
| Array multiplier 8 bit | 4488 | 3.35 |
| Wallace tree 8 bit | 4170 | 2.42 |
| Dadda multiplier 8 bit | 3558 | 2.17 |

compression family of multipliers like the wallace tree multiplier. The dadda multiplier compresses columns differently from the Wallace-tree multiplier. The result of this compression scheme for the partial product matrix is that the delay of the structure is nearly equal to that of the Wallace tree multiplier but with a smaller area.

The list of hardware and the area allocated by each hardware element of 8 bit array multiplier is given in Table 6. From Table 8, it is observed that the area occupied by 8-bit array multiplier is 3736.18 $\mu m^2$. The list of hardware and the area allocated by each hardware element of 8 bit wallace tree multiplier is given in Table 7. From Table 7, it is observed that the area occupied by 8 bit array multiplier is 3194.92 $\mu m^2$. The list of hardware and the area allocated by each hardware element of 8 bit Dadda multiplier is given in Table 8. From Table 8, it is observed that the area occupied by 8 bit Dadda multiplier is 2865.66 $\mu m^2$.

The multiplier structures discussed above have been coded in Verilog and synthesized in a 0.18 micron standard CMOS cell library using Leonardo Spectrum of Mentor Graphics Corp. and the results have been given in Table 9. From the comparison chart given in Fig. 2, it is observed that the Dadda multiplier is faster than the Wallace tree multiplier of the same bit-width, possibly due to the lower fan-outs of intermediate signals. The smaller area of the Dadda multiplier is obtained from a reduced number and size of compressors. So, the Dadda tree outperforms the Wallace scheme in both area and timing. The Dadda multiplier is also faster and smaller than all other multiplier schemes examined above of the same

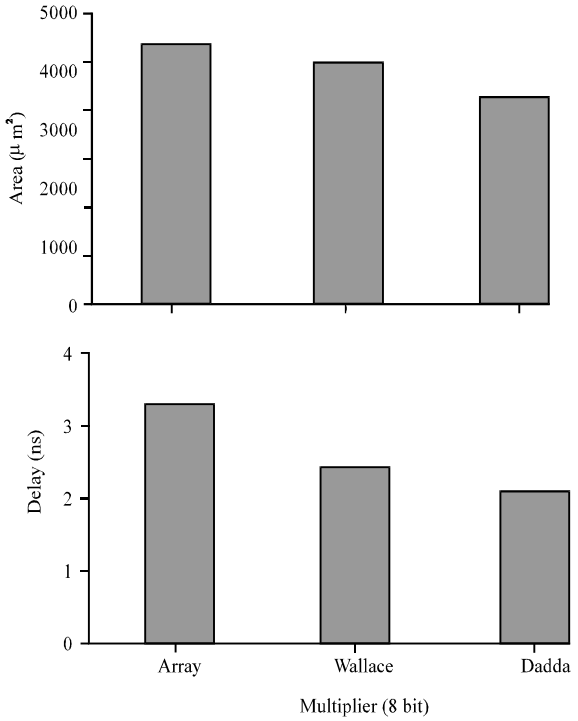


Fig. 2: Area and speed comparisons of 8 bit array, wallace tree and dadda multipliers

bit-width. The smallest multiplier by area is the Dadda multiplier with ripple carry adder which is used in the proposed architecture for multiplication.

**Short-cut multiplier:** Multiplication can be performed by various shortcut approaches. In order to determine the product of two numbers close to 100, the following approach can be used. Let, the product of 97×95 be considered for illustration purpose. Initially, the difference of two numbers with respect to 100 must be determined. Difference of first number is subtracted from the second number and vice versa (since, both these values are always equal). The number obtained is multiplied by 100 (apply left shift operation). The product of both the differences is calculated. These two results are added to get the final result. The same approach can also be used to determine the product of numbers close to 10, 20, 200, 300 and so on. An illustrative representation is shown in Fig. 3. These types of multipliers require less number of functional elements.

