VLSI implementation of the modified sign-error LMS adaptive algorithm

Ming Liu^{a)}, Mingjiang Wang^{b)}, De Liu, and Boya Zhao

Harbin Institute of Technology, Shenzhen, Xili University Town,
Shenzhen 518055, Guangdong, China

a) lm_hit_1986@126.com

b) mjwang@hit.edu.cn

Abstract: Motivated by reduction of computational complexity, this work develops a delay-optimized VLSI architecture of the adaptive filter based on the modified sign-error LMS (MSLMS) algorithm. The proposed algorithm uses a three-level quantization strategy applied to the modified sign function containing a threshold parameter. The amount of computation of the proposed architecture is not only less than half of the traditional structure, but also the convergence characteristic is close to that of the LMS algorithm. The fine-grained dot-product unit and multiple-input-addition unit are adopted to reduce the latency of critical path. From the ASIC synthesis results we find that the proposed design for filter length 8-tap has roughly 31% less power and 53% less area-delay-product (ADP) than the best of existing structures.

Keywords: modified sign-error LMS algorithm, fine-grained, dot-product unit, multiple-input-addition unit

Classification: Integrated circuits

References

- [1] P. K. Meher and S. Y. Park: "Critical-path analysis and low-complexity implementation of the LMS adaptive algorithm," *IEEE Trans. Circuits Syst. I, Reg. Papers* **61** (2014) 778 (DOI: [10.1109/TCSI.2013.2284173](https://doi.org/10.1109/TCSI.2013.2284173)).
- [2] N. R. Shanbhag and K. K. Parhi: *Pipelined Adaptive Digital Filters* (Kluwer Academic Publishers, 1994) 1st ed. 274.
- [3] K. Matsubara, *et al.*: "A new pipelined architecture of the LMS algorithm without degradation of convergence characteristics," *Proc. IEEE ICASSP* (1997) 4125 (DOI: [10.1109/ICASSP.1997.604854](https://doi.org/10.1109/ICASSP.1997.604854)).
- [4] S. C. Douglas, *et al.*: "A pipelined LMS adaptive FIR filter architecture without adaptation delay," *IEEE Trans. Signal Process.* **46** (1998) 775 (DOI: [10.1109/78.661345](https://doi.org/10.1109/78.661345)).
- [5] P. K. Meher and M. Maheshwari: "A high-speed FIR adaptive filter architecture using a modified delayed LMS algorithm," *ISCAS* (2011) 121 (DOI: [10.1109/ISCAS.2011.5937516](https://doi.org/10.1109/ISCAS.2011.5937516)).
- [6] Y. Yi, *et al.*: "High speed FPGA-based implementations of delayed-LMS filters," *J. VLSI Signal Process.* **39** (2005) 113 (DOI: [10.1023/B:VLSI.0000047275.54691.be](https://doi.org/10.1023/B:VLSI.0000047275.54691.be)).
- [7] L.-K. Ting, *et al.*: "Virtex FPGA implementation of a pipelined adaptive LMS predictor for electronic support measures receivers," *IEEE Trans. Very Large*

- Scale Integr. (VLSI) Syst. **13** (2005) 86 (DOI: [10.1109/TVLSI.2004.840403](https://doi.org/10.1109/TVLSI.2004.840403)).
- [8] S. Y. Park and P. K. Meher: “Low-power, high-throughput, and low-area adaptive FIR filter based on distributed arithmetic,” IEEE Trans. Circuits Syst. II, Exp. Briefs **60** (2013) 346 (DOI: [10.1109/TCSII.2013.2251968](https://doi.org/10.1109/TCSII.2013.2251968)).
- [9] G. Long, *et al.*: “The LMS algorithm with delayed coefficient adaptation,” IEEE Trans. Acoust., Speech, Signal Process. **37** (1989) 1397 (DOI: [10.1109/29.31293](https://doi.org/10.1109/29.31293)).
- [10] L.-D. Van and W.-S. Feng: “An efficient systolic architecture for the DLMS adaptive filter and its applications,” IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process. **48** (2001) 359 (DOI: [10.1109/82.933794](https://doi.org/10.1109/82.933794)).
- [11] D. Liu, *et al.*: “Delay-optimized floating point fused add-subtract unit,” IEICE Electron. Express **12** (2015) 20150642 (DOI: [10.1587/elex.12.20150642](https://doi.org/10.1587/elex.12.20150642)).
- [12] M. Lotfizad and H. S. Yazdi: “Modified clipped LMS algorithm,” EURASIP J. Appl. Signal Process. **2005** (2005) 310205 (DOI: [10.1155/ASP.2005.1229](https://doi.org/10.1155/ASP.2005.1229)).
- [13] S. Srinivasan, *et al.*: “Split-path fused floating point multiply accumulate (FPMAC),” Proc. Symp. Comput. Arith. (2013) 17 (DOI: [10.1109/ARITH.2013.32](https://doi.org/10.1109/ARITH.2013.32)).

