CHAPTER-6
LUT-MULTIPLIER BASED BLOCK LIFTING 2-D DWT STRUCTURE

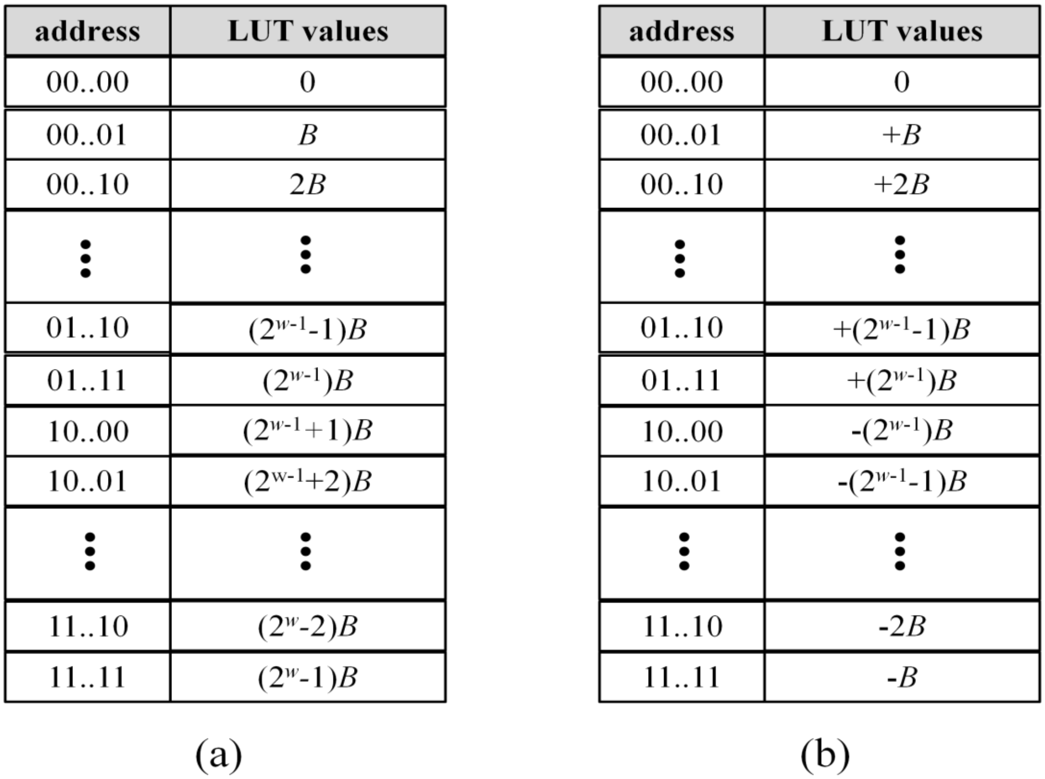
6.1 INTRODUCTION

It has been observed that DA does not offer an efficient design for short-length FIR filter such as wavelet filters. The convolution-based DWT algorithm is considered to develop a DA design. But, 2-D DWT computation based on convolution scheme involves more on-chip memory words than the lifting scheme [Mohanty and Meher (2013)]. For example: the line-based 1-level 2-D DWT structure using (9/7) filter requires ($17N$) on-chip memory words (single-port RAM) and the corresponding lifting-based structure requires ($11N$) on-chip memory words (single-port RAM) [Mohanty and Meher (2013)]. Since the image size (N) is significantly large than the wavelet filter size, the DA scheme does not provides a better hardware design. A LUT-multiplier based design (direct-memory based) could be a better scheme to obtain an area-delay efficient parallel 2-D DWT design. Several design schemes have been proposed in the technical literature for efficient realization of LUT multiplier [Guo *et.al* (1992), Meher (2009), Hsiao *et.al* (2015)]. In these designs, the major issue is the reduction of LUT size. The lifting constants of (9/7) filter are both positive and negative numbers. Therefore, lifting DWT involves signed multiplications. However, the existing LUT multiplier designs are proposed for multiplication of unsigned numbers. We find that an unsigned LUT multiplier cannot be used for multiplication of signed numbers in a straight forward manner as mentioned in the literature. Negative partial products are required for signed multiplications. Address bits required to read the positive and negative partial products are different than those used by the unsigned LUT multiplier. Separate address generator logic is required to design LUT multiplier for sign multiplication. An external adder unit is required when small size LUTs are used to implement the constant multiplier. The partial product values retrieved from different LUTs are shift-added in the adder unit to obtain the final multiplier output. The design of adder unit is straight-forward in case of unsigned multiplication. But, the design of adder unit of signed LUT multiplier is different than that of unsigned LUT multiplier due to the presence of sign bit in positive and negative partial products. These issues will be discussed in this chapter which is overlooked in the existing LUT multiplier design. The rest of the Chapter is organized as follows: The proposed LUT multiplier design is presented in Section 6.2. The proposed LUT multiplier based lifting DWT structure is discussed in Section 6.3. Hardware complexity of 2-D DWT structure using LUT multiplier is discussed in Section 6.4. The conclusion is presented in Section 6.5