Another method to determine the product of a number multiplied by 11, the following approach can be used. For an example, multiplying 384735 by 11 the illustration is given as follows (Fig. 4):

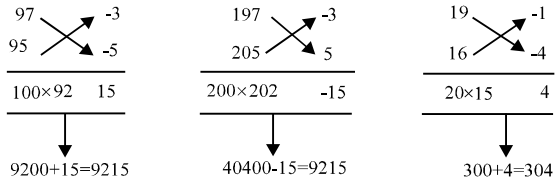


Fig. 3: Multiplication illustration with reference to 100, 200 and 20

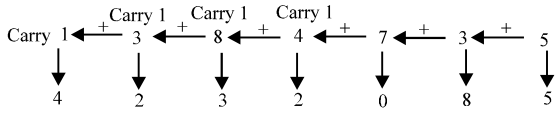


Fig. 4: The method of multiplier illustration

Table 10: Area requirements of 8-bit short-cut multiplier

| Hardware requirements | Area (μm) | Quantity | Total area (μm) |
|---|---|---|---|
| Full adder | 62.18 | 17 | 1057.06 |
| Half adder | 17.84 | 4 | 71.3600 |
| AND gate | 8.82 | 42 | 370.44 |
| Net area (μm) | -- | -- | 1498.86 |

- The ones digit of the multiplier 5 is copied to the temporary result; Result: 5
- Add 3+5=8, so 8 is placed on the left side of the result; Result : 85
- Similarly add 7+3 = 10, so 0 is placed on the left side of the result and carry 1; Result : 085
- Add 4+7 = 11+carry 1 = 12, so 2 is placed on the left side of the result and carry 1; Result : 2085
- Add 8+4 = 12+carry 1 = 13, so 3 is placed on the left side of the result and carry 1; Result : 32085
- Add 3+8 = 11+carry 1 = 12, so 2 is placed on the left side of the result and carry 1; Result :232085
- Add the carry 1 to the highest valued digit in the multiplier, 3+1 = 4 and the final result is obtained. Final result: 4232085

The list of hardware and the area allocated by each hardware element of 8 bit short-cut multiplier is given in Table 10. From Table 10, it is observed that the area occupied by 8-bit short-cut multiplier is 1498.86 $\mu m^2$ that is very less for the product of numbers close to 20, 100, 200, 300 and so on.

**Processing elements and reconfigurable architecture:** Based on the above analysis, two different types of PEs which consist of suitable functional units such as adder/subtractor, multiplier, register and shifter are proposed for the reconfigurable architecture. PE1 is dedicated for multiply and accumulate operation as shown in Fig. 5, PE2 is dedicated for performing add/subtract and

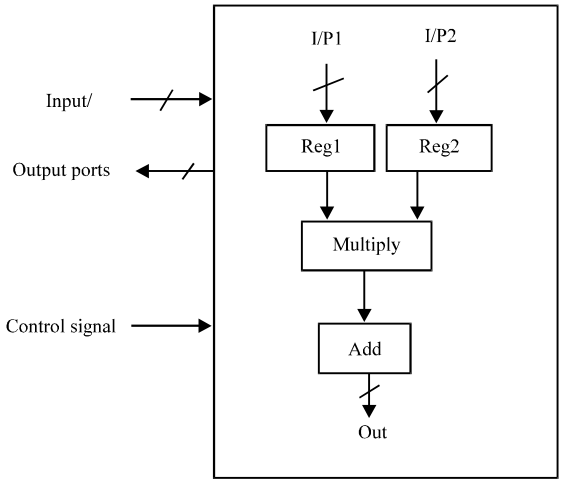


Fig. 5: PE1 for multiply and accumulate operation

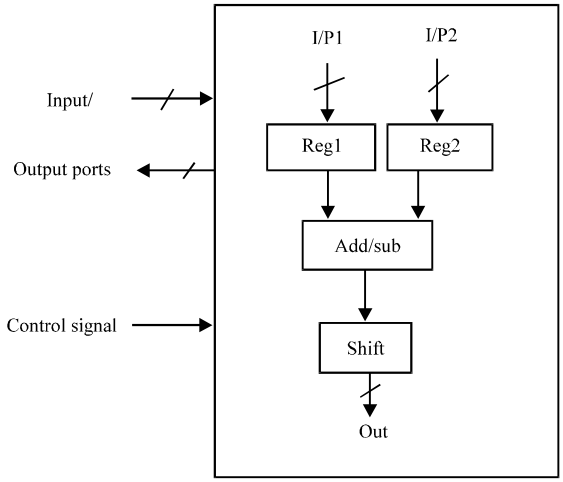


Fig.6: PE2 for add and shift operation

shift operation as shown in Fig. 6 and Processing Elemnent Slice (PES) consists of PE1 and PE2. Figure 7 shows the block diagram of the proposed reconfigurable architecture which consists of an array of PES for mapping application. This architecture is configured dynamically either to Dadda multiplier with ripple carry adder or to short-cut multiplier with ripple carry adder. To process the data for any signal processing functions the incoming data are stored in memory block. The received data are fed to the array of PESs through I/O block. The memory accesses for these operations are coordinated by the control unit. The configuration controller serves as an arbiter for controlling the proposed architecture and surrounding units to specify operation for reconfiguration.

