New Design Algorithm for Low Complexity Programmable Shifters by Integer Partition

Yujia Wang
Undergraduate student, Freshman
Singapore University of Technology and Design (SUTD)
Singapore
yujia_wang@mymail.sutd.edu.sg

Jiajia Chen
Pillar of Engineering Product Development (EPD)
Singapore University of Technology and Design (SUTD)
Singapore
jiajia_chen@sutd.edu.sg

Abstract—**Conventional programmable shifters are widely adopted currently in many computer arithmetic manipulation applications such as residue number system, decoders and etc. However, it consumes a considerable amount of power and dominates the overall complexity of the entire circuit in some applications. More importantly, such power and circuit area consumption become redundant when only part of the shifting amount is required in the design. Unfortunately, the structure of conventional programmable shifter is relatively fixed. Therefore, a new design is needed to reduce this redundancy with partial programmable shifters which reduces the complexity and power consumption. As there are no existing approach in the literature to solve the problem to our best knowledge, this paper presents an illuminating algorithm by adopting integer partition technique to dynamically generate the simplest numerical combination of all the required shifting amounts. A new structure of partial programmable shifters is designed based on the multiplexer control signal Array. Such design algorithm has been validated by using 8-bit programmable shifters benchmark. The synthesis results show that the proposed design reduces the logic complexity and dynamic power by 41.9% and 32.6% over the conventional logarithm shifters, similarly, 67.3% and 51.6% over the base-line multiplexer-based conventional barrel shifters.**

Keywords- Programmable Shifter, digital IC Design, Digital Signal Processing.

I. Introduction

Programmable shifters are a commonly adopted component in many bit manipulation [1] and specific digital signal processing circuits, such as coordinate rotation digital computer [2], programmable finite impulse response (FIR) filters [3], residue number system (RNS) [4] and low-density parity-check (LDPC) decoder [5]. Existing designs [2] [3] [4] [5] employed the conventional *w*-bit barrel or logarithm shifters, which can perform shifting operations with a shifting amount ranges from *0* to *w-1* bit, The programmability is achieved by the control signal of log_2w bit. The entire circuit's complexity may increase dramatically due to these expensive shifters to achieve the desired numerical computations. In conventional structure, all shifting amounts capability are designed independent of the actual desired shifting amounts in specific applications, which is the primary reason for its high complexity and power consumption. Therefore, it is crucial to reduce the complexity and optimize the conventional structure to design more power and area efficient programmable shifters.

There are many applications where not all shifting amounts are required. Therefore, we propose our design to replace the traditional full programmable shifters (FPS) by partial programmable shifters (PPS) which can fulfill the shifting requirements. To design a low power and low complexity PPS, in this paper, we formulate our approach as a new algorithm using integer partition of the required shifting amounts.

II. Shifting Amount Array Evolution

A. *Evolving the shifting amount array using Integer Partition*

The required shifting amount of one *w*-bit programmable shifter can be expressed as a shifting amount array (SAA), in which all required shifting amount are initialized. Apparently the number of elements inside the array should not exceed *w*, which is the maximum number of shifting amounts performed by *w*-bit programmable shifter. For example, an 8-bit PPS with shifting amount 0-bit, 1-bit, 2-bit, 3-bit, 4-bit and 5-bit can be expressed as follows:

$$SAA(S_i)= \{0, 1, 2, 3, 4, 5\}; \; w=8 \tag{1}$$

It is obvious that when all shifting amounts are present, the conventional full programmable shifter should be adopted. On the other hand, partial programmable shifter will be used if the shifting amount is less than *w*.

To proceed with the initial SAA for simpler PPS design, integer partition technique is adopted which is a mathematical combinatory method to decompose an integer *n* as the sum of a number of positive integers. We define a function *p(n)* represents the partition of a given number *n*. In our method, each shifting amount is partitioned and is expressed as a summation of the essential shifting amounts (ESA) including the previously generated shifting amounts S_i and elements {1} and {2} being the basic summands, which can be expressed as:

$$p(S_n) = \sum_{i=1}^{N} S_i + C \tag{2}$$

where *N* is the number of existing shifting amounts and *C* is the set of {1} and {2}. Using the same example given in (1), our proposed partition function would be:

$p(S_1)=1=1$ $\qquad$ $p(S_2)=2=2$

$p(S_3)=3=1+2$ $\qquad$ $p(S_4)=4=2+2$

$p(S_5)=5=3+2=p(S_3)+2=1+2+2$

In this example, it is easy to conclude that only one 1-bit shifting and two 2-bit shifting are required to produce all shifting amounts stored in SAA. It should be highlighted that, although the intermediate steps to partition one shifting amount might be different, the ultimate result after partition should be the same because of the addition commutative property. Therefore, all the required shifting amounts in SAA can be expressed as linear combinations of these essential shifting amounts.