1 Introduction

In the digital signal processing field, adaptive filter has been used widely in speech process [1, 2], image process, communication system, measure system, etc. It is generally known that the least-mean-square (LMS) algorithm has two obvious characteristics including high performance and simple calculation. The LMS algorithm is well suited to software-based simulation but is not applicable to hardware implementation [3, 4]. In practical applications, the complexity of the adaptive algorithm is as simple as possible under the premise of guaranteeing the required performance [5]. We use the modified SLMS algorithm to result in a simple structure and a good convergence characteristics.

Some researchers have done a lot of work on the systolic architectures of the DLMS algorithm in [6, 7, 8]. Since the number of registers of systolic architectures is more than that of conventional architecture in [9], a more power consumption and larger area will be generated. P. K. Meher have proposed an architecture of DLMS algorithm with 2 adaptation-delay. Van and Feng [10] have proposed a systolic structure using large processing element (PE) in order to generate less adaptation delay. Y. Yi et al. [6] have proposed a retimed 8-tap adaptive filter with the TDF-RDLMS architecture that has the critical path of multiplier.

In order to compare the speed of various computing components designed, we use logic level to estimate the latency of different circuits without limitation of manufacturing technology [11]. We define 2 logic levels as the XOR gate delay which is two times that of the AND/OR gate. Fig. 1 shows an 8-bit *Kogge-Stone* adder structure which has a delay of 10 logic levels consisting of 1 level of $\diamond$ (2LL), 3 ($\log_2 8$) levels of $\blacksquare$ ($3 * 2LL = 6LL$) and 1 level of $\circ$ (2LL). The delay of N-bit *Kogge-Stone* adder is roughly $(2 + \log_2 N + 2)$ logic levels.

The rest of chapters are arranged as follows. In Sec. 2, we give the background of simplified adaptive algorithm such as sign-error LMS algorithm. The detailed derivation of the proposed structure is given in Sec. 3. Sec. 4 presents the results of

Fig. 1. The Kogge-Stone adder (8-bit).

logic synthesis and compares with other recent works. The conclusions are presented in Sec. 5.

2 The proposed modified sign-error LMS algorithm

Before clarifying the content of the new algorithm, we firstly introduce the sign-error LMS (SLMS) algorithm as background knowledge. The SLMS adaptive filtering algorithm is given by

$$y(n) = \sum_{k=0}^{N-1} x(n-k)w_k(n) \quad (1)$$

$$W(n+1) = W(n) + \mu X(n) \operatorname{sgn}\{e(n)\}, \quad (2)$$

$$e(n) = d(n) - W^T(n)X(n), \quad (3)$$

where N is equal to the filter length, $e(n)$ is the error signal of the SLMS algorithm, μ is the step size for adaptation of the weight vector, $d(n)$ is the desired signal, $W(n)$ and $X(n)$ which are the filter weight vector and the input vector respectively are given by

$$X(n) = [x(n), x(n-1), \dots, x(n-N+1)]^T \quad (4)$$

$$W(n) = [w_0(n), w_1(n), \dots, w_{N-1}(n)]^T, \quad (5)$$

where $[X]^T$ represents the transpose of vector X . From equation (2), we can see that the sign function is applied to the error signal $e(n)$. In the following we will give the equation for sign function.

$$\operatorname{sgn}(x) = \begin{cases} 1; & x > 0 \\ 0; & x = 0 \\ -1; & x < 0 \end{cases} \quad (6)$$

The computation of SLMS algorithm is greatly reduced especially when $e(n)$ is equal to zero, and at the same time this algorithm does not introduce any operations [12]. The prominent feature of SLMS algorithm is utilization of the quantization scheme to reduce the computation complexity.