6.2 PROPOSED LUT MULTIPLIER DESIGN

Consider two *w*-bit numbers *A* and *B* are multiplied using LUT multiplier, where *A* is multiplier and *B* is a multiplicand which is a constant. The multiplication of *A* with *B* is unsigned when *A* is a positive number else the multiplication is said to signed multiplication. Unsigned multiplication of *A* and *B* can be performed using a LUT of width (2^w) and depth 2^w, where ($2w$) is the bit-width of multiplication result. Since *B* is constant, (2^w) possible multiplication results corresponding to *B* can be pre-computed and stored in a LUT. The (2^w)-word LUT of width 2*w*-bit stores the (2^w) possible product values corresponding to the *w*-bit multiplier operand (*A*) which is used as address word to read the multiplication result from the LUT. Therefore, the multiplication operation becomes a simple LUT read operation. Content of the LUT of *w*-bit unsigned multiplier is shown in Figure 6.1(a). However, the LUT content of singed multiplier is different from the LUT contents of unsigned multiplier. In signed multiplication, the multiplier operand *A* is a signed number and represented in 2's complement format. The multiplication result of a *w*-bit signed number (*A*) with a constant *B* takes one value from a set of (2^w) possible results $\{-2^{w-1}B, -(2^{w-1}+1)B, \ldots, -2B, -B, 0, +B, +2B, \ldots, +(2^{w-1}-1)B, +2^{w-1}B\}$. LUT content of *w*-bit signed multiplier is shown in Figure 6.1(b). For 12-bit or higher bit-width the LUT size (depth) become too large to implement in a hardware design which are inherently resource constrained. To overcome this difficulty LUT decomposition scheme is used [Meher (2010)], where the single large LUT is implemented using a group of small size LUTs and one external adder unit. The LUT decomposition is undertaken based on the number of multiplier bits are consumed by each small LUT to read the LUT values. A (2^p)-word small LUT of unsigned multiplier consumes *p*-multiplier bits to read the LUT values. Therefore, (*w*/*p*) number of small LUTs of size (2^p) is required to implement one large LUT of size (2^w)-words. Each small LUT consumes a group of *p*-consecutive multiplier bits and produces one partial product. Mostly 8-

word/16-word LUTs are preferred to implement the small size LUT. Contents of 16-word LUT of unsigned multiplier is shown in Figure 6.2(a). The (*w*/*p*) number of small LUTs consumes *w*-bits multiplier operand and produce (*w*/*p*) partial products. These partial products are shift-accumulated in a (*w*/*p*)-word shift adder tree (SAT).



address	LUT values
00..00	0
00..01	B
00..10	$2B$
⋮	⋮
01..10	$(2^{w-1}-1)B$
01..11	$(2^{w-1})B$
10..00	$(2^{w-1}+1)B$
10..01	$(2^{w-1}+2)B$
⋮	⋮
11..10	$(2^{w}-2)B$
11..11	$(2^{w}-1)B$

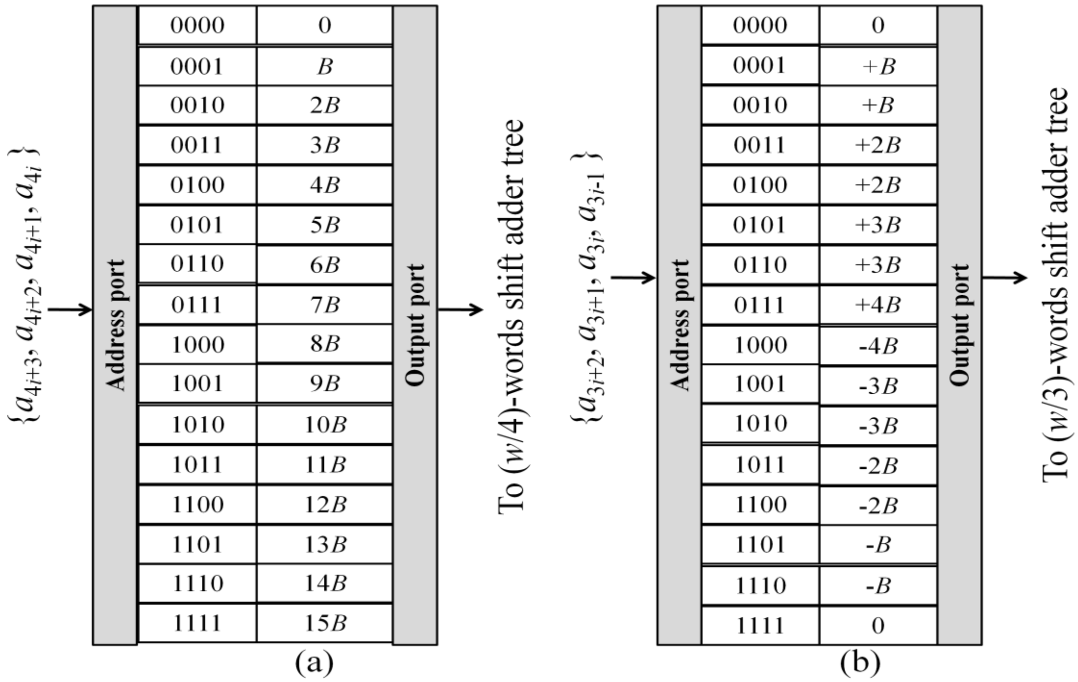
(a)

address	LUT values
00..00	0
00..01	$+B$
00..10	$+2B$
⋮	⋮
01..10	$+(2^{w-1}-1)B$
01..11	$+(2^{w-1})B$
10..00	$-(2^{w-1})B$
10..01	$-(2^{w-1}-1)B$
⋮	⋮
11..10	$-2B$
11..11	$-B$

(b)

Figure 6.1: LUT contents of *w*-bit constant multiplication. (a) Unsigned multiplication. (b) Signed multiplication. *A* and *B* are, respectively, the multiplier and multiplicand operands of the *w*-bit constant multiplier.

The LUT contents of small size LUTs of signed multiplication is different from the LUT contents of unsigned multiplier. For signed multiplication, each small LUT stores the partial products corresponding to the *p*-bit signed digit with the multiplicand operand *B*. Note that *p*-bit signed digit is used as the address bit to read the partial products stored in the (2^{p})-word small LUT. The LUT contents or the partial product set depends on how the signed digits are formed from the *w*-bit multiplier word. Since Booth recording represents the multiplier word in signed digit for Booth multiplication, this algorithm is considered for grouping of multiplier bits in the form of signed digits. According to Booth recording, signed digits are obtained with one overlapping bit with the adjacent digit. The LSB of each digit represents the overlapping bit and the MSB represents the signed bit. Therefore, the multiplication of Booth encoded digit $\{d_i=a_{3i+2},\ a_{3i+1},\ a_{3i}\ a_{3i-1}\}$ with multiplicand B represents the partial products corresponding to signed multiplication of r=(p-1) multiplier bits with B. Therefore, the number of small size LUTs required for sign multiplication using Booth encoding would be (w/r) where the number of LUTs required for unsigned multiplication is (w/p). For example, for w=12, and p=4, three 16-word LUT is required to perform the unsigned multiplication whereas four 16-word LUTs are required to perform the signed multiplication.


