DOI 10.29042/2019-5269-5274

Helix Vol. 9 (5): 5269- 5274

A Fast and Optimized Architecture to Perform Multi-Bit Permutation Operation

[*1]Sushma Wadar, [2]D S Bormane

[*1]Research Scholar, E&TC Department, AISSMSIOIT, Pune, MS-India

[2]Principal, AISSMSCOE, Pune, MS-India

Email: [1]sushma97in@yahoo.co.in, [2]principal@aissmscoe.com

Received: 13th September 2019, Accepted: 30th September 2019, Published: 31st October 2019

Abstract

The advanced bit processing operations implemented in the microprocessors and microcontrollers very inefficient. Normally programming techniques are used to emulate the complex bit-related operations. The bit manipulation functions are every now and then required in the areas that are eventually becoming very important. This paper is proposing a techniques which can directly support these bit operations in the form of multimedia shifter unit that can implement standard shifter operations in microprocessors and controllers. The design of the proposed shifter unit is based on the butterfly and inverse butterfly circuits. We show how the proposed design for new shiftes can implement the standard multi-bit scatter and deposit functions found in some processors. The technique proposed in this paper for performing the two operations is based on only Mux. The design of Shifter-Permute functional unit is very challenging work towards its power consumption, speed and area. We have implemented 8-bit Shift-Permute functional unit for bit manipulation and have analyzed the proposed design with the existing design in terms of power consumption, speed and area. Here the circuits are implemented and analyzed by using VHDL and is synthesized by using Xilinx ISE and the targeted device used is Vertex 4 FPGA xc4vlx15-12-sf363 and the same is reflected in the mathematical model purposed for each circuit.

Keywords

Control Unit, Data Reversal, Deposit, Extract, Multiplexer, VHDL.

Introduction

Todays general purpose processors are designed with special instructions for multimedia applications [1]. They are provided with larger sets of multimedia instructions as compared to the earlier generation processors. Consequently, providing efficient multimedia hardware has become an important design task.[12] The multi-bit scatter and gather operations for microprocessors and microcontrollers have not been considered and implemented thoroughly as integer and floating-point arithmetic and data transfer operations. The design of the microprocessors is basically around the processing of words. This is the main reason that bit-level operations are typically not as well supported by current word-oriented microprocessors and microcontrollers.[11] AND, OR, XOR and NOT are the basic bit-oriented logical operations implemented by the Arithmetic Logic Unit (ALU), which is a very important functional unit of a controller or a processor. The very regular operations like shift and rotate where all bits in an operand change their place by the same value, are typically supported by a separate shifter functional unit.[3] [4] [9]

The emerging applications, like biometrics, imaging and cryptography need advanced multi-bit manipulation operations. These bit-manipulation operations can be implemented in a single circuit using only multiplexers or demultiplexers or circuit including both.

Parallel scatter operation can be performed only with the butterfly network and that parallel gather operation can be performed only with the inverse butterfly network.[2][11]

1. Parallel Deposit [11]

This design circuit is explained by Yedidya[11]. The structure of the butterfly network is shown in Figure 1. The rightmost bits from the source register are scattered in the destination register according to a mask bits in the mask register. The i-bit network consists of log(i) stages. Each stage is designed using i/2 two-input multiplexer, for a total of i×log(i) multiplexers as shown in figure 1. In the n^{th} stage, the paired input bits to a switch are i/2n positions apart for the butterfly network and 2n–1 positions apart for the inverse butterfly network. A switch either passes through or swaps its inputs based on the value of a control bit. Thus, the operation requires i/2 × log(i) control bits.

