New hardware and power efficient sporadic logarithmic shifters for DSP applications

Chen, Jiajia; Chang, Chip-Hong; Wang, Yujia; Zhao, Juan; Rahardja, Susanto

2017

Chen, J., Chang, C.-H., Wang, Y., Zhao, J., & Rahardja, S. (2018). New hardware and power efficient sporadic logarithmic shifters for DSP applications. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 37(4), 896-900.
doi:10.1109/TCAD.2017.2740300

<https://hdl.handle.net/10356/105840>

<https://doi.org/10.1109/TCAD.2017.2740300>

© 2017 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. The published version is available at:
<https://doi.org/10.1109/TCAD.2017.2740300>

Downloaded on 26 Apr 2023 22:59:21 SGT

New Hardware and Power Efficient Sporadic Logarithmic Shifters for DSP Applications

Jiajia Chen, Chip-Hong Chang, *Senior Member, IEEE*, Yujia Wang, Juan Zhao and Susanto Rahardja, *Fellow, IEEE*

Abstract—Shifting an input data by variable amounts is commonly found in arithmetic operations, data encoding and bit-indexing. Although some shift amounts along the entire shift range are not required, the typical realization by a full-range logarithmic shifter requires full implementation and therefore suffers from complexity and power overhead. In this paper, the notion of sporadic logarithmic shifter is introduced for the first time, and a new design methodology is proposed for its optimization. By reusing parts of existing substructure of conventional logarithmic shifter or post-multiplexing the hardwired shifts, contagious subranges of desirable shift amounts are successively realized. Synthesis results on 8-bit and 16-bit sporadic logarithmic shifters show average ASIC area and power savings of up to 73.24% and 63.90% respectively, over conventional logarithmic shifters. In addition, by applying the proposed sporadic logarithmic shifters to the DCT architecture, at least 47.5% area savings and 2.9% power savings can be achieved over two constant multipliers based DCT architectures reported in the literature.

Index Terms — Programmable Shifters, digital IC Design, Digital Signal Processing.

I. INTRODUCTION

SHIFTING is a commonly required and often important operation for data manipulation in many general-purpose micro-processors [1] and embedded digital signal processors, such as low-density parity-check (LDPC) decoder [2], programmable finite impulse response (FIR) filters [3], [4], residue number system (RNS) [5] and reconfigurable multipliers (RM) such as those in discrete cosine transform (DCT) architecture [6]. Fast but expensive s -bit logarithmic shifters are usually used in existing designs [1] – [3], [5], [6] to flexibly shift a variable by 0 to $s - 1$ bits at run time. According

to [3], more than 40% of the total hardware cost and power consumption of an 8-bit 100-tap programmable FIR filter are due to the logarithmic shifters. Two main factors that escalate the complexity of logarithmic shifter are the shifter range and data precision. Large shift range makes them the dominant cost contributors in applications that require multiple programmable shifters.

In many specific applications, the required shift amounts are sporadic, because the programmability over the whole gamut of shifts is typically not necessary. One of the examples is the full-range logarithmic shifter in the reconfigurable multipliers proposed in [6]. Unfortunately, the inherent unwanted shifts cannot be trivially removed from the full-range logarithmic shifter to lower its implementation cost. As logarithmic shifter minimizes the number of shifter stages by decomposing the total shift amount into shifts over power of two, hardware components required by the unwanted shift values are shared with other required shift values. To solve this problem, we represent the required shift amounts in a binary index vector at design time, which will be bisected into groups of wanted shift amounts. This bisection helps to discriminate reducible components or substructures due the unused shift values. In this paper, we propose the first systematic approach to design sporadic logarithmic shifter (SLS) to minimize sporadic shifting of an input. The SLS designed by our proposed algorithm has lower complexity and consumes less power than the existing full-range logarithmic shifter (FRLS) [5] and mux-based full-range barrel shifter (FRBS) [7]. By adopting SLS to DSP applications such as the RMs in DCT, it can effectively reduce the hardware complexity and power consumption of the entire architectures.