In order to improve the convergence characteristics, we present the modified sign-error LMS (MSLMS) algorithm. The new algorithm possesses three-level quantization strategy that we use a modified sign function to limit the $e(n)$. The modified sign function is given by

$$\text{msgn}(x) = \begin{cases} +1; & x \geq \delta \\ 0; & -\delta < x < \delta \\ -1; & x \leq -\delta \end{cases} \quad (7)$$

Applying the modified sign function to the $e(n)$, we can get the new algorithm, namely, MSLMS algorithm. The equation (2) is changed into as follows:

$$\mathbf{W}(n+1) = \mathbf{W}(n) + \mu \mathbf{X}(n) \text{msgn}\{e(n)\} \quad (8)$$

The MSLMS algorithm makes a three level quantization operation on the $e(n)$. Notice that when $-\delta < e(n) < \delta$, this algorithm does not involve multiplication operations. When $e(n)$ is equal to other values, this algorithm involves only one multiplication operation. Equation (7) can be changed into another form when error signal is greater than quantization threshold (δ). The new modified sign function applying to $e(n)$ is described as

$$\text{msgn}\{e(n)\} = e(n)/|e(n)|. \quad (9)$$

The improved tap-weight update equation is given by

$$\mathbf{W}(n+1) = \mathbf{W}(n) + \frac{\mu}{|e(n)|} \mathbf{X}(n)e(n). \quad (10)$$

From equation (10), we can get a timing-varying step-size LMS algorithm. So the convergence characteristics of the MSLMS algorithm is related to the quantization threshold (δ). Beyond that, the method using three-level quantization scheme of the MSLMS algorithm can reduce computation greatly.

3 The proposed architecture

Here we will give the derivation process for the proposed architecture. The proposed architecture uses both the multiple-input-addition unit and dot-product unit to reduce the delay of the critical path. Note that, in the following, we set the rules that t_{add} and t_{mul} are the delay consumed to the addition and multiplication operation.

3.1 Multiple-input-addition

The easiest way to achieve the sum of l N-bit words is that we use $l-1$ carry-propagate adders (CPA). Owing to use of *Koggle-Stone* structure for CPA, the delay of Fig. 2(a) is $\{2 + (\log_2 4) * 2 + 2\} * 3 = 24$ logic levels [13]. In order to reduce the delay further, the parallel scheme is adopted as shown in Fig. 2(b) which has a delay of $8 * 2 = 16$ logic levels. Not only the area of these two architectures is too large, but also the operation speed is slow. A more popular technique is using [3:2] adder implemented in parallel. Since a [3:2] adder has two levels of XOR gate, it has a delay of $2 * 2 = 4$ logic levels. From Fig. 2(c), we can see that the circuit has 2 levels of *carry-save adder* (CSA) and 1 level of CPA. Although CSA can be implemented in many ways, we usually use full adder called [3:2] adder with three input signal. The logic levels of the structure in Fig. 2(c) is roughly $(4 + 4 + 8) = 16L$.

An optimized organization mode for CSA is the *Wallace-tree* architecture as shown in Fig. 2(d). The amount of operand of each level is two-thirds of upper

Fig. 2. Multiple-input adders.

level in *Wallace-tree*. The N-operand adder needs $\lceil \log_{3/2} N \rceil$ levels CSA in *Wallace-tree* mode and one level of CPA, where $\lceil x \rceil$ represents the smallest integer greater than x . The operating efficiency is highest when the number of operands is equals to 3, 4, 6, 9, 13, 19 and so on.

Although the *Wallace-tree* structure has a relatively faster speed, it is characterized by increasing the consumption of area. In order to gather the data with the same weight, it needs more routing resources. Another drawback of *Wallace-tree* structure is lack of the symmetry and the regularity.

Fig. 3. The structure of [4:2] compressor.

In VLSI design, the issue of regular structure is particularly important. Currently the most widely used arithmetic unit for multiple-input-addition is the [4:2] compressor. Actually, a [4:2] compressor is an encoder with 5 inputs and 3 outputs as shown in Fig. 3(a). A novel structure of [4:2] compressor is implemented by 2-to-1 multiplexers (MUXs) and XOR logic circuits. From Fig. 3(a),

we can see that the [4:2] compressor has a delay of $2 * 3 = 6$ logic levels. There is an example for 4 4-bit addition in Fig. 3(b). For N-operands-addition, it needs only $\lceil \log_2 N/2 \rceil$ levels of [4:2] compressor.