Control bits for 8-bit input, for each stage are calculated as follows:

1st Stage:

- The mask bits are divided in two parts, L and R, each 4-bits.
- Number of 1's in the R are counted i.e. from I_3 to I_0= count.
- Left rotate and complement (LROTC) of '0000' is done depending on the value of count.
- This generates the control bits= S_{03} S_{02} S_{01} S_{00}

 Helix E-ISSN: 2319-5592; P-ISSN: 2277-3495

2nd Stage:

- The mask bits are further divided in LL, LR, RL and RR.
- Count the number of 1's from LR i.e. from I_5 to I_0 = count.
- Left rotate and complement (LROTC) of '00' is done depending on the value of count.
- This generates the control bits= S_{13} S_{12}
- Count the number. of 1's from RR i.e. from I1to I0 = count.
- Left rotate and complement (LROTC) of '00' is done depending on the value of count.
- This generates the control bits= S_{11} S_{10}

3rd Stage:

- The mask bits are further divided in LLL, LLR, LRL,LRR, RLL, RLR, RRL and RRR.
- Count the number of 1's from LLR i.e. from I_6 to I_0 = count.
- Left rotate and complement (LROTC) of '0' is done depending on the value of count.
- This generates the control bit= S_{23}
- Count the no. of 1's from LRR i.e. from I_4 to I_0 = count.
- Left rotate and complement (LROTC) of '0' is done based on the value of count.
- This generates the control bit= S_{22}
- Count the no. of 1's from RLR i.e. from I2 to I0 = count.
- Left rotate and complement (LROTC) of '0' is done depending on the value of count.
- This generates the control bit= S21
- Count the no. of 1's at LRR i.e. I0 = count.
- Left rotate and complement (LROTC) of '0' is done depending on the value of count. This generates the control bit= S20

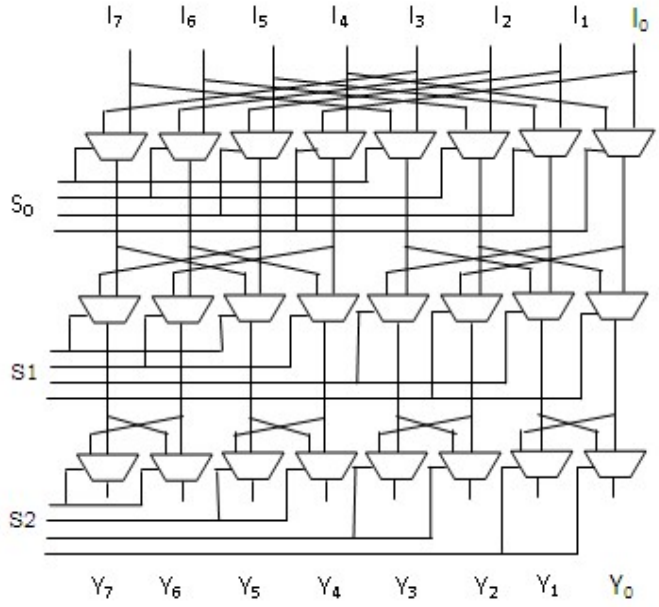

Figure 1: Parallel Deposit

2. Parallel Gather[11]

This operation collects the scattered bits from the location given by the mask register and placeses them continuously in the destination register.The parallel extract can be implemented by the inverse butterfly network. The inverse butterfly network is decomposed into even and odd sub-networks, in contrast to the butterfly network which is decomposed into right and left sub-networks. Control bits will be generated in the same way as done for the parallel deposit. It works similarly as parallel deposit but in reverse order. This circuit reduces the hardware required for performing the operations.

2.1. Algorithm:

1st Stage:

- The mask bits and input data is given to the registers
- If control bit is '0' then 4 bit shift input will be passed else the input data will be passed as it is.
- The input data is divided as L and R of 4 bits each. Counting of the no. of 1's in the R is done and saved in count.

  Helix E-ISSN: 2319-5592; P-ISSN: 2277-3495

- Then left rotate and complement of '0000' is done depending on the value of count. Hence this becomes the control bits S03 S02 S01 S00.

2nd Stage

- If control bit is '0' then 2 bit shift input will be passed else the input data will be passed as it is.
- The input data is divided as LL, LR, RL and RR of 2 bits each.
- Count the no. of 1's from LR and save in count.
- Then left rotate and complement of '00' is done depending on the value of count.
- Count the no. of 1's from RR and save in count i.e. count = 0 and the control bitsS01 S00.
- The data will move accordingly giving the output of the second stage.

3rd Stage

- Now for the last stage the input data is divided as LLL, LLR ,LRL, LRR, RLL ,RLR ,RRL and RRR of 1 bit each.
- Similarly count the no. of 1's and saves in the count and this process will continue four times. Then left rotate and complement of '0' is done depending on the value of count.
- The data will move accordingly giving the output of the last stage.

