Automatic Design for Bit-Serial MSPA Architecture

Hiroaki Kunieda Yusong Liao Dongju Li Kazuhito Ito

Dept. of EE Eng.
Tokyo Institute of Technology
Ookayama, Meguro-ku, Tokyo 152
Tel: +81-3-5734-2574
Fax: +81-3-5734-2911
e-mail: kunieda@ss.titech.ac.jp

Dept. of EE Syst. Eng.
Saitama University
Shimookubo, Urawa, Saitama 338
Tel: +81-48-858-
Fax: +81-48-855-0940
e-mail: kazuhito@elc.ees.saitama-u.ac.jp

Abstract— Memory Sharing Processor Array (MSPA) architecture is effective in both data storage and processor cell utilization efficiency. In this paper, the design methodology for MSPA is extended to synthesize bit-serial datapath. As a synthesis example, we propose a new bit-serial multiplier with smaller number of logic gates than conventional bit-serial multipliers.

I. Introduction

Systolic array is suitable for VLSI implementation. It can realize the multiple data flow through the processor cell array by the local communication. However, it sometimes suffers from the poor processor cell utilization efficiency at the expense of the local communication. Inheriting the advantages of both systolic arrays and common bus architectures, the Memory Shared Processor Array (MSPA) for the regular algorithms has been proposed[1]. The MSPA architecture consists of the processor cell array with several memory units and their address generation hardware units in order to minimize the data storage[2][3].

In this paper, we extend the MSPA technology to an application of a bit-level system organization, and propose the bit-serial data path synthesis methodology for a regular algorithm.

The proposed architecture consists of bit-level systolic (or semi-systolic) processor cell array with local communications and the data format converters used efficiently to convey the bit-serial data from input to the processor cell array. Our approach can achieve the optimal trade-off of the hardware complexity between the datapath and the control path. If we select the same number of processor cells as the number of operations, the conventional systolic array with the simplest data control scheme may be obtained. However, if we select the less number of processor cells than the number of operations, the novel MSPA architecture will be derived with the reasonable balance between the datapath and control hardware.

The design methodology of the execution ordering and the resource allocation for the bit-level MSPA are similar to the processor cell level MSPA array, which are based on modified linear transformations with keeping the regularity of the algorithm and increasing the resource utilization rate. Next to the scheduling, we design the data format converters. They efficiently convey the input data to the processor cell array.

By demonstrating the design of a two's complement multiplier with the proposed methodology, we make clear that the proposed methodology achieves higher parallel efficiency than the conventional methodology do.

II. Bit-Serial MSPA Architecture

In this paper, we discuss how to schedule bit-level regular algorithms on processor cell array and how to generate the data format converters which convey bit serial input data to the linear array. We consider the bit-level algorithms expressed in two dimensional index space with two bit serial input data. The LSB of bit serial input data come in at first and the MSB at last.

Let's consider algorithms described by the uniform recurrence relations as

$$\begin{array}{l} for\ i = 0\ to\ n-1\ step\ 1 \\ for\ j = 0\ to\ n-1\ step\ 1 \end{array}$$

$$v_1(p) = f_1(x(0,j), y(i,0), v_1(p-d_{11}), v_2(p-d_{21}))$$

$$v_2(p) = f_2(x(0,j), y(i,0), v_1(p-d_{11}), v_2(p-d_{21})).$$

where $v_k(p)(p=(i,j))$ denotes the value at the index point (i,j) of the k-th variable. d_{kl} are column vectors in the same index space to show the data dependency of the variable $v_k(i,j)$ against the variable $v_l(i,j)$. The function $f_k()$ represents the operation to produce the variable $v_k(i,j)$. The input variables are devoted by $x(0,j)$ and $y(i,0)$ which express the j-th or i-th bit of the input bit serial data.

In this paper, we restrict the data dependence relations between the variables and the input data such that they are described in the index space (i,j). Those dependence vectors are written as

$$d_{x_i v} = (i, 0)^T \quad d_{y_j v} = (0, j)^T. \tag{1}$$

The other dependence vectors such as those between variables are determined according to the algorithms.

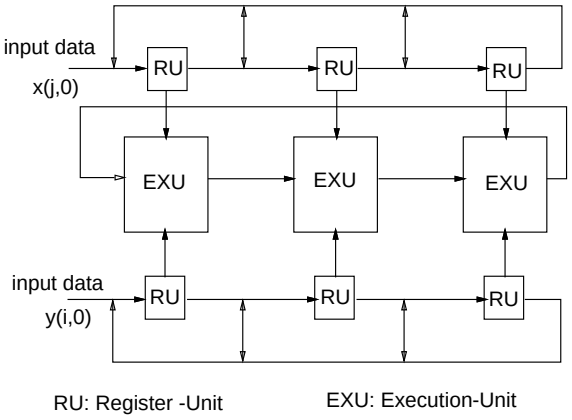

Fig. 1. Bit-Serial MSPA

The proposed bit-serial processing architecture is shown in Fig.1. It consists of two input data format converters and the linear processor cell array. The processor cell array performs the executions to calculate the variables with the local communications between the processor cells. This scheme is similar to the systolic array. The two input bit-serial data are inserted into data format converters, which convert the data format from a bit-serial form to multiple data streams. This scheme can also be seen as the data broadcasting. The data format converters are constructed by plural register units each of which consists of two stages registers.