II. FULL-RANGE LOGARITHMIC SHIFTER

An s -bit full-range logarithmic shifter consists of $n = \lceil \log_2 s \rceil$ stages of multiplexers, where $\lceil \cdot \rceil$ is the integer ceiling function. Each stage of multiplexers is controlled by a bit b_i of the binary representation of the shift amount B . If $b_i = 1$, data input to the multiplexers in stage $(n - i)$ will be shifted by 2^i bits. Otherwise, they will flow through directly to the next stage. Fig. 1 shows an 8-bit full-range logarithmic left-shifter, which consists of 4-bit, 2-bit and 1-bit shifts in stages 1 to 3, respectively.

Manuscript received March 4, 2017; revised June 8, 2017; accepted July 23, 2017. The research and the material reported in this document are supported by the SUTD-MIT International Design Centre (IDC) at Singapore University of Technology and Design.

The authors Jiajia Chen, Yujia Wang and Juan Zhao are with Singapore University of Technology and Design, Singapore. E-mail: jiajia_chen@sutd.edu.sg

The author Chip-Hong Chang is with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. E-mail: echchang@ntu.edu.sg

The author Susanto Rahardja (Corresponding Author) is with Northwestern Polytechnical University, Xi'an, P. R. China. E-mail: susantorahardja@ieee.org

Fig. 1. 8-bit baseline full-range logarithmic left shifter architecture

Let w_x be the word length of the input x to be shifted by an s -bit full-range logarithmic shifter with the shift amount determined by $B = b_{n-1}b_{n-2} \cdots b_0$, where b_{n-l} is the control input to the multiplexers at level l . At each level l , $l = 1, 2, \dots, n$, the input x is either shifted by 2^{n-l} bits when $b_{n-l} = 1$ or not shifted when $b_{n-l} = 0$. Therefore, the number of 1-bit 2-to-1 multiplexers for the baseline full-range shifter at level l is

$$c_{MUX,BFRLS}(l) = w_x + \sum_{i=1}^l 2^{n-i}. \quad (1)$$

Its total number of multiplexers is

$$c_{MUX,BFRLS} = \sum_{l=1}^n \left(w_x + \sum_{i=1}^l 2^{n-i} \right). \quad (2)$$

Due to the leading and trailing zero-paddings, each multiplexer of the baseline design with a zero input can be replaced by a two-input AND gate with the non-zero data input bit and the control bit $\bar{b}_l$ or b_l of the multiplexer as inputs. For example, the least significant bit (LSB) y_0 of the shifter output can be expressed as $y_0 = ((x_0 \cdot \bar{b}_2) \cdot \bar{b}_1) \cdot \bar{b}_0$ to replace a multiplexer in each level by a two-input AND gate. The simplified 8-bit full-range logarithmic left-shifter is shown in Fig. 2.

Fig. 2. 8-bit optimized full-range logarithmic shifter architecture

When $b_{n-l} = 0$, the least significant 2^{n-l} bits at the l -th level are the same as the least significant 2^{n-l} bits at its upper level, and when $b_{n-l} = 1$, these bits are equal to 0. Similarly, the most significant 2^{n-l} bits at the l -th level are either equal to the most significant 2^{n-l} bits at its upper level when $b_{n-l} = 1$ or equal to 0 when $b_{n-l} = 0$. Each of these multiplexers can be replaced by a two-input AND gate. Therefore, the number of two-input AND gates at the l -th level is 2^{n-l+1} . The total number of two-input AND gates of the optimized full-range logarithmic shifter is given by

$$c_{AND,FRLS} = \sum_{i=1}^n 2^{n-i+1}. \quad (3)$$

The remaining number of 1-bit 2-1 multiplexers of the optimized full-range shifter is given by

$$c_{MUX,FRLS} = \sum_{l=1}^n \left(w_x + \sum_{i=1}^l 2^{n-i} - 2^{n-l+1} \right). \quad (4)$$