$\{a_{4i+3}, a_{4i+2}, a_{4i+1}, a_{4i}\}$
Address port
Output port
To (w/4)-words shift adder tree
(a)
$\{a_{3i+2}, a_{3i+1}, a_{3i}, a_{3i-1}\}$
Address port
Output port
To (w/3)-words shift adder tree
(b)

0000	0
0001	B
0010	2B
0011	3B
0100	4B
0101	5B
0110	6B
0111	7B
1000	8B
1001	9B
1010	10B
1011	11B
1100	12B
1101	13B
1110	14B
1111	15B

0000	0
0001	+B
0010	+B
0011	+2B
0100	+2B
0101	+3B
0110	+3B
0111	+4B
1000	-4B
1001	-3B
1010	-3B
1011	-2B
1100	-2B
1101	-B
1110	-B
1111	0

Figure 6.2: 16 word LUT contents of w-bit constant multiplication. (a) Unsigned multiplication. (b) Signed multiplication. A and B are, respectively, the multiplier and multiplicand operands of the w-bit constant multiplier.

Content of 16-word LUT of 12-bit unsigned and signed multiplication is shown in Figure 6.2(b). The LUT table of signed multiplication is obtained from the encoding scheme given in Table 4.3 for radix-8 Booth multiplication. Using the symmetric product coding, the LUT size of Figure 6.2 (b) is reduced to 8-word as shown in Figure 6.3. However, the LUT uses an external reset circuit to obtain the '*zero*' partial product. Also, a separate control logic is required to obtain the address bits ($s_{i,2}$, $s_{i,1}$, $s_{i,0}$) from the Booth recorded digit $\{d_i=a_{3i+2},\ a_{3i+1},\ a_{3i}\ a_{3i-1}\}$. The reset control ($s_{i,3}$) is also obtained from the control logic. As shown in Figure 6.3, half of the lower half LUT values are anti-symmetric to the upper-half LUT values. Using anti-symmetric product coding, the LUT size is reduced to 4-words (shown in Figure 6.4). An extra complement circuit is used in addition to the reset circuit to introduce sign to the partial products. The sign control signal ($s_{i,2}$) of each LUT is added with the corresponding partial-product term in the shift-adder unit.

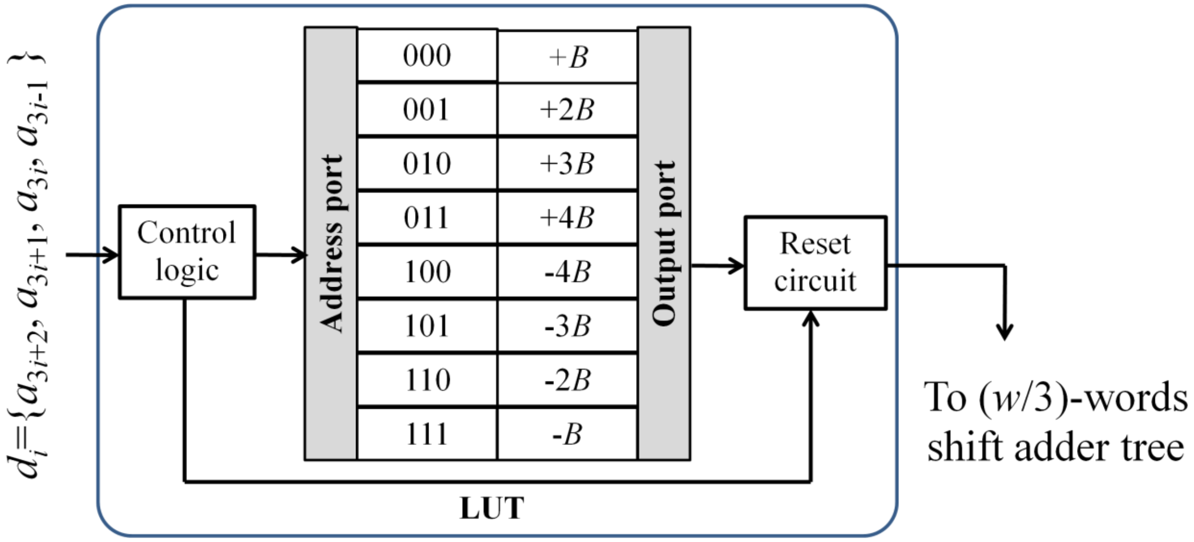


Figure 6.3: LUT contents of signed multiplier using symmetric product coding of radix-8 Booth recording.

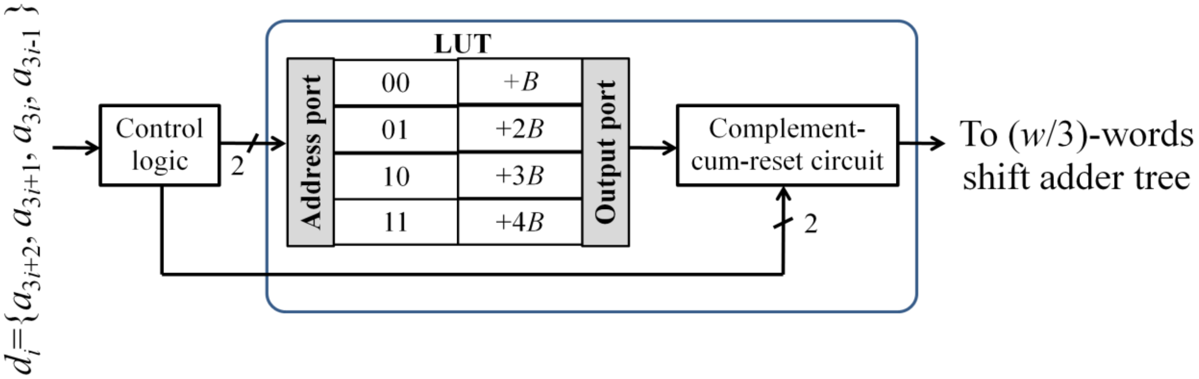

$d_i=\{a_{3i+2}, a_{3i+1}, a_{3i}, a_{3i-1}\}$
LUT
Control logic
2
Address port
00
+B
01
+2B
10
+3B
11
+4B
Output port
Complement-cum-reset circuit
2
To (w/3)-words shift adder tree

Figure 6.4: LUT contents of signed multiplication using symmetric and anti-symmetric product coding of radix-8 Booth recording.