III. Design Procedure

A. Execution Ordering

We consider the time coordinate by the linear transformation with a row vector T for each operation set f_k. For an algorithm description of

$$v_k(p) = f_k(x(0, j), y(i, 0), v_1(p - d_{1k}), v_2(p - d_{2k})), \tag{2}$$

the operation $f_k()$ invokes at

$$Tp = t_1 i + t_2 j, \tag{3}$$

and produces the variable $v_k(p)$. The mapping with T can be selected to satisfy the valid execution ordering for any data dependences. For a data dependence d_{kl},

$$Td_{kl} > 0. \tag{4}$$

must hold. This condition indicates that the variable $v_l(p)$ is generated by using the variable $v_k(p - d_{kl})$ whose generation time is earlier than that of $v_l(p)$.

Among the mappings with the valid execution ordering, we select the optimal mapping which requires the minimum total execution time. We derived the optimal vector for two dimensional cases, but this will be reported in the other paper. Here, we describe only the result without the proof in the form of the theorem.

Theorem 1 *The optimal mapping vector on exu processing elements which achieves the minimum total execution time for the uniform recurrence algorithm with the two n iteration loops of exu < n is given by*

$$T = (1, \lfloor \frac{n}{exu} \rfloor), \tag{5}$$

where $\lfloor x \rfloor$ represents the smallest integer more than or equal to x.

This mapping vector is valid for the algorithm consisting of the depedence vectors of the range specified by eq.(4).

B. Resource Allocation

The position of each processing cell in the array is described by 1 dimensional coordinate. We introduce the linear mapping with modular operations against the given algorithm.

$$Op \bmod exu = (o_1 i_1 + o_2 i_2) \ (\mathrm{mod} exu) \tag{6}$$

This method increases the resource utilization rate while preserving the regularity of the algorithm. In fact, the data transfer schemes derived by this mapping remain in regular fashion. Of course, the conventional linear mapping for the systolic array is included as special cases. The mapping with O can be selected such that any two operations of the same execution times are not mapped onto the same operation cell. (Conflict free allocation condition) For any two index points p, p' $(p \neq p')$ which invoke at the same time $(Tp = Tp')$, the mapping with the O satisfies the inequivalence of

$$Op \not\equiv Op' \pmod{exu}. \tag{7}$$

Furethermore, the mapping vector O should realize the interprocessor communication correspondings to the dependence vectors by the successive one among adjacent processors. (Local communication condition)

Theorem 2 *The optimal mapping vector on exu linear PE array which achieves both the conflict free allocation and the local communication for the uniform recurrence algorithm with the two n iteration loops of exu < n is given by*

$$O = (0, 1) \ (\mathrm{mod} exu). \tag{8}$$

This vector is proved to satisfy the conflict free allocation condition. Furthermore, the range of the dependence vectors satisfying the local communication condition with this vector is described as

$$|Od_{kl} \ (\mathrm{mod} exu)| \leq Td_{kl}. \quad (9)$$

Fig.2 shows the case for $t_2 = 2$. The dependence vectors, which are originated from the large black circle node to any black circles, satisfy eq.(9).

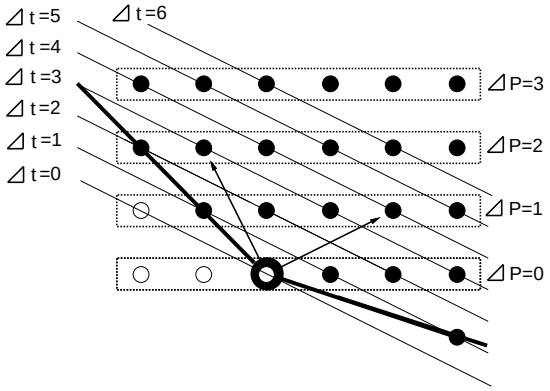

Fig. 2. Valid Range for Dependence Vectors

Thus, the bit-serial MSPA architecture will be realized by the loosely coupled array structure in which the intermediate variable data circulate in the systolic fashion and by the memory units or register data format converter, in which input and output data distribute or collect data to or from processor elements in the shared memory architecture fashion.

IV. Data Format Converter

Both execution ordering and resource allocation determine the data communication schemes which include the input data loading and the distribution of the input data and the variable data. We consider here the systematic data loading and distribution of the bit serial input data by employing the data format converters. The similar techniques can be applied to the distribution of the variable data if those are not circulated within the processor cell array.

The systematic synthesis methodology of the DSP data format converter architectures have been proposed with the minimum number of registers[9]. We are able to make use of the method to derive our data format converters. However, the register organization designed by the method requires the complicate control mechanism, so that the chip area of the converter become so large as to degrade the advantage of the minimum number of registers.