III. PROPOSED ALGORITHM FOR SLS

If only a sporadic amount of shifts in an s -bit shifter is needed, the shift amounts can be indicated by a binary index vector $K = [k_i]$, where $k_i \in \{0, 1\} \forall i = 0, 1, \dots, s-1$. $k_i = 1$ if an i -bit shift of an input data is required and $k_i = 0$ otherwise. For example, if an application requires an input data to be shifted by 0, 1, 2 and 7 bits at run time, $K = [1, 1, 1, 0, 0, 0, 0, 1]$.

A. Successive addition of sporadic shifts

When $\prod_{i=0}^{s-1} k_i = 0$, a full-range logarithmic shifter is not needed. A reduced core contiguous subrange of the shift amounts in K can be implemented by a smaller full-range logarithmic shifter before the differences are incrementally added to make up the remaining larger input shifts in K .

To identify the core subrange and order of implementation, the s -bit binary vector K is successively bisected into two subvectors of equal number of elements. If any subvector contains both '1' and '0' elements, then these inhomogeneous subvectors will be further bisected into two equal length subvectors until all subvectors contain only all '1' or all '0' elements or are single-element vectors. If s is not a power-of-two integer, i.e., $s \neq 2^n$, then s will be rounded up to the nearest power-of-two integer so that K has 2^n elements. The largest subvector of ones (called the 1-subvector henceforth) is the core subvector. If there is a tie, the largest 1-subvector that has smaller shift indices is selected as the core subvector. Fig. 3 shows the bisection result of $K = [k_0, k_1, k_2, k_3, k_4, k_5, k_6, k_7] = [1, 1, 1, 0, 0, 0, 0, 1]$ into three 1-subvectors, $K_1 = [k_0, k_1]$, $K_2 = [k_2]$ and $K_3 = [k_7]$. K_1 is the core subvector.

Fig. 3. Bisection of K into homogeneous subvectors

If K contains j consecutive zeros from k_0 to k_{j-1} , i.e., $K = [0, \dots, 0, k_j, k_{j+1}, \dots, k_{s-1}]$, then K can be reduced to $K' = [k_0, k_1, \dots, k_{s-1-j}]$ by eliminating the j zero elements until $k_0 = 1$. After the SLS for the reduced K' has been implemented, the output will be appended with j hardwired zero bits. The final shifter will have only $n = \lceil \log_2(s-j) \rceil$ stages, although the output word length $w_y = w_x + s - 1$ with s as the total number of elements in K .

The shift amounts in the core subvector can be realized using the substructure of a full-range logarithmic shifter. For a core subvector of α elements from k_i to $k_{i+\alpha-1}$, the substructure produces a subset of output bits from $w_x + k_i$ to $w_x + k_{i+\alpha-1}$. Once the core subvector has been realized, its nearest 1-subvector neighbor will be implemented at an incremental cost by using one of the following two proposed techniques to be detailed in the next section until all the 1-subvectors have been implemented.

B. The proposed techniques for SLSs complexity reduction

Depending on the distance between the previously implemented 1-subvector and its nearest 1-subvector, either reusing some parts of the existing intermediate signals or post-multiplexing the programmable and hardwired shifts technique will be applied to successively implement the nearest 1-subvector. The distance, which is the number of zeros between any two 1-subvectors, is denoted by n_z in our method.