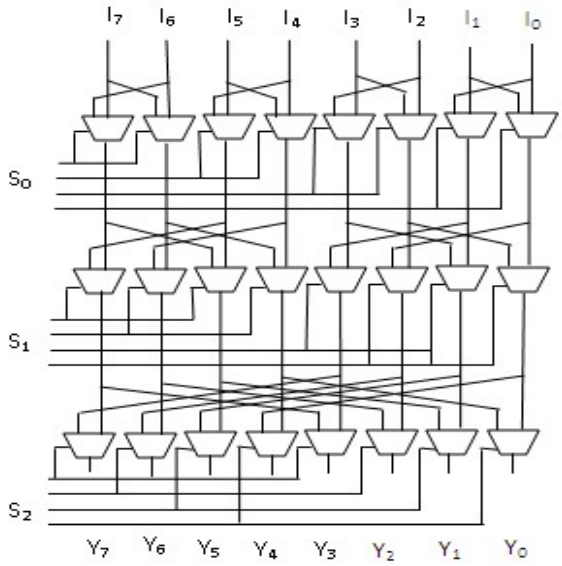

Figure 2: Parallel Extract

3. Data Reversal Circuit [7]

An array of log(n) multiplexers are used for data reversal. Data reversal circuit is shown in Figure 3. In the earlier mentioned inverse butterfly model, only parallel extract operation can be performed and for parallel deposit we have to shift to the butterfly model. This design involves data reversal technique combined with inverse butterfly technique to perform both the extract and deposit operations. [4]

$S1_7 = I_7(\overline{L/\bar{R}}) + I_0(L/\overline{R})$

$S1_6 = I_3(\overline{L/\bar{R}}) + I_4(L/\overline{R})$

$S1_5 = I_6(\overline{L/\bar{R}}) + I_1(L/\overline{R})$

$S1_4 = I_2(\overline{L/\bar{R}}) + I_5(L/\overline{R})$

$S1_3 = I_5(\overline{L/\bar{R}}) + I_2(L/\overline{R})$

$S1_2 = I_1(\overline{L/\bar{R}}) + I_6(L/\overline{R})$

$S1_1 = I_4(\overline{L/\bar{R}}) + I_0(L/\overline{R})$

$S1_0 = I_0(\overline{L/\bar{R}}) + I_7(L/\overline{R})$

The ectract operation will be performed when the $e/\overline{d}$ =1, the data will not be reversed and passed directly to the inverse butterfly circuit and output is also obtained without reversing the data.

To perform parallel deposit operation, ($e/\overline{d}$ = 0) the data input is reversed initially and then given to the inversr butterfly circuit. [8]

  Helix E-ISSN: 2319-5592; P-ISSN: 2277-3495

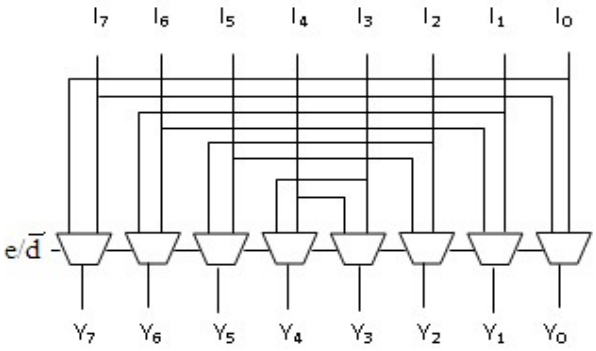

Figure 3: Data Reversal Circuit

Proposed Method:

Multimedia Shifter Designs : Extract and Deposit Unit using Data Reversal

The method proposed in this paper for performining the multi-bit scatter and gather operation uses single inverse butterfly circuit.

Instead of using two different circuits for gather and scatter operations with separate control bits and data paths i.e. butterfly circuit for multi-bit scatter operation and inverse butterfly circuit for gather operation, single design with data reversal circuit can be used.

The block diagram of the circuit used for performing the extract and deposit operation is shown in figure 4. The multi-bit extract and deposit mechanism is as explained in the ibfy model. Control bits are generated using control bit generator circuit. The input to the control bit generator circuit is the data provided in the mask register.