This is required to covert the 1's complement into 2's complement form. In general the LUT size for decomposition factor (r=p-1) is reduced from 2^{r+1} to 2^{r-1} using symmetric and anti-symmetric product coding at the cost of a small overhead in terms of complement-cum reset circuit and control logic.

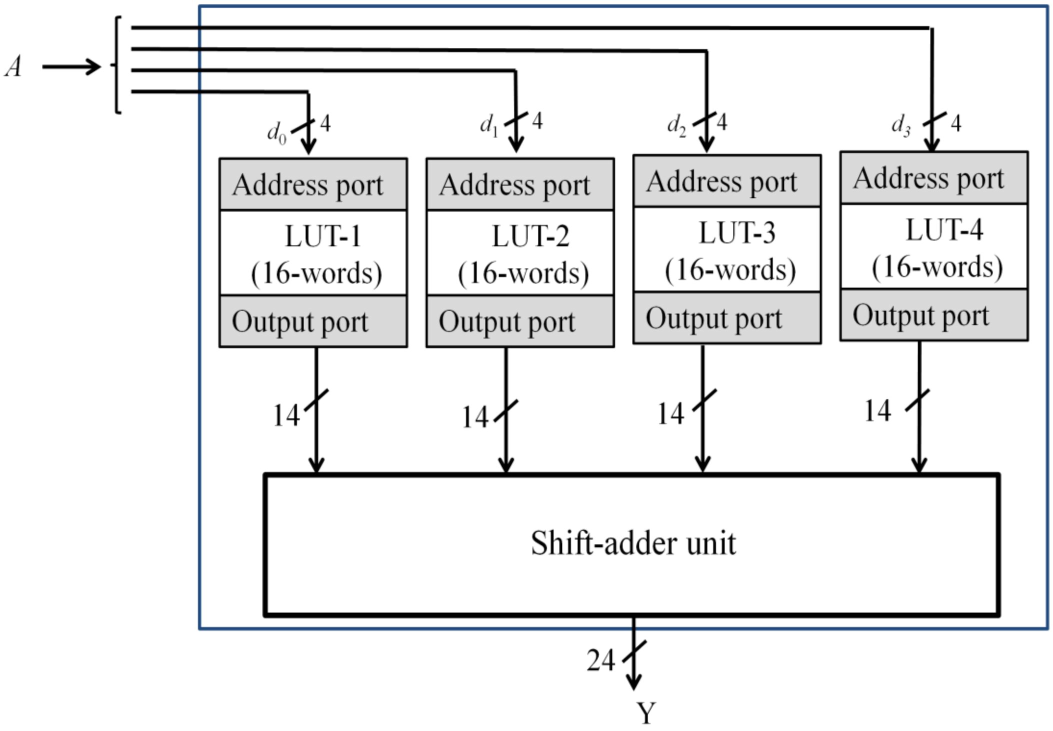


Figure 6.5: Block diagram of 12-bit signed LUT multiplier using radix-2^3 (LUT decomposition factor r=3) Booth recording.

The block diagram of 12-bit signed LUT multiplier using radix-8 Booth encoding is shown in Figure 6.5. Content of 16-word LUT is shown in Figure 6.2(b). Each LUT receives one radix-2^3 Booth recorded digit {d_i =a_{3i+2}, a_{3i+1}, a_{3i} a_{3i-1}} at its address port from the w-bit multiplier operand where {a_{3i-1}} is the overlapping bit. For example: the control logic of first LUT receives the digit {d_0=a_2, a_1, a_0 a_{-1}} and the second LUT receives the digit {d_1=a_5, a_3, a_3 a_2}. Similarly, the third and the forth LUT receives the digits {d_2=a_8, a_7, a_6 a_5} and {d_3=a_{11}, a_{10}, a_9 a_8}. Four partial product values retrieved from four LUT in parallel and these values are added in the shift-adder tree to get the multiplication result.