B1. Reduction by logic reuse

Let K_a be the most recently implemented 1-subvector and K_b be its nearest 1-subvector in K . Using reduction by logic reuse, the shift amounts in K_b is produced by a few 2-input AND gates and 2-to-1 multiplexers. The incremental hardware and connections to produce the additional shifts are determined by the Boolean expressions of the output bits y_i for $i = w_x + j - 1$, where j are the indices of the element k_j of K_b . Fig. 4 illustrates the design of an SLS for $K = [1, 1, 1, 1, 1, 1, 0, 0]$. After the core subvector $K_a = [k_0, k_1, k_2, k_3]$ has been implemented, the logic functions for the shifter outputs corresponding to its nearest 1-subvector $K_b = [k_4, k_5]$ are $y_{11} = \bar{b}_0 \cdot (\bar{b}_1 \cdot (b_2 \cdot x_7)) + b_0 \cdot (\bar{b}_1 \cdot (b_2 \cdot x_6))$ and $y_{12} = b_0 \cdot (\bar{b}_1 \cdot (b_2 \cdot x_7))$. The logic components colored in grey in the dotted box are the extended hardware dedicated to K_b . The data input $\bar{b}_1 \cdot (b_2 \cdot x_6)$ to the existing multiplexer that produces the output y_{10} is reused for the cofactor of y_{11} , and the cofactor $\bar{b}_1 \cdot (b_2 \cdot x_7)$ of y_{11} is reused for the generation of y_{12} .

Fig. 4. 8-bit SLS for $K = [1, 1, 1, 1, 1, 1, 0, 0]$

If n_i is the number of elements in K_b and n_z is the number of zero elements between K_a and K_b in K , then the additional number of components (i.e., 2-input AND gates and 1-bit 2-1 multiplexers) at level l required to realize K_b is given by

$$\Delta c(l) = n_z + n_i. \quad (5)$$

This incremental cost $\Delta c(l)$ for K_b plus all existing components of the already implemented 1-subvectors at level l should not exceed the total number of components required in the l -th level of an s -bit full-range logarithmic shifter expressed in (1). Let $c_{exist}(l)$ be the total number of existing components at level l before K_b is implemented. The cost increment for realizing K_b at level l is bounded by

$$\Delta c(l) \leq c_{BFRLS}(l) - c_{exist}(l). \quad (6)$$

The number $n_{AND}(i)$ of two-input gates required at level l can be determined from the Boolean expressions of y_j for all $k_j \in K_b$. The total cost increment of realizing K_b by this technique is given by

$$\Delta c = \sum_{i=1}^n (n_{AND}(i) \times c_{AND} + (\Delta c(i) - n_{AND}(i)) \times c_{MUX}). \quad (7)$$

where c_{AND} and c_{MUX} are the costs of a two-input AND gate and a 1-bit 2-1 multiplexer, respectively.

B2. Reduction by post-multiplexing with hardwired shifts

When K_a and K_b are not close, it becomes more costly to realize the additional shift amounts in K_b . Due to the large number of shifts that can be skipped, the shift amounts in K_b are better realized independently by direct hardwired shifts of the input x . A post-multiplexer can be added to select between the hardwired shifts of the input data and the already implemented shifted outputs for K_a . Fig. 5 illustrates a left-SLS for $K = [1, 1, 1, 1, 0, 0, 0, 1]$. After the core subvector $K_a = [k_0, k_1, k_2, k_3]$ has been implemented, the outputs corresponding to its nearest 1-subvector $K_b = [k_7]$ can be obtained directly by hardwired left-shifting of the input vector x with seven zero-padded bits. When $b_2 = 1$, the 15-bit zero-padded hardwired shifts of K_b (to the right of the dashed line in Fig. 5) is selected into the output bits y_0 to y_{14} . Otherwise, the existing structure for K_a is selected into the output bits y_0 to y_{10} . The most significant $|k_{a,max} - k_{b,max}|$ output bits, y_{11} to y_{14} , are padded with zeros, where $k_{a,max}$ and $k_{b,max}$ are the maximum shift amount in K_a and K_b , respectively. In this example, $k_{a,max} = 3$ and $k_{b,max} = 7$.

