A NOVEL ARCHITECTURE FOR MULTI-BIT SHIFT AND ROTATE OPERATION

Sushma Wadar¹, D S Bormane², S C Patil³, Avinash Patil⁴

¹ResearchScholar, E&TC Department, AISSMSIOIT, Pune, MS-India
²Principal, AISSMSCOE, Pune, MS-India,
³Professor, Dept. of Information Tech., Rajarshi Shahu College of Engineering, Pune, MS-India,
⁴Research Scholar, E&TC Department, Rajarshi Shahu College of Engineering, Pune, MS-India

¹sushma97in@yahoo.co.in, ²bdattatraya@yahoo.com, ³shailaja.patil11@gmail.com, ⁴avispatil@yahoo.co.uk

Corresponding Author: Sushma Wadar
Email: sushma97in@yahoo.co.in

https://doi.org/10.26782/jmcms.spl.9/2020.05.00003

Abstract

In the available microprocessors and microcontrollers, the multi-bit operations are implemented with very less efficiency. Generally, these complex bit operations are emulated using programming logic. These bit manipulation operations are frequently required in the applications that are becoming very important. In this paper, we propose two new techniques which can directly support these bit operations in the form of shifter unit that can implement standard shifter operations in microprocessors and controllers. The design of the proposed shifter unit is based on the inverse butterfly circuit. In this paper, we propose two techniques that have shift/rotate and mask circuits which enable the same circuit to perform all types of the standard shift and rotate operations found in some processors. The first technique is using Data reversal method and second using Two’s complement method. The design of Shifter-Permute functional unit is the important and critical task towards optimizing parameters such as speed, area and power consumption. Here we have implemented an 8-bit Shift-rotate functional unit for bit manipulation in the form of two approaches and have analyzed the circuits in terms of speed, area, and power consumption. Here the circuits are implemented and analyzed by using the most popular semi-custom design tool Vivado ISE 2015 and is synthesized by using Artix-7 FPGA and the same is reflected in the mathematical model purposed for each circuit.

Copyright reserved © J. Mech. Cont.& Math. Sci.
Sushma Wadar et al.

Conference on “Emerging Trends in Applied Science, Engineering and Technology” Organized by MDSG Research Group, Malaysia

22
Keywords: Data reversal, Two’s complement, Shifter, Butterfly and Inverse butterfly, Shift/Rotate, Mask.

I. Introduction

The multi-bit operations are generally not implemented efficiently for microprocessors and microcontrollers and not implemented as thoroughly as the arithmetic operations performed on integer and floating-point data and data transfer operations. The design of the microprocessors is basically focused on the processing of bytes of various sizes.[X][XI] This is the main reason that the focus on better implementations of bit-wise operations are not supported that efficiently by current microprocessors and microcontrollers. In this paper, we propose two techniques to implement the shifter unit to perform these basic shifts and rotate operations using a single circuit. The shifts and rotate in which each bit moves in the same relative amount as every other bit in the given word (register), a separate Shifter functional unit is typically used to implement these operations. There are many recent and real time applications, where advanced bit manipulation operations are needed. While these applications can be built from the simpler logical and shift operations, the applications using these advanced bit manipulation operations can significantly speed up if the processors supports more powerful bit manipulation instructions. It has been shown that these operations can be implemented in a single new Permutation functional unit, utilizing two simple data paths: an inverse butterfly circuit and a butterfly circuit [X].

II. Proposed Design

The new design represents an evolution of the shifter from the classical barrel shifter and log shifter to a new multi-bit Shift-Rotate functional unit which can perform both shift operations and rotate operations. The contributions of this paper are as follows:

1. We propose two new bases for the design of shifters, based on the inverse butterfly circuit. These new bases implement a much more powerful set of shift and rotate bit manipulation instructions, previously requiring two existing functional units and a new permutation functional unit.
2. Also a common circuit for determining the control bits is described.
3. A shift/rotate unit makes the circuit work as shifter unit or rotator unit.
4. A mask unit masks the bits depending upon a right or left arithmetic or logical shift.