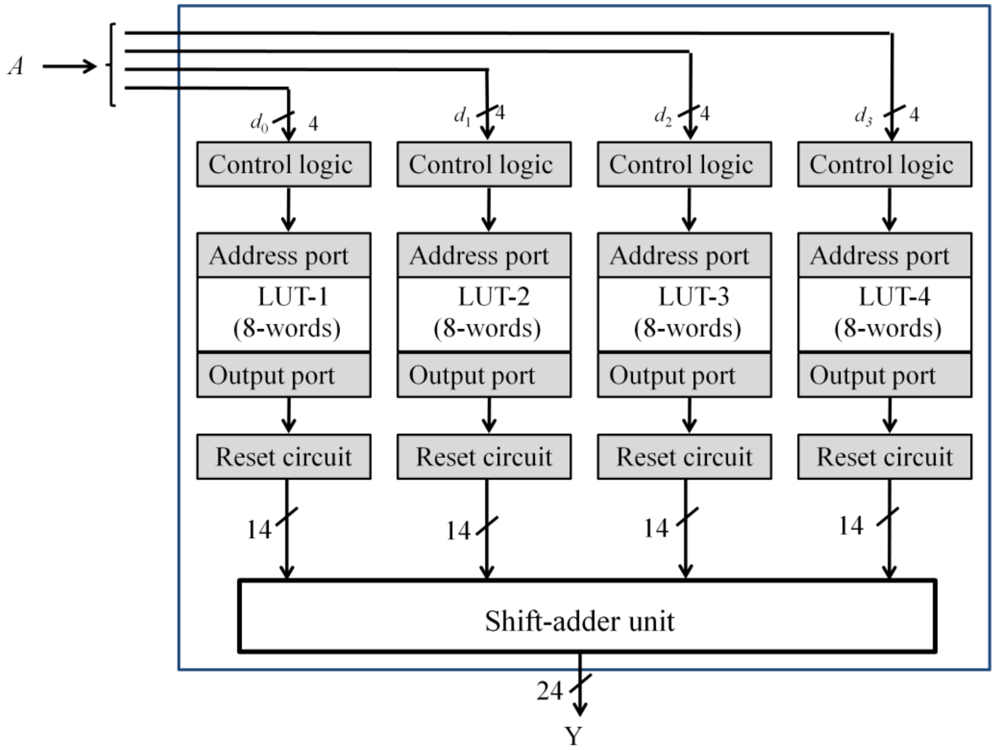


Figure 6.6: Block diagram of 12-bit signed LUT multiplier using symmetric radix-2^3 (LUT decomposition factor *r*=3) Booth recording.

The block diagram of 12-bit signed LUT multiplier using symmetric radix-8 Booth encoding is shown in Figure 6.6 and content of 8-word LUT is shown in Figure 6.3. The control logic of each LUT receives one radix-2^3 Booth recorded digit $\{d_i = a_{3i+2}, a_{3i+1}, a_{3i}\ a_{3i-1}\}$ from the *w*-bit multiplier operand where $\{a_{3i-1}\}$ is the overlapping bit. The control logic of each LUT calculate for signals $(s_{i,3}, s_{i,2}, s_{i,1}, s_{i,0})$ from each 4-bit digit. Out of these four control signals, three control signals $(s_{i,2}, s_{i,1}, s_{i,0})$ are consumed by the LUT to read the partial products and the forth control signal $(s_{i,3})$ is used by the reset circuit. Finally, the shift adder tree accumulates the partial products to calculate the multiplier result.

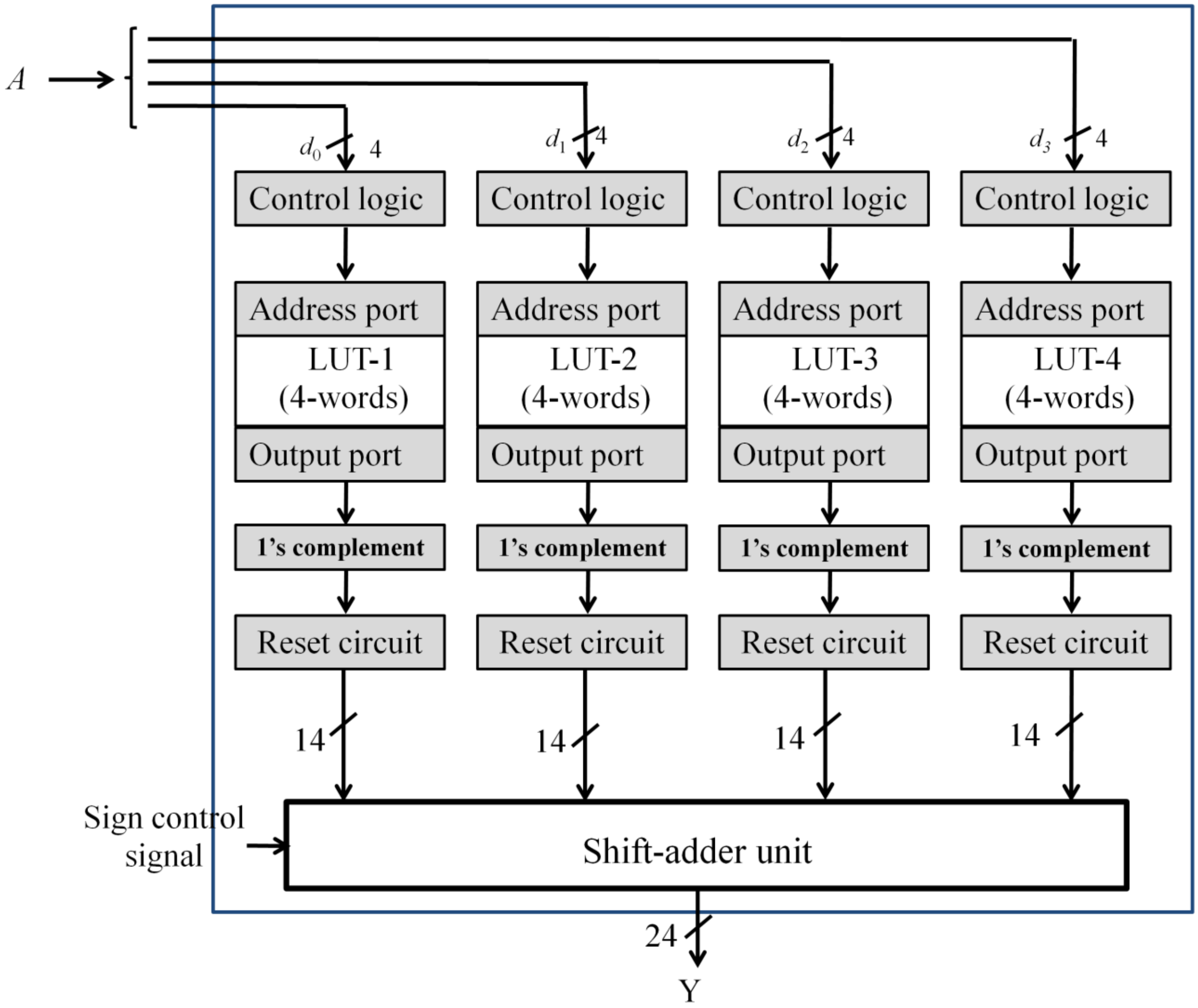


Figure 6.7 Block diagram of 12-bit signed LUT multiplier using symmetric and anti-symmetric radix-2^3 (LUT decomposition factor r=3) Booth recording.