Fig. 5. 8-bit SLS for $K = [1, 1, 1, 1, 0, 0, 0, 1]$

In general, a w -bit (n_i+1) -to-1 multiplexer is needed, where $w = w_x + \max\{k_{a,max}, k_{b,max}\}$ is the word length of input data and $\max\{\}$ is the maximum value function. According to [8], the area of a w -bit m -to-1 multiplexer can be approximated by $0.5 \times w \times m \times c_{mux}$, where c_{mux} is the area of a 1-bit 2-to-1 multiplexer determined by the ASIC technology and cell library. The incremental cost of implementing K_b by post-multiplexing is given by

$$\Delta c = 0.5 \times c_{MUX} \times (n_i + 1) \times (w_x + \max\{k_{a,max}, k_{b,max}\}). \quad (8)$$

Equation (9) assumes that no multi-bit multiplexer is available for reuse. If there exists a w_a -bit post-multiplexer from the implementation of previous 1-subvector, Δc for K_b will be reduced to

$$\Delta c = 0.5 \times c_{MUX} \times n_i \times \max\{w_a, w_x + k_{b,max}\}. \quad (9)$$

C. Proposed design algorithm

In practice, the shift amounts indicated by K for an s -bit SLS can possibly have 2^s combinations. Therefore, it is unrealistic to decide the optimal solution for each K and hence a heuristic approach is needed to decide if logic reuse or post-multiplexing should be adopted. The proposed algorithm for the design of SLS is described by the pseudo code in Fig. 6.

```

SLS( $w_x, s, K$ ) { FRLS = design_FRPS( $s$ ); // design  $s$ -bit FRLS
 $K' = \text{reduce}(K)$ ; //eliminate consecutive zeros
{core_subvector, 1-subvectors} = bisect( $K'$ );
 $SLS = \text{design\_FRLS}(\text{core\_subvector}, \text{FRLS})$ ; // design core subvector
 $K_a = \text{core\_subvector}$ ;
while (1-subvectors  $\neq \emptyset$ )
 $K_b = \text{min\_distance}(\text{1-subvectors}, K_a)$ ; //find nearest  $K_b$ 
 $n_i = \text{size}(K_b)$ ;  $\text{cost\_reuse} = \text{reuse}(K_a, K_b, n_i)$ ;
 $\text{cost\_post\_mux} = \text{post\_multiplexing}(K_a, K_b, n_i, w_x)$ ;
if  $\text{cost\_reuse} \leq \text{cost\_post\_mux}$ 
 $SLS = \text{design}(K_b, SLS, \text{reuse})$ ; // implement  $K_b$  using logic reuse
else  $SLS = \text{design}(K_b, SLS, \text{post\_mux})$ ; //implement  $K_b$  using post-mux
end if;
 $K_a = K_b$ ; remove( $K_b$ , 1-subvectors); end while //end of while loop
return  $SLS$ ; }

```

Fig. 6. Proposed design algorithm for SLS

The output of $SLS(w_x, s, K)$ is an s -bit SLS for a w_x -bit input signal with the shift amounts specified in K . The function **design_FRPS**(s) returns the optimized s -bit full-range programmable shifter. The **core_subvector** is realized with the substructure of a full-range logarithmic shifter **FRLS** by **design_FRL**. The remaining 1-subvectors are successively implemented in the while loop with K_a initialized to the **core_subvector**. The nearest 1-subvector K_b to K_a is determined by the function **min_distance**. The functions **reuse** and **post_multiplexing** implement K_b by the logic reuse and post-multiplexing techniques, respectively. The technique with the smaller incremental cost is adopted by the function **design** for K_b . K_a is then set to K_b for the next iteration and this process continues until the **1-subvectors** list is empty.