The extract operation and deposit operation can be performed using single circuit designed only with the MUX which uses one control unit. When the enable signal i.e $e/\overline{d}$ =1 is high, it performs the parallel scatter operation and when enable signal is low it performs the parallel gather operation.

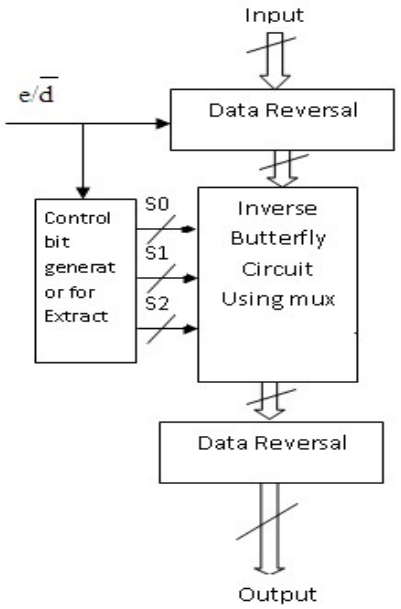

Figure 4: Proposed Model for Extract and Deposit Using Data Reversal

Parameters	Area	Slices	Delay	ADP
Parallel Deposit Curcuit with Cotrol Unit	41	23	8.013	328.533
Parallel Extract Curcuit with Cotrol Unit	41	23	8.03	329.23
Proposed Parallel Extract and Deposit Curcuit with Data Reversal Circuit	57	32	8.781	500.517

Table 1 : Comparison of Proposed Model

  Helix E-ISSN: 2319-5592; P-ISSN: 2277-3495

Unit for Area: Number of LUT; Unit for Delay: ns;
Delay = Latency
ADP: Area-Delay Product = Area x Delay

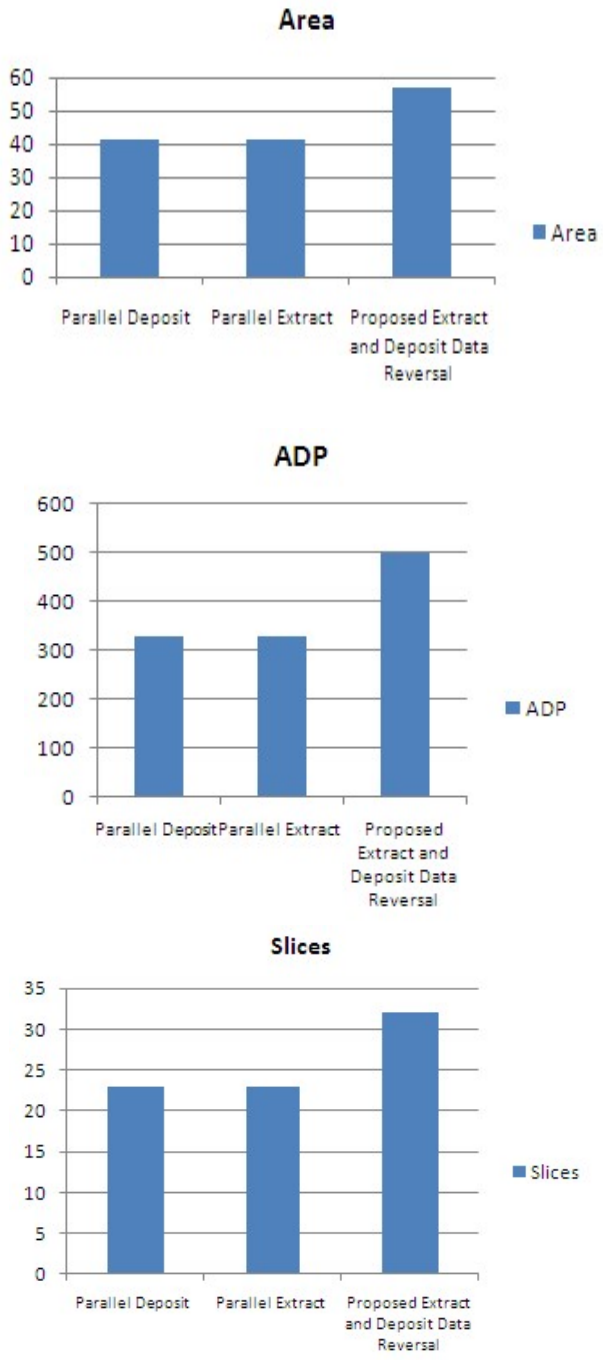

Fig 5: Comparison of the Results of the Three Circuits, Implemented in Same Environment, in terms of Performance Parameters such as Area (a), Propogation Delay (b) Area Delay Product (c) as a Function of Same Number of Input Bits.