We can learn from the above analysis that multiple-input-addition using the [4:2] compressor tree has the least delay and more regular layout. In addition, according to the research of Carnegie Mellon University, the power of multiplier using the [4:2] compressor tree structure is the least.

3.2 The fine-grained structure of Dot-product-unit

In order to be more convenient to reduce the delay of the critical path, we have designed a fine-grained Dot-product unit. Fig. 6 shows a complete structure of the Dot-product unit composed of 4 parts. The function of this Dot-product unit is to perform the operation of $A1 * B1 + A2 * B2 + A3 * B3 + A4 * B4$.

Before clarifying the proposed architecture, we firstly introduce a modified Booth multiplier used to improve circuits speed. Fig. 4(a) shows a modified Booth multiplier structure consisted of partial product generation (PPG) element, [4:2] compressor tree element and CPA element. Because there was no special advantages in delay and area for conventional Booth coding mode, we propose a novel Booth coding structure as shown in Fig. 4(c) and Fig. 4(d). The data width in proposed design is 16 bits. Fig. 5 shows the conventional architecture of Dot-product unit proposed by P. K. Meher in [1].

Fig. 4. The Booth multiplier.

1. The delay analysis for the modified Booth decoding multiplier:

- *Partial Product Generation.* The new Booth coding structure is used for this block. Since this block contains 3 levels of basic gate level circuits, the delay of this module is about four logic levels—4L.
- *[4:2] Compressor Tree.* There are 9 partial products for the signed multiplication. This module contains one level of half adder and two levels of

Fig. 5. The conventional structure of Dot-product unit.

[4:2] compressor. Because the delay of half adder is only one level XOR gate, it has a delay of two logic levels. As I mentioned, the delay of one level of [4:2] compressor is six logic levels. We can get the delay for this module is $(2 + 6 * 2)$ logic levels—**14L**.

- *The CPA Module.* This module is a 32-bit carry-propagate adder implemented by *Kogge-Stone* structure mentioned above. The delay of this module is roughly $(2 + 2 * \log_2 32 + 2)$ logic levels—**14L**.

2. The architecture of Dot-product unit proposed by P. K. Meher [1]:

- *The Four Multipliers Module.* This module is the first level of the Dot-product unit. Since the four multipliers perform the operation in parallel, the delay of this module is that of only one multiplier. According to the mentioned above, this module has a delay of 32 logic levels—**32L**.

Fig. 6. The structure of Dot-product unit.

- *The Two Adders Module.* This module has two adders performing operations in parallel. Since the data width is 32 bits, the delay of one addition implemented by *Koggle-Stone* algorithm is about 14 logic levels—**14L**.
 - *The Adder Module.* Because there is only one adder in this module, it has a delay of 14 logic levels—**14L**.
3. The proposed architecture of Dot-product unit:
- *The PPG Module.* Since this module has four identical partial product generation with the same weight, this module has the same delay as the one PPG unit, that is 4 logic levels—**4L**.
 - *The [4:2] Compressor Tree Module.* Because every PPG module generates 9 partial products for signed operation, there are 36 partial products generated from the upper module. Fig. 6 shows the structure of the [4:2] compressor tree consisting of five levels of [4:2] compressor. Because every [4:2] compressor has a delay of 6 logic levels, the delay for this module is 30 logic levels—**14L**.
 - *The CPA Module.* The data width for the two inputs of the CPA is 32-bit. The delay of this module is roughly $(2 + 2 * \log_2 32 + 2)$ logic levels—**14L**.

According to the detailed analysis of the previous, we can get the conclusion that the delay of proposed Dot-product unit is less than that of P. K. Meher in [1]. The Dot-product unit presented by P. K. Meher has a delay of 60 logic levels, and the proposed architecture has a delay of 48 logic levels. The final level of [4:2] compressor can be replaced by [3:2] adder, which is caused by unaligned partial products in the actual circuit. The amount of ultimate logic levels for Dot-product unit designed can be reduced to 46.

3.3 Implementation of modified SLMS algorithm

There is an assumption that the step size μ is a power of two, so we can replace multiplication operation with shift operation. Fig. 7 shows an 8-tap adaptive filter structure of the proposed design divided into 2 steps.