To compare the cost Δc for the two techniques, the ratio of c_{AND} to c_{MUX} for (7)-(9) is found to be 13.5:17.7 by simulation with the STM 65nm standard cell library. Using this ratio for the proposed design algorithm, the architecture of the SLS for $K = [1, 1, 1, 1, 1, 0, 0, 1]$ is shown in Fig. 7. The three logic components in the dotted box are the additional hardware used to implement $K_b = \{k_4\}$ by logic reuse technique. The bottom multiplexer is used to implement $K_b = \{k_7\}$ by post-multiplexing technique. If the designs are mapped to other cell library, the ratio of $c_{AND}:c_{MUX}$ will be re-evaluated. The new ratio will be used for the evaluation of (7) and (8) to adapt the cost function of the minimization to the target cell library.

Fig. 7. 8-bit SLS for $K = [1, 1, 1, 1, 1, 0, 0, 1]$

IV. LOGIC SYNTHESIS RESULTS AND DISCUSSION

A. SLS versus Conventional Shifters

Wordlength of 8-bit is assumed for the input signal. The design complexity can be scaled up with the increased wordlength. Since no sporadic shifter has been reported in the

literature, the savings of the proposed 8-bit and 16-bit SLS for different K vectors over the 8-bit and 16-bit full-range programmable shifters are evaluated. Nine 8-bit SLS test cases (1-9) and four 16-bit SLS test cases (10-13) are listed in Table I, where ‘Sporadicity’ indicates the fraction of unused shift amount over the complete shift range.

TABLE I. Test cases used for the design of 8-bit and 16-bit SLSs

Test Cases	K	Sporadicity
1	[1, 1, 1, 1, 0, 0, 0, 1]	37.5%
2	[1, 1, 1, 1, 1, 0, 0, 0]	37.5%
3	[1, 0, 0, 0, 1, 1, 1, 1]	37.5%
4	[0, 0, 0, 1, 1, 1, 1, 1]	37.5%
5	[1, 1, 1, 1, 0, 0, 1, 1]	25%
6	[1, 1, 1, 1, 1, 1, 0, 0]	25%
7	[1, 1, 1, 1, 0, 1, 1, 0]	25%
8	[1, 1, 1, 1, 1, 0, 0, 1]	25%
9	[1, 1, 1, 1, 1, 1, 1, 0]	12.5%
10	[1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]	68.75%
11	[1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1]	56.25%
12	[1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1]	50.00%
13	[1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0]	43.75%

All designs are described in Verilog, and synthesized with STM 65nm libraries. The synthesis area, delay and power results of 8-bit and 16-bit FRLSs [5] and mux-based FRBSs [7] are shown in Table II. The proposed 8-bit SLS for *Test Cases* 1 to 9 and the proposed 16-bit SLS for *Test Cases* 10 to 13 are shown in Table III and Table IV, respectively. The area (A) is the silicon area in μm^2 for ASIC. D and P are the critical path delay in ns and the average power consumption in μW , respectively. Monte Carlo power simulation is performed by applying epochs of random inputs until the error is bounded below 4% with 95% confidence interval.

TABLE II. ASIC synthesis results of 8-bit and 16-bit FRLS [5] and FRBS [7]

Full-range shifters		8-bit FRLS	8-bit FRBS	16-bit FRLS	16-bit FRBS
ASIC	Area (μm^2)	668.2	1565.1	1216.8	3837.8
	Delay (ns)	0.13	0.12	0.12	0.18
	Power (μW)	0.46	1.09	0.88	1.70

TABLE III. ASIC synthesis results of proposed 8-bit SLS

8-bit SLS test cases		1	2	3	4	5	6	7	8	9
ASIC	A (μm^2)	354	538.2	483.6	462.3	366.1	512.7	511.2	562.1	618.8
	D (ns)	0.11	0.13	0.13	0.13	0.11	0.13	0.13	0.11	0.11
	P (μW)	0.31	0.45	0.49	0.39	0.33	0.43	0.32	0.50	0.60

TABLE IV. ASIC synthesis results of proposed 16-bit SLS

16-bit SLS test cases		10	11	12	13
ASIC	Area (μm^2)	372.3	607.4	681.7	879.3
	Delay (ns)	0.12	0.12	0.11	0.14
	Power (μW)	0.39	0.46	0.52	0.65