The block diagram of 12-bit signed LUT multiplier using symmetric and anti-symmetric radix-8 Booth encoding is shown in Figure 6.7 and content of 4-word LUT is shown in Figure 6.4. The control logic of each LUT calculate for signals ($s_{i,3}$, $s_{i,2}$, $s_{i,1}$, $s_{i,0}$) from each 4-bit digit. Out of these four control signals, two control signals ($s_{i,1}$, $s_{i,0}$) are consumed by the LUT to read the partial products. The control signal ($s_{i,2}$) is used by 1's complement circuit to introduce the sign to the negative partial products and the forth control signal ($s_{i,3}$) is used by the reset circuit. The shift adder tree receives the 4 partial products from four LUTs and four sign control signals {s_{32}, s_{22}, s_{12}, s_{02}} from four control logics. The sign control bit is added to the LSB of the corresponding partial product term to convert it into 2's complement number and then shift added to obtain the multiplier result. The shift-adder tree can use a simple adder tree or a Wallace tree. In 2’s complement arithmetic, appropriate sign extension scheme has to be considered while accumulation the partial products in the shift-adder tree. Sign extension scheme used in Wallace tree is shown in Figure 6.8 for 4-word 12-bit shift-adder tree.

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$p_{0,13}$	$p_{0,13}$	$p_{0,13}$	$p_{0,13}$	$p_{0,13}$	$p_{0,13}$	$p_{0,13}$	$p_{0,13}$	$p_{0,13}$	$p_{0,13}$	$p_{0,13}$	$p_{0,12}$	$p_{0,11}$	$p_{0,10}$	$p_{0,9}$	$p_{0,8}$	$p_{0,7}$	$p_{0,6}$	$p_{0,5}$	$p_{0,4}$	$p_{0,3}$	$p_{0,2}$	$p_{0,1}$	$p_{0,0}$
$p_{1,13}$	$p_{1,13}$	$p_{1,13}$	$p_{1,13}$	$p_{1,13}$	$p_{1,13}$	$p_{1,13}$	$p_{1,13}$	$p_{1,12}$	$p_{1,11}$	$p_{1,10}$	$p_{1,9}$	$p_{1,8}$	$p_{1,7}$	$p_{1,6}$	$p_{1,5}$	$p_{1,4}$	$p_{1,3}$	$p_{1,2}$	$p_{1,1}$	$p_{1,0}$			
$p_{2,13}$	$p_{2,13}$	$p_{2,13}$	$p_{2,13}$	$p_{2,13}$	$p_{2,12}$	$p_{2,11}$	$p_{2,10}$	$p_{2,9}$	$p_{2,8}$	$p_{2,7}$	$p_{2,6}$	$p_{2,5}$	$p_{2,4}$	$p_{2,3}$	$p_{2,2}$	$p_{2,1}$	$p_{2,0}$						
$p_{3,13}$	$p_{3,13}$	$p_{3,12}$	$p_{3,11}$	$p_{3,10}$	$p_{3,9}$	$p_{3,8}$	$p_{3,7}$	$p_{3,6}$	$p_{3,5}$	$p_{3,4}$	$p_{3,3}$	$p_{3,2}$	$p_{3,1}$	$p_{3,0}$									
z_{23}	z_{22}	z_{21}	z_{20}	z_{19}	z_{18}	z_{17}	z_{16}	z_{15}	z_{14}	z_{13}	z_{12}	z_{11}	z_{10}	z_9	z_8	z_7	z_6	z_5	z_4	z_3	z_2	z_1	z_0

(a)

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$p_{0,13}$	$p_{0,13}$	$p_{0,13}$	$p_{0,13}$	$p_{0,13}$	$p_{0,13}$	$p_{0,13}$	$p_{0,13}$	$p_{0,13}$	$p_{0,13}$	$p_{0,13}$	$p_{0,12}$	$p_{0,11}$	$p_{0,10}$	$p_{0,9}$	$p_{0,8}$	$p_{0,7}$	$p_{0,6}$	$p_{0,5}$	$p_{0,4}$	$p_{0,3}$	$p_{0,2}$	$p_{0,1}$	$p_{0,0}$
$p_{1,13}$	$p_{1,13}$	$p_{1,13}$	$p_{1,13}$	$p_{1,13}$	$p_{1,13}$	$p_{1,13}$	$p_{1,13}$	$p_{1,12}$	$p_{1,11}$	$p_{1,10}$	$p_{1,9}$	$p_{1,8}$	$p_{1,7}$	$p_{1,6}$	$p_{1,5}$	$p_{1,4}$	$p_{1,3}$	$p_{1,2}$	$p_{1,1}$	$p_{1,0}$			s_0
$p_{2,13}$	$p_{2,13}$	$p_{2,13}$	$p_{2,13}$	$p_{2,13}$	$p_{2,12}$	$p_{2,11}$	$p_{2,10}$	$p_{2,9}$	$p_{2,8}$	$p_{2,7}$	$p_{2,6}$	$p_{2,5}$	$p_{2,4}$	$p_{2,3}$	$p_{2,2}$	$p_{2,1}$	$p_{2,0}$			s_1			
$p_{3,13}$	$p_{3,13}$	$p_{3,12}$	$p_{3,11}$	$p_{3,10}$	$p_{3,9}$	$p_{3,8}$	$p_{3,7}$	$p_{3,6}$	$p_{3,5}$	$p_{3,4}$	$p_{3,3}$	$p_{3,2}$	$p_{3,1}$	$p_{3,0}$			s_2						
														s_3									
z_{23}	z_{22}	z_{21}	z_{20}	z_{19}	z_{18}	z_{17}	z_{16}	z_{15}	z_{14}	z_{13}	z_{12}	z_{11}	z_{10}	z_9	z_8	z_7	z_6	z_5	z_4	z_3	z_2	z_1	z_0

(b)

Figure 6.8: (a) Sign extension scheme of 4-words and 12-bit shift-adder tree used in unsigned multiplication using 8-words LUT. (b) Sign extension scheme of 4-words and 12-bit shift-adder tree used in signed multiplication using 4-words LUT.