The evolved SAA with all shifting amounts partitioned gives fundamental information about the architecture of the partial programmable shifter. It is a combination of all the partition results of the required shifting amounts:

$$SAA=\{S_0+p(S_1)+\ldots+p(S_{n-1})\} \quad (3)$$

B. Mapping the Control Bit to SAA

As one may notice, the PPS design based on the evolving SAA has a different control signal sequence from the conventional FPS when shifting the same amount. Therefore, to compensate the difference, a simple method to calculate the multiplexer selection digits in our proposed designs is introduced below.

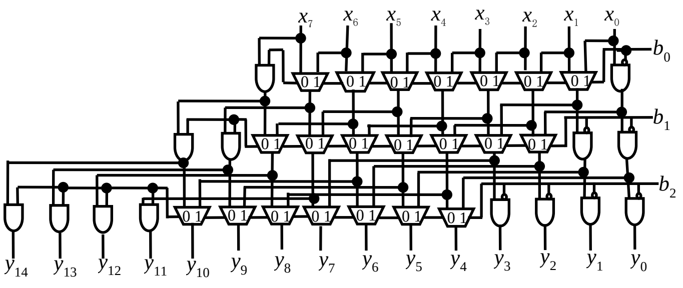
Suppose SAA has n elements, the shifting amount S_i can be expressed as:

$$S_i = b_0 ESA[0] + b_1 ESA[1] + \ldots + b_{n-1} ESA[n-1] = \sum_{j=0}^{n-1} ESA[j] \quad (4)$$

where the coefficient b_0, b_1,… are the new control signal sequence required with b_0 as the LSB. These coefficients will be used to select the input of the multiplexers at different stage of the proposed PPS structure. For example, if we have ESA={1, 2, 2}, in order to shift 4-bit, the control signal would be designed as 011 (0 ESA[0]+1 ESA[1]+1 ESA[2]). Similarly, 111 should become the control signal for shifting 5-bit (1 ESA[0]+1 ESA[1]+1 ESA[2]).

III. Proposed Design of Low Complexity PPS

An architecture of logarithm shifter is studied replacing original 2-1 multiplexers with 2-input AND gate at some output bit and intermediate signals, as shown in Figure 1. It reduced the costs of a full programmable shifter. Our design presented in this paper is developed based on this architecture below.



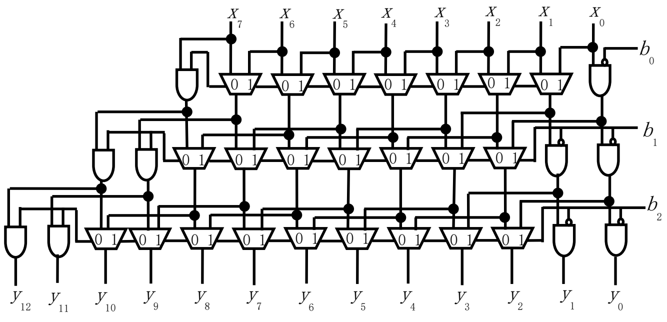
x_7 x_6 x_5 x_4 x_3 x_2 x_1 x_0
b_0
b_1
b_2
0 1
y_{14} y_{13} y_{12} y_{11} y_{10} y_9 y_8 y_7 y_6 y_5 y_4 y_3 y_2 y_1 y_0

Figure 1. 8-bit FPS structure

For the full programmable shifter, the evolved SAA is *{1,2,4}* which can form the linear combination of any shifting amounts between 0-bit and 7-bit.

The design process start with listing all the required shifting amounts in the initial SAA and perform the integer partition for every element in it. As mentioned in Section II, the partition is to express the current shifting amount (S_n) using the previous shifting amount (S_{n-1}) in the array combined with basic partition summands. For 8-bit programmable shifter, only two basic partition summands are needed which are {1} and {2}. However, more basic summands must be incorporated for programmable shifters that shift larger amount, such as 16-bit programmable shifter, {4} is added as a basic summand.

To illustrate, we take an example that has been examined in section II in which the required shifting amounts are SAA={0,1,2,3,4,5} and $w=8$. By performing the integer partition, we separate each shifting amount and eliminate the common shifting amount. The SAA is obtained as: $SAA=\{0+ p(1) + p(2) + p(3) + p(4) + p(5)\}=$ {1,2,2}. With this evolved SAA, the structure of the PPS can be developed with three control bits. Starting from the LSB, the three control bits control 1-bit shifting, 2-bit shifting and 2-bit shifting respectively. The proposed design architecture is shown in Figure 2.