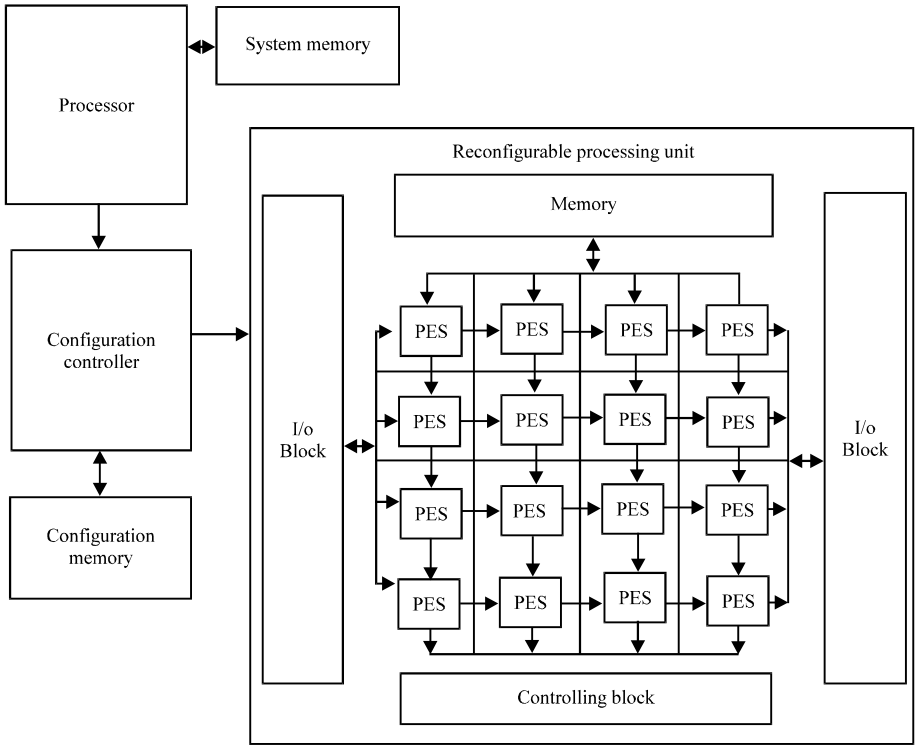


Fig. 7: Block diagram of reconfigurable architecture for mapping application

## RESULTS AND DISCUSSION

The proposed reconfigurable architecture described was simulated for fundamental DFT equation for an *N* point FFT as in (Eq. 1):

$$x(n) = \frac{1}{N\sum_{k=0}^{N-1} X(K) W_N^{-nk}}, n = 0,1,\ldots..,N-1 \quad (1)$$

For Eq. 1 was implemented using MAC unit consists of PE1 and PE2, coded in VHDL and mapped onto a Virtex 5 FPGA device (xc5v1x30ff324) with speed grade (-3) using the tool Xilinx ISE 9.2i and synthesized. The metrics extracted from Xilinx ISE after synthesis and implementation are the followings: the number of used Slice LUTs (Look Up Tables) logic registers and I/O pins.

Table 11-13 give the comparison between conventional MAC and reconfigurable MAC for different bit widths of inputs. Conventional MAC uses more number of LUTS and registers whereas reconfigurable MAC utilizes hardware resources efficiently based on the bit widths of the input bits.