6.3 PROPOSED LUT MULTIPLIER BASED LIFTING DWT STRUCTURE

The block diagram of lifting 1-D DWT is shown in Figure 6.6(a) and the internal diagram of the functional element (FE) in Figure 6.6(b). LUT tables of 8-words and 4-words LUTs of 12-bit multiplication of for lifting constant and two scaling constants are given in Table 6.1-6.6.

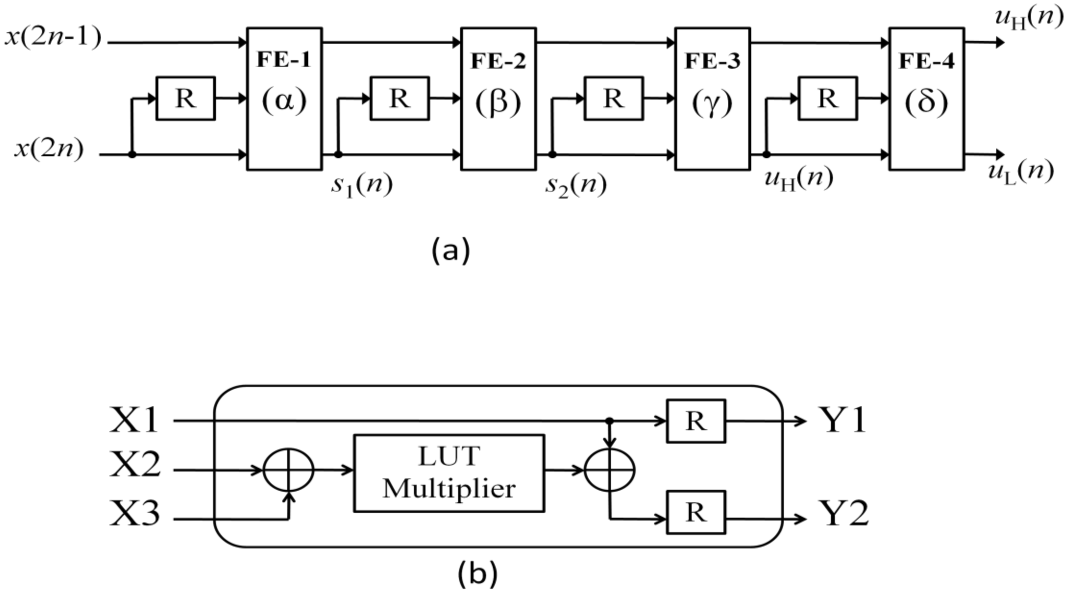


Figure 6.9: (a) General block diagram of lifting 1-D DWT using 9/7 filter. (b) Internal structure of functional element (FE).

Table 6.1: LUT table (8-words) for 12-bit multiplication of lifting constant α=value and β=value

Address ($s_2\,s_1\,s_0$)	LUT values	LUT values in 12-bit 2's complement format		Address ($s_2\,s_1\,s_0$)	LUT values	LUT values in 12-bit 2's complement format
000	α	111110.0110101000		000	β	111111.1111001010
001	2α	111100.1101010000		001	2β	111111.1110010100
010	3α	111011.0011111000		010	3β	111111.1111011110
011	4α	111001.1010100000		011	4β	111111.1100101000
010	-α	000001.1001011000		100	- β	000000.0000111000
101	-2α	000011.0010110000		101	-2β	000000.0001110000
110	-3α	000100.1100001000		110	-3β	000000.0000100010
111	-4α	000110.0101100000		111	-4β	000000.0001110000

Table 6.2: LUT table (8-words) for 12-bit multiplication of lifting constant γ=value and δ=value

Address ($s_2\,s_1\,s_0$)	LUT values	LUT values in 12-bit 2's complement format		Address ($s_2\,s_1\,s_0$)	LUT values	LUT values in 12-bit 2's complement format
000	γ	000000.1110001000		000	δ	000000.0111000110
001	2γ	000001.1100010000		001	2δ	000000.1110001100
011	3γ	000000.1110001000		010	3δ	000001.0101010010
010	4γ	000011.1000100000		011	4δ	000001.1100011000
101	-γ	111111.0001111000		100	- δ	111111.1000111010
110	-2γ	111110.0011110000		101	-2δ	111111.0001110100
111	-3γ	111111.0001111000		110	-3δ	111111.0001110100
000	-4γ	111100.0111100000		111	-4δ	111110.0001110100

Table 6.3: LUT table (8-words) for 12-bit multiplication of scaling constants k^2=value and (1/k^2) =value

Address ($s_2\,s_1\,s_0$)	LUT values	LUT values in 12-bit 2's complement format		Address ($s_2\,s_1\,s_0$)	LUT values	LUT values in 12-bit 2's complement format
000	k^2	000001.1001100011		000	$1/k^2$	000000.1011101110
001	$2k^2$	000011.0011000110		001	$2/k^2$	000001.0111011100
010	$3k^2$	000100.1100101001		010	$3/k^2$	000010.0011001101
011	$4k^2$	000110.0110001100		011	$4/k^2$	000010.1110111100
100	$-k^2$	1111111.1001100100		100	$-1/k^2$	1111111.0111101010
101	$-2k^2$	1111111.0010011000		101	$-2/k^2$	111111.0101100100
110	$-3k^2$	1111110.1110010110		110	$-3/k^2$	111111.0011011111
111	$-4k^2$	1111111.0010011001		111	$-4/k^2$	111111.0001011001

Table 6.4: LUT table (4-words) for 12-bit multiplication of lifting constant α=value and β=value

Address ($s_1\,s_0$)	LUT values	LUT values in 12-bit 2's complement format		Address ($s_1\,s_0$)	LUT values	LUT values in 12-bit 2's complement format
00	α	111110.0110101000		00	β	111111.1111001010
01	2α	111100.1101010000		01	2β	111111.1110010100
10	3α	111011.0011111000		10	3β	111111.1111011110
11	4α	111001.1010100000		11	4β	111111.1100101000