X_7 X_6 X_5 X_4 X_3 X_2 X_1 X_0
b_0
b_1
b_2
0 1
y_{12} y_{11} y_{10} y_9 y_8 y_7 y_6 y_5 y_4 y_3 y_2 y_1 y_0

Figure 2. 8-bit PPS of SAA={1,2,2}

In the next stage, it is simply mapping the shifting amount to SAA using ESA. If the element in the SAA is a summand of a shifting amount, the corresponding control signal should be set as 1, otherwise it should be set as 0. The control signals correspond to the array starts from the LSB to the MSB.

In the example, we map the shifting amount *{1,2,3,4,5}* to SAA as follows(LSB-MSB):

shift 0-bit→000	*shift 1-bit→100*	*shift 2-bit→010*
shift 3-bit→110	*shift 4-bit→011*	*shift 5-bit→111*

Obviously, different structures of partial programmable shifter will be adopted based on the specific shifting amounts required in different applications, but the overall algorithm and procedure are the same. The pseudo code to summarize the proposed PPS design algorithm is shown in Figure 3:

```
PPS(w, S){
    PPS=∅; //initialized PPS design
    SAA{S_i}=Shifting_Amount_Array; //initialize shifting amount array
    CSA={S_i: 0} (for S_i in S) //initialize control signal array
    If (SAA{S_i}=Set(w)) {
        PPS=design_FPS(w); // if shifting full range, adopt the FPS structure
        }
        else {
            for ($S_i$ in $SAA\{S_i\}$) {
                $p(S_i)$=integer_partition($S_i$, $SAA\{S_{i-1}\}$, {2}, {1}) //perform integer partition for every element in the array
            }
            $SAA\{S\}$=simplify($S_0+p(S_1)+\ldots+p(S_{n-1})$)   //eliminate the common summands to get simplified shifting amount array.
            PPS=design_PPS($A\{S\}$)  }//design partial programmable shifter
        for ($S_i$ in $SAA\{S\}$) {
            $S_i=b_0\ ESA[0]+b_1\ ESA[1]+\ldots$   / / mapping shift amount to control signal
            $CSA\{S_i\}$=map($b_0b_1\ldots b_i$) //store control signal information}
    return PPS, CSA
}
```

Figure 3. Proposed PPS design algorithm

The function **PPS**(*w*, *S*) is to generate the structure of a *w*-bit programmable shifter with a given shifting amount of *S*. First of all, the program generates a shifting amount array, $SAA\{S_i\}$, with all the shifting amounts in *S* and examine whether all the shifting amounts are required. If so, then conventional logarithm shifter as shown in Figure 1 is generated using **design_FPS**(*w*) function. Otherwise, the proposed algorithm will use **integer_partition**(S_i, $A\{S_{i-1}\}$, *{2}*, *{1}*) function to perform integer partition for every shifting amount. The function **simplify**($S_0+p(S_1)+\ldots+p(S_{n-1})$) selects all the essential shifting amounts (ESA) and updates the SAA. The proposed design of PPS is generated using **design_PPS**($A\{S\}$) based on the evolved SAA. After performing linear combination of the required shifting amounts, **map**($b_0b_1\ldots b_i$) function maps the control bits to each shifting amount and stores in CSA. The PPS function returns both the architecture of the designed PPS and CSA.

IV. Synthesis Results and Discussion

The proposed PPS algorithm with different required shifting amounts is implemented in Matlab. In the experiment, six 8-bit PPS with different shifting amounts combinations listed in Table I are designed together with two existing conventional full programmable shifters mentioned in [4] and [6], which are conventional logarithmic shifter (LS) and base-line multiplexer-based conventional barrel shifters (BS). The input is assumed of 8-bits word length which is often used in ADC resolution. All the programmable shifter structures are written in Verilog codes and synthesized by Synopsys Design Compiler using STM 65*n*m standard cell libraries.

TABLE I. Shifting Amounts Required for PPS Design

Design	Original SAA
Exp. 1	SAA={0,1,2,3}
Exp. 2	SAA={0,1,2,3,4,5}
Exp. 3	SAA={0,1,2,3,5,6}
Exp. 4	SAA={0,1,2,3,6,7}
Exp. 5	SAA={0,1,2,3,4,5,6}
Exp. 6	SAA={0,1,2,3,4,7}

Table II lists synthesized silicon areas in μm^2, critical path delays in *n*s as well as the total dynamic power in *m*W between FPSs and PPSs designed in Table I. From Table II, the proposed PPS reduces the silicon area on average of 41.9% and 67.3% compared with conventional logarithm shifter and multiplexer-based conventional barrel shifter respectively. On the other hand, there is a 22.5% and 17.1% decrease on average of path delay compared between proposed PPS and conventional designs respectively. Dynamic power consumption is another important performance factor. Based on the data obtained, the proposed PPS saves on average 32.6% and 51.6% over conventional logarithm shifter and multiplexer-based conventional barrel shifter. As expected, the improvement is more significant when the shifting amounts are fewer since more redundancy will be eliminated from the proposed PPS architecture. To further illustrate, the area-time (AT) complexity of all the designs, which considers the hardware complexity and speed, are plotted in Figure 4.