II.i. Inverse Butterfly Circuit

The Inverse butterfly circuit can be used for right rotate operations only. This circuit also consists of \( n \times \log(n) \) number of 2:1 Mux’s and has \( \log(n) \) stages, where \( n \) is the number of bits in the data. Here we are operating on 8-bit data, therefore 8 \( x \) 3=24 Mux’s are used and it will have 3 stages as shown in Figure 1. The left bit rotation/shifting through the butterfly design takes a high figure of latency due to its
complex design of data path at the first stage itself which is not able to compensate the speed of execution with the next stages of shifting which happens much faster [VII]. Due to this very reason we came up with the solution of making a shifter/rotator for the inverse butterfly design because till the time the first two stages produces the results, the third stage which is much complex in its design of data path gets compensated with the speed of functioning of the previous stages resulting in less latency [II]. The three stages of the inverse butterfly circuit can be explained as follows:

1st stage: Only one control bit is required for data shifting in first stage. Depending upon the control bit the data will be passed to the next stage. If the control bit is ‘0’ the data will be passed as it is, else for ‘1’ the one bit shifted data will be passed. The control bit for stage 1 is generated according to the equation given below, $S_{b0}$ indicates the complement of the control bit.

$$c_{b0}=S_0$$ (1 control bit)

$$S_{7} = I_S S_{b0} + I_I S_0$$
$$S_{6} = I_S S_{b0} + I_I S_0$$
$$S_{5} = I_S S_{b0} + I_I S_0$$
$$S_{4} = I_S S_{b0} + I_I S_0$$
$$S_{3} = I_S S_{b0} + I_I S_0$$
$$S_{2} = I_S S_{b0} + I_I S_0$$
$$S_{1} = I_S S_{b0} + I_I S_0$$

2nd stage: Two control bits are required in the second stage. If the control bit is ‘0’ the data will be passed as it is else for ‘1’ the 2-bit shifted data will be passed, also || sign represents concatenation of bits. The control bits of stage 2 are generated according to the equation given below. The suffix b indicates the complement of the value.

$$c_{b1}= (S_0 \oplus S_I)||S_I$$

$$S_{7} = S_7(S_0 \oplus S_I) + S_7(S_0 \oplus S_I)$$
$$S_{6} = S_6(S_0 \oplus S_I) + S_6(S_0 \oplus S_I)$$
$$S_{5} = S_5(S_0 \oplus S_I) + S_5(S_0 \oplus S_I)$$
$$S_{4} = S_4(S_0 \oplus S_I) + S_4(S_0 \oplus S_I)$$
$$S_{3} = S_3(S_0 \oplus S_I) + S_3(S_0 \oplus S_I)$$
$$S_{2} = S_2(S_0 \oplus S_I) + S_2(S_0 \oplus S_I)$$
$$S_{1} = S_1(S_0 \oplus S_I) + S_1(S_0 \oplus S_I)$$
$$S_{0} = S_0(S_0 \oplus S_I) + S_0(S_0 \oplus S_I)$$
3rd stage: Depending on the control bit i.e. if ‘0’ the data will be passed as it is else for ‘1’ the 4-bit shift data will be passed. 4 control bits will be used in this stage. The control bits of stage 3 are generated according to the equation given below. The suffix b indicates the complement of the value.