The results show that the SLS can save some areas even for test cases with very low sporadicity. The average area savings over FRLS [5] and FRBS [7] are 33.18% and 73.24%, respectively. The area saving and sporadicity is strongly correlated. *Test Case* 9, with only one unused shift amount in K consumes the most area among all sporadic shifters, and its area is very close to that of FRLS. Owing to its high sporadicity, SLS *Test Case* 1 has the least area. It is realized by a 15-bit 2-1 post-multiplexer and a simple 4-bit FRLS, as shown in Fig. 5. On average, it saves about 10.22% of power over FRLS [5] and a significant 63.90% over FRBS [7]. On average, the SLS reduces the critical path delays by 11.66% and 10.84% over

FRLS [5] and FRBS [7], respectively. When the sporadicity is very low, e.g., 12.5% for test case 9, the proposed SLS has similar performance as the FRLS. To confirm that such exception happens only for the worst-case scenario, we randomly selected 10 additional test cases with sporadicity of 25%. Including the 4 test cases of the same sporadicity in Table I, the average area, delay and power savings of SLS over the 8-bit FRLS are 21.8%, 11.5% and 13.8%, respectively. As the performance of SLS generally improves with increasing sporadicity, the use of SLS can lead to significant area and power savings in most practical DSP applications where the sporadicities of the shifters are substantially greater than 25%, such as the programmable filters [3] and reconfigurable multipliers in DCT.

For 16-bit SLSs, *Test Cases* 10 and 11 possess the same number of 1s in K as *Test Cases* 1 and 9, respectively, which account for the marginally higher area, delay and power of these two 16-bit SLSs over the corresponding 8-bit SLSs. Although *Test Case* 12 has more 1s in K than *Test Cases* 10 and 11, the delay of this 16-bit SLS is slightly lower because its critical path consists of only a reduced-range two-level logarithmic shifter and a post-multiplexer.

B. DCT Architectures with SLSs

The RMs in reconfigurable DCT architectures suffer from complexity overhead caused by the programmable shifters. If SLSs are applied, as shown in Fig. 8, the complexity of RMs can be significantly reduced. In this section, the area, delay and power results of 16-point and 32-point DCT architectures designed with RMs and SLSs, are evaluated and compared with two recent methods [9] and [10]. The DCT coefficients are truncated to 9-bit and 8-bit wordlength is assumed for the input. Monte Carlo power simulation is performed using 400 randomly generated input vectors until the average error is bounded below 4% with 95% confidence interval. The areas in μm^2 , delays in ns and power consumption in μW are compared in Tables V. Based on the maximum delay of 3.6ns among all designs, 270MHz stimulus frequency is used.

Fig. 8. RM block design incorporating SLS

TABLE V. Areas, Delays and Power for 16-point and 32-point DCT

Designs		[9]	[10]	DCT with RM and SLS
16-point DCT	Area (μm^2)	47779.7	117613.6	25070.2
	Delay (ns)	1.688	3.578	0.979
	Total Power (μW)	14.97	15.85	9.84
32-point DCT	Area (μm^2)	169225.3	241129.2	50297.5
	Delay (ns)	2.629	3.528	1.058
	Total Power (μW)	17.16	32.90	16.66

The DCT architecture using RM and SLS is 42.0% and 70.6% faster than [9] and [10], respectively for the 16-point DCT and 59.8% and 70.0% faster than [9] and [10], respectively for the 32-point DCT. For [10] and SLS based DCT, the delays of their 32-point designs are very close to their

corresponding 16-point designs, because of their N parallel submodules. The delay of 16-point design of [9] is conspicuously shorter than its 32-point design. This exception implies that shorter delay can be achieved from its 16-point design than 32-point design under the timing-constrained optimization. It is noteworthy that SLS based DCT architectures are more area-efficient than [9] and [10] by 47.5% and 78.7%, respectively for 16-point DCT and 70.3% and 79.1%, respectively for 32-point DCT. For 16-point DCT computation, our proposed design conserves 34.3% and 37.9% of power over [9] and [10], respectively. For 32-point DCT computation, our proposed design is 2.9% and 49.4% more power efficient than [9] and [10], respectively. These area and power savings are contributed by the simplified RMs due to SLSs. Therefore, the area and power savings of applying SLS to DCT are often more significant than its standalone savings over the full-range programmable shifters.