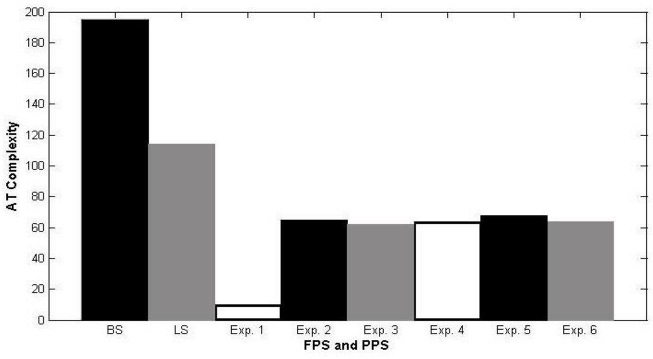


Figure 4. AT complexity comparison of 8-bit programmable shifters

V. Conclusion

Programmable shifters are wildly used today in application-specific digital circuits but consumes large amount of complexity and power. In this paper, a new algorithm is proposed to further simplify the structure of programmable shifters when not all shifting amounts are required in the design. By creating a Shifting Amount Array from performing integer partition to the required shifting amounts, a new architecture of the partial programmable shifter can be designed. In addition, similar algorithm is used for mapping the shifting amounts with the actual control signal inputs. This new approach has been proven more effective with silicon area reduction of 41.9% and 67.3% over the conventional logarithm shifters and base-line multiplexer-based conventional barrel shifters respectively. Conversely, 32.6% and 51.6% of the total dynamic power is saved comparing to the conventional logarithm shifters and base-line multiplexer-based conventional barrel shifters. In addition, this design algorithm provides a new inspiration for the simplification of digital circuits.

TABLE II. SYNTHESIS RESULTS FOR SILICON AREA, DELAY AND POWER FOR 8-BIT PPS COMPARING WITH CONVENTIONAL FPS

Design		Synthesis Results and Power Simulation Results					
		Silicon Area (μm^2)	***Reduced Area over BS / LS***	***Path Delay (ns)***	***Reduced Delay over BS / LS***	***Total Dynamic Power (mW)***	***Reduced Power over BS / LS***
BS		1565.12		0.1243		5.3907	
LS		857.5		0.133		3.8694	
PPS	Exp. 1	142.5	90.8% / 83.4%	0.0686	44.8% / 48.4%	1.0021	81.4% / 74.1%
	Exp. 2	568.35	63.7% / 33.7%	0.1138	8.4% / 14.4%	2.5897	52.0% / 33.1%
	Exp. 3	619.79	60.4% / 27.7%	0.1004	19.2% / 24.5%	3.3065	38.7% / 14.6%
	Exp. 4	562.12	64.1% / 34.4%	0.1133	8.9% / 14.8%	2.7285	49.4% / 29.5%
	Exp. 5	618.79	60.5% / 27.8%	0.1090	12.3% / 18.1%	3.3065	38.7% / 14.6%
	Exp. 6	562.12	64.1% / 34.4%	0.1133	8.9% / 14.8%	2.7285	49.4% / 29.5%
Average:		*Average Reduced Area over:*	BS: 67.3% LS: 41.9%	*Average Reduced Delay over:*	BS: 17.1% LS: 22.5%	*Average Reduced Power over:*	BS: 51.6% LS: 32.6%

ACKNOWLEDGMENT

This 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 (SUTD). We also gratefully acknowledge the support and contribution from Ms. Juan Zhao and Mr. Sachin Kumar to this work.

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. on Computers*, vol. 58, no. 8, pp. 1035–1048, Aug. 2009.

[2] L.Vachhani, K. Sridharan and P. K. Meher, "Efficient CORDIC algorithms and architectures for low area and high throughput implementation," *IEEE Transactions on Circuits and Systems II*, vol. 56, no. 1, pp. 61–65, Jan. 2009.

[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 Transactions on Circuits and Systems I*, vol. 62, no. 1, pp. 224–233, Jan. 2015.

[4] 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 Transactions on Circuits and Systems I*, vol. 59, no. 12, pp. 2911–2919, Dec. 2012.

[5] D. Oh and K. K. Parhi, "Low-complexity switch netwlrk for reconfigurable LDPC decoders," *IEEE Trans. on VLSI Systems*, vol. 18, no. 1, pp. 85–94, Jan. 2010.

[6] R. Rajalakshmi and P. A. Priya, "Design and analysis of a 4-bit low power universial barrel-shifter in 16nm FINFET technology," *IEEE Inter. Conf. Advanced Communication Control and Computing Technologies*, pp. 527-532, Tasilnadu, India, May 2014.