We assumed that the bit-width is equal to 16, such as the input data $x(n)$, desired data $d(n)$, the weight signal $w(n)$ and so on. In the first step, we firstly give the architecture of the MSLMS algorithm. This architecture with no adaptation delay uses a fine-grained dot-product unit. In this figure, the critical path in the first step is given by

$$T = t_{ppg} + t_{[4:2]-tree} + \left\lceil \log_2 \frac{N}{4} \right\rceil * t_{[4:2]} + 3t_{add} \quad (11)$$

where N is the order of the filter. The $t_{[4:2]-tree}$ represents the delay consumed by too many levels of [4:2] compressor with a delay of $t_{[4:2]}$.

The symbol ● in Fig. 7 stands for a special adder/subtractor. The proposed architecture of the special adder/subtractor as demonstrated in Fig. 8. The OR tree logic circuit is used to control the quantization threshold (δ). The OR tree block has two levels of circuits in gate level. The NOR2 gate has a delay of 1 logic level, which is same as the NAND3 gate. The OR tree block has a delay of 2 logic levels. In this adder/subtractor, only 3 logic levels more than the CPA. We can get delay of this module is equal to $\{3 + (\log_2 16) \times 2\} = 11$ logic levels—**11L**.

In order to improve the throughput further, we insert delays into the critical path as shown in Fig. 7. From this figure, we can see that there are two adaptation delay. The new weight update equation will be generated by

$$W(n+1) = W(n) + \mu X(n-m) \text{msgn}\{e(n-m)\} \quad (12)$$

Fig. 7. The modification of SFG (filter length $N = 8$ -tap).

where m is the amount of adaptation delays inserted into the CP . When the value of the m is small, the convergence characteristics is almost the same as that of MSLMS algorithm. The proposed architecture involves additional 4 registers to the architecture with zero adaptation delay.

Fig. 8. The structure of special adder/subtractor.

Critical path analysis for MSLMS with 2 adaptation delay in Fig. 7

- *The PM0 Module.* The path of the red line in this module is occupied by only a PPG unit. The delay of PPG unit as mentioned above is 4 logic levels—**4L**.
- *The [4:2] Compressor-tree adder.* There are 4 levels of [4:2] compressor and 1 level of [3:2] adder. So this block has a delay of $6 * 4 + 4 = 28$ logic levels—**28L**.

- *The MOA(4) Module.* This module is a multi-operand-addition with four inputs and two outputs. This module can be implemented by only one level of [4:2] compressor. We assumed that the logic levels for the inverter is one. The delay for this module is 6 logic levels—**6L**.

From the detailed analysis above, we can get that the critical path has a delay of 38 logic levels—**38L**.

4 Results and comparisons

Before giving the comparisons to other structures, we firstly present the most Area-Delay-Power efficient structure in the recent works proposed by P. K. Meher in [1]. Fig. 9 shows the 8-tap architecture with the same assumption above. The critical-path is marked by the red line.

Fig. 9. The structure for 2-adaptation-delay DF-LMS adaptive filter.

Fig. 10. The structure of shifter.

Critical path analysis for [1] with 2 adaptation delay in Fig. 9

- *The Binary-tree adder.* This module has $\lceil \log_2 N/2 \rceil$ levels of CPA. Because the width for CPA is 32, the delay of CPA as mentioned above is 14 logic levels. This module has a delay of $14 \times 2 = 28$ logic levels—**28L**.
- *The Error Calculation Module.* There is a subtractor here to compute the error: $e(n-1) = d(n-1) - y(n-1)$. We will use a 16-bit subtractor based on the truncation technique. Because there is one level of XOR logic more than adder, this block has a delay of $\{2 + (\log_2 16) * 2 + 2\} + 2 = 14$ logic levels—**14L**.
- *The Shifter Module.* This module is a barrel shifter as shown in Fig. 10. From the figure, there are 4 levels of 2-to-1 multiplexer with a delay of 2 logic levels.

Table I. Comparison of hardware and time complexities

Design	Critical-Path	AD (m)	Number of calculator		
			ADD	MUL	REG
K. K. Parhi [2]	$(N + 1)t_{\text{add}} + 3t_{\text{mul}}$	0	$2N + 1$	$2N + 1$	$2N$
Long et al. [9]	$t_{\text{add}} + t_{\text{mul}}$	$1 + \log_2 N$	$2N$	$2N$	$3N + 2\log_2 N + 1$
Y. Yi et al. [6]	t_{mul}	$2 + \log_2 N$	$2N$	$2N$	$6N + \log_2 N + 2$
Meher [1]	$t_{\text{add_tree}} + t_{\text{add}} + t_{\text{shift}}$	2	$2N$	$2N$	$2.5N + 3$
Van&Feng [10]	$t_{\text{add}} + t_{\text{mul}}$	$N/4 + 3$	$2N$	$2N$	$5N + 3$
This work	$t_{\text{dot}} + \log_2(N/4) \times t_{[4:2]}$	2	$N + 2$	0	$9N/4 + 3$

AD: adaptation-delay, ADD: adders, MUL: multipliers, REG: registers.