\[
\begin{align*}
C_{b2} &= (S_0 + S_1) \text{ xor } S_2 \| (S_1 \text{ xor } S_3) \| (S_0 \text{ xor } S_1) \| S_2 \\
S_{3} &= S_{2}(S_0 + S_1) \text{ xor } S_2 + S_{3}(S_0 + S_1) \text{ xor } S_2 \\
S_{6} &= S_{5}(S_1 \text{ xor } S_2) + S_{6}(S_1 \text{ xor } S_2) \\
S_{3} &= S_{4}(S_0 S_1) \text{ xor } S_2 + S_{5}(S_0 S_1) \text{ xor } S_2 \\
S_{4} &= S_{5}(S_1 S_2) + S_{6}(S_1 \text{ xor } S_2) \\
S_{3} &= S_{4}(S_0 S_1) \text{ xor } S_2 + S_{5}(S_0 S_1) \text{ xor } S_2 \\
S_{5} &= S_{6}(S_1 \text{ xor } S_2) \\
S_{3} &= S_{6}(S_0 S_1) \text{ xor } S_2 + S_{7}(S_0 S_1) \text{ xor } S_2 \\
S_{0} &= S_{6}(S_1 \text{ xor } S_2)
\end{align*}
\]

**Fig. 1**: Inverse Butterfly Circuit

**II.ii. Data Reversal Inverse Butterfly**

In the earlier mentioned inverse butterfly model, only right rotate operation can be performed and for left rotate operation we have to switch to the butterfly model. An array of \( \log(n) \) multiplexers are used for data reversal, circuit is as shown in fig2. Data reversal circuit helps in using the same inverse butterfly circuit for left rotate operation also it reverses the input and output inverse butterfly circuit. [IV]
II.iii. Two’s Complement Inverse Butterfly

For an n-bit data, s-bit right rotate is equivalent to n-s bit left rotate. It was observed that a 2’s complement circuit can be used for performing the left rotate in the ibfy model using select bits, s2 s1 s0. The equations generated for the 2’s complement circuit are as follows:

\[ c_0 = S_0 \]
\[ c_1 = S_1 \cdot (L/R_{b}) + (S_0 \text{ xor } S_1) \cdot (L/R_{b}) \]
\[ c_2 = S_2 \cdot (L/R_{b}) + [(S_0 + S_1) \text{ xor } S_2] \cdot (L/R_{b}) \]

Where, c0, c1 and c2 are the three bit two’s complement of three bit select value s0, s1 and s2.

By adding the two’s complement circuit to the inverse butterfly circuit the extra hardware required in data reversal circuit due to which increase in delay can be compensated. This is proved by the simulation results.

II.iv. Masking Bits

For shifting operation, an extra circuit for masking the data bits is required. Depending upon the select lines (s2 s1 s0) at its input each data output is separately masked with masked bits (f7 – f0) generated by the circuit shown in Figure 3.3.

![Masking circuit for 8-bit data](image)

Following are the equations of the circuit:
II.v. Shifting and Rotating Circuit

As discussed in earlier section there are two types of shifts: arithmetic shift and logical shift. These are combined together in the circuit shown in the Figure 4. In logical shift right, the output bits obtained from the butterfly are ANDed with the respective mask bits ($f$). Similarly for logical shift left, the output bits obtained from the butterfly are ANDed with the reversed mask bits ($\bar{f}$). For example, $Y_7$ will be ANDed with $f_0$ and similarly others will be calculated. In arithmetic shift, the sign bit ($I_7$) of input data is considered for shifting. Here two cases are considered:

When $I_7=0$, the output bits obtained from the inverse butterfly are ANDed with respective mask bits ($f$). Here the mask bits are made of $I_7$ and 1 bits. For example, for 5 bit right shift the mask bits generated will be: $f = I_7 I_7 I_7 I_7 I_7 1 1 1$.