Table 11: Comparison between conventional mac and reconfigurable mac for 16 bit

| Performance factor | Conventional MAC | ReconfigurableMAC |
|---|---|---|
| No of LUTs | 116 out of 12480 | 72 out of 12480 |
| No. of logic registers | 113 out of 12480 | 84 out of 12480 |
| No. of I/O pins | 73 out of 343 | 47 out of 343 |

Table 12: Comparison between conventional mac and reconfigurable mac for 32 bit

| Performance factor | Conventional MAC | ReconfigurableMAC |
|---|---|---|
| No of LUTs | 256 out of 12480 | 162 out of 12480 |
| No. of logic registers | 243 out of 12480 | 186 out of 12480 |
| No. of I/O pins | 143 out of 343 | 107 out of 343 |

Table 13: Comparison between conventional mac and reconfigurable mac for 64 bit

| Performance factor | Conventional MAC | ReconfigurableMAC |
|---|---|---|
| No of LUTs | 529 out of 12480 | 293 out of 12480 |
| No. of logic registers | 517 out of 12480 | 276 out of 12480 |
| No. of I/O pins | 261 out of 343 | 181 out of 343 |

Figure 8-10 describe the hardware utilization in terms of no. of LUTs, dedicated logic registers and I/O pins of conventional MAC compared with reconfigurable MAC for different bit widths of input operands. From these graphs it is found that Reconfigurable MAC utilizes 37% less hardware resources than the conventional MAC.

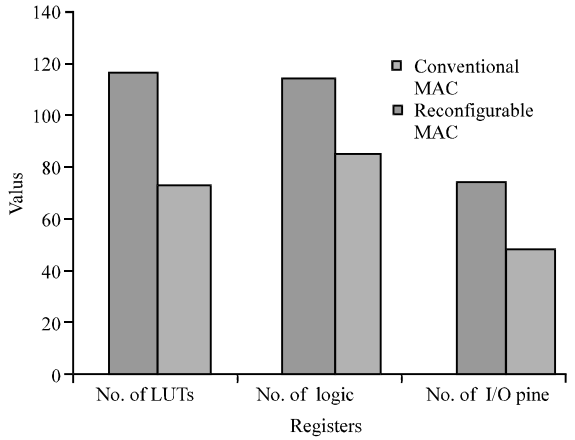


Fig. 8: Comparison of hardware resources between Conventional MAC and Reconfigurable MAC for 16 bit data

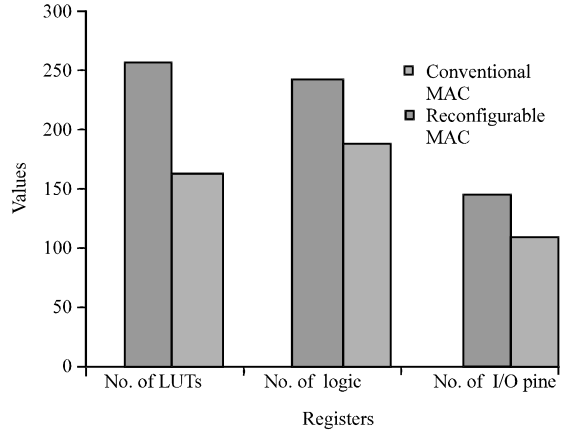


Fig. 9: Comparison of hardware resources between Conventional MAC and reconfigurable MAC for 32 bit data

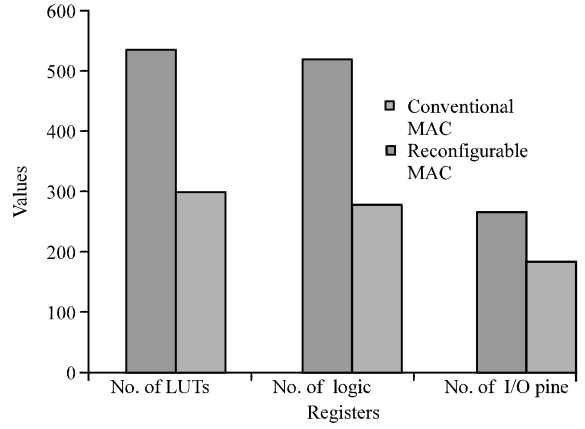


Fig. 10: Comparison of hardware resources between Conventional MAC and reconfigurable MAC for 64 bit data

## CONCLUSION

The different types of adder and multiplier structures have been compared and analyzed based on the area in terms of device utilization. From the comparison results the suitable adders and multipliers have been selected for the signal processing applications and applied in the proposed reconfigurable architecture. Then the reconfigurable architecture described was simulated for fundamental DFT operation using MAC unit, utilizing processing elements PE1 and PE2. The proposed reconfigurable multiplier architecture has been compared with the conventional architecture in terms of the hardware utilization for various bit widths. It has been proved that the proposed reconfigurable architecture with the suitable multiplier is more efficient than the conventional architecture for implementing any signal processing algorithms.