Table 6.5: LUT table (4-words) for 12-bit multiplication of lifting constant γ=value and δ=value

Address ($s_1\,s_0$)	LUT values	LUT values in 12-bit 2’s complement format		Address ($s_1\,s_0$)	LUT values	LUT values in 12-bit 2’s complement format
00	γ	000000.1110001000		00	δ	000000.0111000110
01	2γ	000001.1100010000		01	2δ	000000.1110001100
10	3γ	000000.1110001000		10	3δ	000001.0101010010
11	4γ	000011.1000100000		11	4δ	000001.1100011000

Table 6.6: LUT table (8-words) for 12-bit multiplication of scaling constants k^2=value and $(1/k^2)$=value

Address ($s_1\,s_0$)	LUT values	LUT values in 12-bit 2’s complement format		Address ($s_1\,s_0$)	LUT values	LUT values in 12-bit 2’s complement format
00	k^2	000001.1001100011		00	$1/k^2$	000000.1011101110
01	$2k^2$	000011.0011000110		01	$2/k^2$	000001.0111011100
10	$3k^2$	000100.1100101001		10	$3/k^2$	000010.0011001101
11	$4k^2$	000110.0110001100		11	$4/k^2$	000010.1110111100

The PE design of PU of j-th decomposition level of 2-D DWT is shown in Figure 3.8. Each section of the PE is replaced by the proposed LUT-multiplier based lifting 1-D DWT structure of Figure 6.9(a) to have a PE design using LUT multiplier. Each multiplier of the scaling unit of Figure 3.10 is also implemented using the proposed LUT-multiplier. The LUT-multiplier based PE and scaling unit is used in the PU design shown in Figure.3.7 to have a PU design using LUT multiplier. Therefore, the proposed LUT-based lifting 2-D DWT structure is identical to the proposed generic multiplier based 2-D DWT structure of Figure 3.6 except that each generic multiplier of multiplier based 2-D DWT structure is replaced by LUT-based multiplier which comprises of ROM LUTs and adders.

Table 6.7: Comparison of synthesis result of structures

DWT level	Designs	Block Size	MCP (ns)	Total area (μm^2)	ADP ($\mu m^2 \times ms$)	Core Power (mW)
J=2	Hu and Jong (2013)	16	1.76	307727	2.23	12.1
	Proposed (Using radix-8)	16	1.69	296511	2.05	11.4
	Proposed (Using LUT)	16	0.86	183352	0.16	10.47
J=3	Hu and Jong (2013)	64	1.81	766000	1.42	29.9
	Proposed (Using Radix-8)	64	1.72	615964	1.08	27.1
	Proposed (Using LUT)	64	0.97	526063	0.52	23.63

6.4. AREA-DELAY COMPLEXITIES

The proposed multiplier based 2-D DWT structure proposed in Chapter 4 [Mohanty and Choubey (2017)] is coded in VHDL using the proposed LUT-multiplier and the proposed 2-D DWT structure proposed in Chapter 5 using radix-8 constant fixed-width multiplier are coded for decomposition level *J*=2 and 3; and block-sizes *S*=16 and 64. The existing structure of [Hu and Jong (2013)] is also coded in VHDL using radix-4 generic constant Booth multiplier in VHDL for the same decomposition level and block-sizes. All designs are synthesized in Synopsys Design Compiler using TSMC 65 nm CMOS library. The area, minimum clock period (MCP) and power consumption reported by the Design Compiler are listed in Table 6.10 for comparison. Power consumption of all designs is estimated at common clock of frequency 50 MHz. As shown in Table 6.10, the proposed LUT-multiplier based structure for block-size 16 and 64 involves 49% and 44% less MCP, 38% and 15% less area than those of the proposed radix-8 constant multiplier-based structure, respectively. Compared with the structure of [Hu and Jong (2013)], the proposed LUT-multiplier based structure for block-size 16 and 64 involves 51% and 46% less MCP, 40% and 31% less area, respectively. The proposed LUT-multiplier based structure for block-size 16 and 64 involves 92% and 52% less ADP, consumes 8% and

13% less power than those of proposed radix-8 constant multiplier-based structure, respectively. Compared with the structure of [Hu and Jong (2013)], the proposed LUT-multiplier based structure for block-size 16 and 64 involves 93% and 63% less ADP, and dissipates 13% and 21% less power, respectively.

6.5 CONCLUSION

Signed constant multiplication using LUT is presented. Booth encoding scheme is used to design the LUT contents of signed multiplication which contains both positive and negative partial products terms. In general, (w/r) small size LUTs of size (2^p) words are required to perform w-bit sign multiplication, where r=(p-1). Symmetric and anti-symmetric coding schemes are used to reduce the LUT size. However, symmetric and anti-symmetric coding schemes involve separate address generator logic to fetch the correct partial-product from the LUT. LUT multiplier design of 12-bit multiplication using 8-words and 4-words LUT is presented. The proposed LUT multiplier is used design the lifting 1-D DWT structure. The lifting 1-D DWT design is used as a building block in the block-based lifting 2-D DWT structure presented in Chapter-4 to find a LUT-multiplier based design. ASIC synthesis result shows that the proposed LUT-multiplier based structure for block-size 16 and 64 involves 49% and 44% less MCP, 38% and 15% less area than those of the proposed radix-8 constant multiplier-based structure, respectively. Compared with the existing similar structure of [Hu and Jong (2013)], the proposed LUT-multiplier based structure for block-size 16 and 64 involves 93% and 63% less ADP, and dissipates 13% and 21% less power, respectively. The proposed LUT-multiplier based structure for block-size 16 and 64 involves 92% and 52% less ADP, and consumes 8% and 13% less power than those of proposed radix-8 constant multiplier-based structure, respectively. Therefore, the proposed LUT-multiplier offers an efficient hardware design for high-speed implementation of lifting 2-D DWT.