V. CONCLUSION

A new design methodology for sporadic logarithmic shifter is proposed. The 1-subvectors are realized incrementally from a reduced full-range logarithmic shifter through reuse or post-multiplexing of existing substructures with hardwired shifts. The 8-bit and 16-bit SLSs designed by our proposed algorithm save about 22.51% to 73.24% of area and 10.22% to 63.90% power consumption for 12.5% to 68.75% of unused shift amounts compared with the use of full-range programmable shifters. The savings are more prominent in DSP implementations such as reconfigurable DCT architectures, where each RM can be substantially simplified by SLS.

REFERENCES

- [1] Y. Hilewite and R. B. Lee, "A new basis for shifters in general-purpose processors for existing and advanced bit manipulations," *IEEE Trans. Comput.*, vol. 58, no. 8, pp. 1035–1048, Aug. 2009.
- [2] D. Oh and K. K. Parhi, "Low-complexity switch network for reconfigurable LDPC decoders," *IEEE Trans. VLSI Syst.*, vol. 18, no. 1, pp. 85–94, Jan. 2010.
- [3] J. Chen, C. H. Chang, F. Feng, W. Ding, and J. Ding, "Novel Design Algorithm for Low Complexity Programmable FIR Filters Based on Extended Double Base Number System," *IEEE Trans. Cir. Syst. I*, vol. 62, no. 1, pp. 224–233, Jan. 2015.
- [4] J. Park, W. Jeong, H. M. Meimand, Y. Wang, H. Choo and K. Roy, "Computation sharing programmable FIR filter for low-power and high-performance applications," *IEEE J. Solid-state Cir.*, vol. 39, no. 2, pp. 348–357, Feb. 2004.
- [5] J. Y. S. Low and C. H. Chang, "A VLSI efficient programmable power-of-two scaler for $\{2^n-1, 2^n, 2^{n+1}\}$ RNS," *IEEE Trans. Cir. Syst. I*, vol. 59, no. 12, pp. 2911–2919, Dec. 2012.
- [6] J. Chen and C. H. Chang, "High-level synthesis algorithm for the design of reconfigurable constant multiplier," *IEEE Trans. CAD*, vol. 28, no. 12, pp. 1844–1856, Dec. 2009.
- [7] R. Rajalakshmi and P. A. Priya, "Design and analysis of a 4-bit low power universal barrel-shifter in 16nm FINFET technology," *IEEE Inter. Conf. Advanced Comm. Control and Comput. Tech.*, pp. 527–532, Tasilnadu, India, May 2014.
- [8] P. Tummeltshammer, J. C. Hoe and M. Puschel, "Time-multiplexed multiple-constant multiplication," *IEEE Trans. CAD*, vol. 26, no. 9, pp. 1551–1563, Sep. 2007.
- [9] P. K. Meher, Y. P. Sang, B. K. Mohanty, K. S. Lim and C. Yeo, "Efficient Integer DCT Architectures for HEVC," *IEEE Trans. on Circuits Syst. Video Technol.*, vol. 24, no. 1, pp. 168–178, Jan. 2014.
- [10] S. An and C. Wang, "Recursive algorithm, architectures and FPGA Implementation of the Two-dimensional Discrete Cosine Transform," *IET Image Processing*, vol. 2, no. 6, pp. 286–294, Dec. 2008.