When $I_7=1$, the output bits obtained from the inverse butterfly are ORed with respective mask bits ($\bar{f}$). Here the mask bits are made of $I_7$ and 0 bits. For example, for 5 bit right shift the mask bits generated will be: $f = I_7 I_7 I_7 I_7 I_7 0 0 0$.

\[
\begin{align*}
    f_0 &= 1 \\
    f_1 &= ((\bar{s}_0 + \bar{s}_1) + \bar{s}_2) \\
    f_2 &= (\bar{s}_1 + \bar{s}_2) \\
    f_3 &= ((\bar{s}_0 \times \bar{s}_1) + \bar{s}_2) \\
    f_4 &= \bar{s}_2 \\
    f_5 &= ((\bar{s}_0 + \bar{s}_1) \times \bar{s}_2) \\
    f_6 &= (\bar{s}_1 \times \bar{s}_2) \\
    f_7 &= ((\bar{s}_0 \times \bar{s}_1) \times \bar{s}_2)
\end{align*}
\]

**Fig. 4:** Shift/Rotate unit for one bit

Following are the equations for the shifting and rotating circuit shown in Figure 3.4

\[
\begin{align*}
    AND1 &= (Y_n f_n) \\
    ORI &= (Y_n + f_n) \\
    Mux1 &= (Y_n f_n) I_7 + (Y_n + f_n) \\
    AND2 &= (Y_n f_n I_7)
\end{align*}
\]
$Mux2 = (Y_n \cdot f_n) \cdot L / R + (Y_{n-7} \cdot L) / R$

$Mux3 = (mux1) \cdot \text{arith/logic} + (mux2) \cdot ( \text{arith/logic} )$

$Mux4 = Y_n \cdot \text{rot/sh} + (mux3) \cdot ( \text{rot/sh} )$

II.vi. Shift/Rotate Circuit Using Data Reversal Circuit

In the earlier mentioned inverse butterfly model, only right rotate operation can be performed and for left rotate we have to shift to the butterfly model. This design involves data reversal technique combined with inverse butterfly technique to perform both the right and left rotate operations. For right rotate ($L/R_b = 0$) the data will not be reversed and passed directly through the data reversal circuit to the inverse butterfly circuit and output is also obtained without reversing. For left rotate ($L/R_b = 1$) the input data will be first reversed and then given to the inverse butterfly. Similarly, the output for left rotate is obtained by data reversal. The rotating mechanism is as explained in the inverse butterfly model. Control bits are generated using a control bit generator circuit. With select lines (s2 s1 s0) as inputs to the circuit, the respective control bits can be generated and given to the data reversal inverse butterfly circuit as shown in Figure 3.6 and following are its equations:

$S_{in} = I_i(L/R_b) + I_{7-n}(L/R_b)$

$i = \text{stage} \text{ and } n = \text{mux position}$

Using the shift/rotate circuit along with masking circuit, the same hardware can be used to perform all the possible shifts and rotates. The figure below shows the block diagram of this proposed hardware. Control bits are generated using control bit generator circuit as shown in Figure 3.4. With select lines (s2 s1 s0) as inputs to the circuit the respective control bits can be generated and given to the data reversal inverse butterfly circuit as shown in Figure.

![Shift/Rotate using Data Reversal Method](image)

Fig. 5: Shift/Rotate using Data Reversal Method

Depending upon the shift or rotate operation, mask bits will be generated and will be mapped with Shift/Rotate circuit. The above diagram shows the working of the shifter unit, for multi bit shift or rotates operations in both the directions.
II.vii. Shift /Rotate Circuit Using 2’s Complement Inverse Butterfly

For an n-bit data, b-bit right rotate is equivalent to n-b bit left rotate. It was observed that a 2’s complement circuit can be used for performing the left rotate in the ibfy model using select bits (s2 s1 s0). The shift/rotate circuit along with mask circuit is used for shifting operation as shown in figure. The design reduces the hardware requirement for the implementation of the shifter unit and also enhances the speed of operation.

Fig. 6: Shift/Rotate using 2’s Complement Method

III. Result and Discussion

Simulated result: The simulation results of Shift/Rotate Data Reversal inverse butterfly circuit and Shift/Rotate 2’s complement inverse butterfly circuit show that both the circuits can be used for left and right shift and rotate operations. From the comparison it can be seen that the hardware required and the delay generated in the 2’s complement inverse butterfly circuit is nearly half of that of Data reversal inverse butterfly circuit. The second circuit design generates the most optimized results. Also the same circuit can be used for all the shift and rotate operations. This circuit can replace the existing shifter unit in the available microprocessors and microcontrollers.