## REFERENCES

Akhter, S. and S. Chaturvedi, 2014. Hdl based implementation of NxN bit serial multiplier. Proceedings of the International Conference on Signal Processing and Integrated Networks, February 20-21, 2014, India, pp: 470-474.

Almiladi, A., M.K. Ibrahim, M. Al Akidi and A. Aggoun, 2007. High-performance scalable bidirectional mixed radix-$2^n$ serial-serial multipliers. IET Comput. Digital Techniques, 1: 632-639.

Doroz, Y., E. Ozturk and B. Sunar, 2014. A million-bit multiplier architecture for fully homomorphic encryption. Microprocessors Microsyst., 38: 766-775.

Dorrigiv, M. and G. Jaberipur, 2014. Low area/power decimal addition with carry-select correction and carry-select sum-digits. Integration VLSI J., 47: 443-451.

Gao, S., D. Al-Khalili and N. Chabini, 2009. Efficient scheme for implementing large size signed multipliers using multigranular embedded DSP blocks in FPGAs. Int. J. Reconfigurable Comput., Vol. 2009. 10.1155/2009/145130

Haynes, S.D., A.B. Ferrari and P.Y. Cheung, 1999. Flexible reconfigurable multiplier blocks suitable for enhancing the architecture of FPGAs. Proceedings of the IEEE Custom Integrated Circuits, May 16-19, 1999, San Diego, CA., pp: 191-194.

Jevtic, R. and C. Carreras, 2010. Power estimation of embedded multiplier blocks in FPGAs. IEEE Trans. Very Large Scale Integration (VLSI) Syst., 18: 835-839.

Kanhe, A., S.K. Das and A.K. Singh, 2012. Design and implementation of floating point multiplier based on Vedic multiplication technique. Proceedings of the International Conference on Communication, Information and Computing Technology, October 2012, Mumbai, India, pp: 1-4.

Koutroumpezis, G., K. Tatas, D. Soudris, S. Blionas, K. Masselos and A. Thanailakis, 2002. Architecture Design of a Reconfigurable Multiplier for Flexible Coarse-Grain Implementations. In: Field-Programmable Logic and Applications: Reconfigurable Computing is Going Mainstream, Glesner, M., P. Zipf and M. Renovell (Eds.). Springer, New York, pp: 1027-1036.

Liu, C., J. Han and F. Lombardi, 2014. A low-power, high-performance approximate multiplier with configurable partial error recovery. Proceedings of the Conference on Design, Automation and Test in Europe, March 24-28, 2014, Dresden, Germany -.

Mariammal, K., V. Rani, S.P. Joy and T. Kohila, 2013. Area efficient high speed low power multiplier architecture for multirate filter design. Proceedings of the International Conference on Emerging Trends in Computing, Communication and Nanotechnology, March 25-26, 2013, Tirunelveli, pp: 109-116.

Miyamoto, A., N. Homma, T. Aoki and A. Satoh, 2011. Systematic design of RSA processors based on high-radix Montgomery multipliers. IEEE Trans. Very Large Scale Integration (VLSI) Syst., 19: 1136-1146.

Pekmestzi, K.Z., P. Kalivas and N. Moshopoulos, 2001. Long unsigned number systolic serial multipliers and squarers. IEEE Trans. Circ. Syst. II: Analog Digital Signal Process., 48: 316-321.

Ramkumar, B. and H.M. Kittur, 2012. Low-power and area-efficient carry select adder. IEEE Trans. Very Large Scale Integration (VLSI) Syst., 20: 371-375.

Sriraman, L. and T.N. Prabakar, 2012. Design and implementation of two variable multiplier using KCM and Vedic mathematics. Proceedings of the 1st International Conference on Recent Advances in Information Technology, March 15-17, 2012, Dhanbad, pp: 782-787.

Xydis, S., G. Economakos, D. Soudris and K. Pekmestzi, 2011. High performance and area efficient flexible DSP datapath synthesis. IEEE Trans. Very Large Scale Integration (VLSI) Syst., 19: 429-442.