Conclusion

It is expected to mention here, that we have referred the implementation methodology from reference number[11] and implemented it in the same technological environments, Vivado 2015.4 and the performance parameters such as area in terms of LUTs, delay in nanoseconds and area delay product are compared. The extract and deposit functions can be implemented using single circuit design only. The results indicate that

  Helix E-ISSN: 2319-5592; P-ISSN: 2277-3495

proposed circuit uses less LUTs for both the operations as compared to the separate circuits for each operation. The delay is slightly increased by 0.75ns, due to data reversal circuit which is added in the circuit which is negligible. This new circuit designed and implemented using dual data path proves out to be efficient functional unit.

References

1] Woo-Kyeong Jeong and Yong-Surk Lee, 'A Universal Shifter with Packed Data Formats', International Journal for Electronics and Communications, (AE¨U) 57 (2003) No. 6, Pgs. 420−422

2] Neil Burgess, Department of Electrical and Electronic Engineering, University of Bristol, 'Assessment of Butterfly Network VLSI Shifter Circuit', 978-1-4244-9721-8/10/ pp 92-96, Asilomar 2010 ©2010 IEEE

3] Steven Huntzicker, Michael Dayringer, Justin Soprano, Anthony Weerasinghe, David Money Harris, and Dinesh Patil1 'Energy-Delay Tradeoffs in 32-bit Static Shifter Designs', , 978-1-4244-2658-4/08/ pp 626-632 ©2008 IEEE.

4] Ramin Rafati, Sied Mehdi Fakhraie, Member, IEEE, and Kenneth Carless Smith, Life Fellow, IEEE, 'A 16-Bit Barrel-Shifter Implemented in Data-Driven Dynamic Logic (D3L)', IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS, VOL. 53, NO. 10, pp 2194- 2202 OCTOBER 2006

5] Sonia Gonzalez-Navarro, Javier Hormigo, Michael J. Schulte, 'A study of decimal left shifters for binary numbers', Contents lists available at SciVerse Science Direct, Information and Computation, 216 (2012) Pgs. 47–56, www.elsevier.com/locate/yinco

6] S. R. Nassif, "Process Variability at the 65nm Node and Beyond," in IEEE Custom Integrated Circuits Conference, Sept. 2008.

7] S. Kotiyal, H. Thapliyal, and N. Ranganathan "Design of a reversible bidirectional barrel shifter," in Proceedings of the 11th IEEE International Conference on Nanotechnology, Portland, Oregon, USA, Aug 2011, pp. 463-468.

8] S. Kotiyal, H. Thapliyal, and N. Ranganathan, "Design of a ternary barrel shifter using multiple-valued reversible logic," in Proceedings of the 10th IEEE International Conference on Nanotechnology, Seoul, Korea, Aug. 2010, pp. 1104–1108.

9] R. Pereira, J. A. Mitchell, and J. M. Solana, \Fully Pipelined TSPC Barrel Shifter for High-speed Applications," IEEE Journal of Solid State Circuits, vol. 30, pp. 686{690, June 1995.

10] R. Ramadoss, "A New Breed of Power-Aware Hybrid Shifters," in IEEE International SOC Conference, pp. 143-146, 2005

11] Yedidya Hilewitz and Ruby Lee, "Fast bit Compression and Expansion With Parallel Extract and Parallel Deposit Instructions", J Sign Process Systems, Springer, 2012.

12] Claudio Brunelli, 'Design of Hardware Accelerators for Embedded Multimedia Applications', 2009

  Helix E-ISSN: 2319-5592; P-ISSN: 2277-3495