Fig. 7: Waveform for Data Reversal IBFY

Copyright reserved © J. Mech. Cont.& Math. Sci.
Sushma Wadar et al.
Fig. 8: Simulation Result for Two’s Compliment IBFY

Table 1: Comparison of Proposed model

| Parameters       | Data Reversal IBFY Shifting Functional Units | 2’s Compliment IBFY Shifting Functional Units |
|------------------|---------------------------------------------|---------------------------------------------|
| No of Slices     | 24                                          | 16                                          |
| 4-input LUT      | 42                                          | 28                                          |
| Bonded IOB       | 20                                          | 20                                          |
| Delay (ns)       | 13.70                                       | 7.99                                        |
IV. Conclusion

The combinational logic devices, for example multiplexers and logic gates are used for implementing the shifter circuits. An 8-bit shifter / rotator circuit is proposed. The heart of the proposed designs is inverse butterfly circuit. The above table 1 showing the values for various parameters of the proposed two new designs shows that in proposed design 2, nearly 33% of optimization in terms of hardware required that is reflected in terms of LUTs used, also the execution time has reduced by 42%. The proposed design can be used for executing all the shifts and rotate instructions. The result proves that the proposed design 2 is area efficient as well as speed efficient also which can be used for any multimedia applications.

References

I. Claudio Brunelli, ‘Design of Hardware Accelerators for Embedded Multimedia Applications’, 2009

II. Eduardo Jonathan Martínez Montes, Facultatd’Informàtica de Barcelona (FIB), “Design and implementation of a Multimedia Extension for a RISC Processor”, Master in Innovation and Research in Informatics (MIRI-HPC), 2 July 2015.

III. Gaurav Mitra, Beau Johnston, Alistair P. Rendell, and Eric McCreath, “Use of SIMD Vector Operations to Accelerate Application Code Performance on Low-Powered ARM and Intel Platforms”, 2013 IEEE International Symposium on Parallel & Distributed Processing, Workshops and PhD Forum, 10.1109/IPDPSW.2013.207

IV. Mario Garrido, Member, Jesus Grajal and Oscar Gustafsson, “Optimum Circuits for Bit Reversal”, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS PART II: EXPRESS BRIEFS

V. Neil Burgess, “Assessment of Butterfly Network VLSI Shifter Circuit”, 978-1-4244-9721-8/10/ pp 92-96, Asilomar 2010 ©2010 IEEE

VI. Sabyasachi Das and Sunil P. Khatri, “A Timing-Driven Approach to Synthesize Fast Barrel Shifters”, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS, VOL. 55, NO. 1, Pgs. 31-36, JANUARY 2008

VII. Sushma Wadar, D S Bormane, S C Patil, Avinash Patil, ‘A Novel Approach to Perform Shift/Rotate and Bit Permutation Operation’, ICNTET, GRT Institute of Engineering and Technology, Tirutanni, Chennai.
VIII. Sushma Wadar, D S Bormane, S C Patil, Avinash Patil, ‘A Novel Approach to Perform Shift/Rotate and Bit Permutation Operation’, ICNTET, GRT Institute of Engineering and Technology, Tirutanni, Chennai.

IX. White Paper, ‘Developing Embedded Applications with ARM® CortexTM-M1Processors in Actel IGLOO and Fusion FPGAs’, March 2009

X. 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

XI. Yedidya Hilewitz, Member, IEEE, and Ruby B. Lee, ‘A New Basis for Shifters in General-Purpose Processors for Existing and Advanced Bit Manipulation’, IEEE TRANSACTIONS ON COMPUTERS, VOL. 58, NO. 8, AUGUST 2009 pp 1036-1048

Copyright reserved © J. Mech. Cont.& Math. Sci.
Sushma Wadar et al.

32