The delay for this module is $2 \times 4 = 8$ logic levels—**8L**.

From the detailed analysis above, we can reach the conclusion that critical path has a delay of 50 logic levels—**50L**.

Fig. 11. The convergence of adaptive filter for different designs.

Table I shows the hardware and the time complexities of different designs. From this table, we can see that the architecture proposed by Y. Yi et al. [6] has the shortest critical path, but it has the most registers resulted in a higher power consumption. The proposed design uses a fine-grained dot product unit, so there is no multipliers in this design. The number of delays of the proposed design is the second-smallest of all architectures listed in Table I. We can see that the proposed

Table II. Synthesis results using TSMC 65 nm CMOS library

	MSLMS	Meher_2 [1]	Meher_1 [1]	Y. Yi [6]
DAT (ns)	1.29 (75.9%)	1.70 (100%)	1.98	1.55
logic level (L)	38 (76%)	50 (100%)	62	36
Area (μm^2)	31740 (61%)	51649 (100%)	46631	60559
AD (cycles)	2	2	1	6
ADP ($\mu\text{m}^2 \times \text{ns}$)	40945 (47%)	87804 (100%)	92329	93866
Power (mW)	6.86 (69%)	9.93 (100%)	10.43	17.41

DAT: data arrive time, ADP: area-delay product, AD: adaptation-delay.

architecture requires less than half the amount of calculations of addition required by the conventional architectures. The t_{dot} of proposed design is the delay consisting of the PPG unit and [4:2] compressor-tree unit except the last CPA module, so it has less latency than a multiplier.

To prove the correctness of the proposed architecture, we give the results of computer simulation is shown in Fig. 11. In order to make a fair comparison, we use the uniform step-size μ existed in the direct-form (DF) and transpose-form (TF), respectively. It consists of LMS algorithm, DLMS algorithm and proposed MSLMS algorithm. The original signal is the trigonometric sine function. The input signal $x(n)$ is a mixed signal consisting of the original signal and the white Gaussian noise with $SNR = 15$ dB. From this figure we can see that the convergence characteristics of proposed architecture is almost the same as that of the most popular LMS architecture. The best existing architecture proposed by P. K. Meher [1] has the same convergence rate just like the DLMS that is worse than proposed design.

We have coded all the designs in Verilog HDL included in the Table II. To validate the advantages of our proposal, we analyzed the design of P. K. Meher in terms of logic levels in detail as mentioned above. The critical path of proposed architecture has a delay of 38 logic levels (38L) which is 34% lower than that of P. K. Meher (38 VS 50), which is almost identical with DC synthesis results for the DAT. Due to the same time constraints of all designs in DC synthesis, DC will give an automatic optimization for them. The critical path of the design presented by Y. Yi [6] is only a multiplier, DC will give a relatively large optimization which led to the mismatch between the logic levels and the DAT. Compared to the architecture proposed by P. K. Meher, the proposed design saves roughly 39% of area (31740 VS 51649) and 31% of power (6.86×2 VS 9.93) and estimated at DAT. It is discovered that the proposed architecture involves about 53% less ADP than the best existing work by P. K. Meher for the 8-tap filter.

5 Conclusion

This paper proposed a delay-optimized architecture using fine-grained dot product unit for the modified SLMS algorithm. The proposed architecture has a superduper convergence rate compared to other existing architectures. A detailed analysis of latency of the critical path shows that the proposed design has a delay of 38 logic levels compared to 50 logic levels in the design of P. K. Meher. Using the fine-grained dot product unit and multi-operand-addition, MSLMS architecture reduces the DAT by 24.1% compared to that of P. K. Meher. Additionally, the proposed design involves 53% less ADP and 31% power than that of the best existing structure. The proposed design can achieve 2-adaptation-delay, small hardware and lower power consumption, simultaneously.

Acknowledgments

This work is supported and funded by the project of Shenzhen Infrastructure layout. Project No.: JCYJ20150827165024088.