We develop a new method to generate data format converters with the smaller chip area. We employ the register units corresponding to each processor cell. By limiting the number of registers used in each register unit, we try to achieve the minimum number of registers. At the same time, we simplify the control mechanism of registers in each register unit so as to achieve the cheap hardware cost.

The positions of the register units are described in the space coordinate as $0, 1, 2, \cdots, exu - 1$, which correspond to the position of the corresponding processor cell. We assume that the input bit-serial data will be input to the register unit at the position 0. The i-th lower bit is assumed to be input at the time i.

A. Data Load

The time duration $\triangle t_{load}(x_j)$ of the variable $x(0, j)$ and $y(i, 0)$ between input and load time to the register units are defined respectively as

$$\begin{aligned} \triangle t_{load}(x_j) &= (t_1 0 + t_2 j) - j = (t_2 - 1)j \\ \triangle t_{load}(y_i) &= (1i + t_2 0) - i = 0. \end{aligned}$$

The load positions $P_{load}(x_j)$ and $P_{load}(y_i)$ of the variable $x(0, j)$ and $y(i, 0)$ are written respectively as

$$\begin{aligned} P_{load}(x_j) &= o_1 0 + o_2 j = 0 \times 0 + 1j = j \ (\mathrm{mod} exu) \\ P_{load}(y_i) &= o_1 i + o_2 0 = 0i + 1 \times 0 = 0 \end{aligned}$$

The j-th bit of the input variable $x(0, j)$ is loaded to the $(j \bmod exu)$-th register unit at $(t_2 - 1)j$ cycles after the bit-serial input cycle. This scheme can be realized by the cascade connection of the first stage registers in every $(j \bmod exu)$ register units, which consists of $(t_2 - 1)$ registers.

The i-th bit of the input variable $y(i, 0)$ is loaded to the 0-th register unit at the same cycle as the bit-serial input cycle. This scheme can be realized by the direct connection between the input and 0-th register.

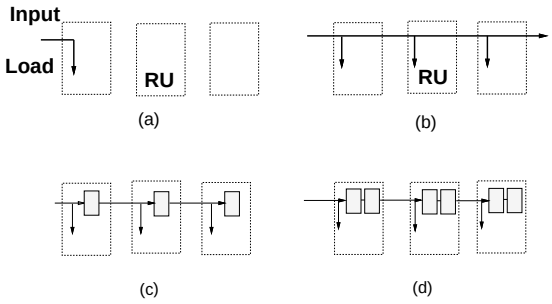

Fig. 3. The First Stage Register (a)$\triangle t_{load} = 0, P_{load} = 0$, (b)$\triangle t_{load} = 0, P_{load} = j$, (c)$\triangle t_{load} = j, P_{load} = j$, (d)$\triangle t_{load} = 2j, P_{load} = j$

Fig.3(a) shows the loading schemes for $y(i, 0)$. While, Fig.3(b)(c)(d) shows the three kinds of the loading schemes for $x(0, j)$ in case of $t_2 = 1, 2$, and 3 respectively.

B. Data Circulation

After the first stage register allocation, we decide the method to distribute the input variables through the register units so that they can be reused in the processor cell array. This scheme is achieved by employing the second stage registers in each register unit.

The loaded data $x(0,j)$ and $y(i,0)$ are transferred to the register units of

$$\begin{aligned}\triangle P_{circ}(x_j) &= (o_1 1 + o_2 j) - (o_1 0 + o_2 j) = o_1 = 0\\ \triangle P_{circ}(y_i) &= (o_1 i + o_2 1) - (o_1 i + o_2 0) = o_2 = 1\end{aligned}$$

position rightward from the current register unit position in every time unit of

$$\begin{aligned}\triangle t_{circ}(x_j) &= (t_1 1 + t_2 j) - (t_1 0 + t_2 j) = t_1 = 1\\ \triangle t_{circ}(y_i) &= (t_1 i + t_2 1) - (t_1 i + t_2 0) = t_2.\end{aligned}$$

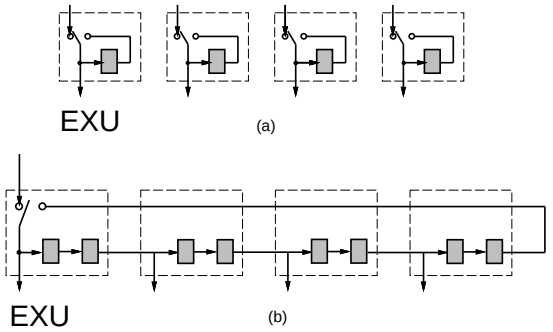

Fig. 4. The Second Stage Register (a)$\triangle t_{circ} = 1, \triangle P_{circ} = 0$, (b)$\triangle t_{circ} = 2, \triangle P_{circ} = 1$

Fig.4(a) shows the circulation schemes for $x(0,j)$. While, Fig.4(b)) shows the circulation schemes for $y(i,0)$.

V. MSPA Bit-Serial Multiplier

We present here an practical example of our excellent bit-serial datapath synthesis using the novel MSPA technology. The result will make clear that the proposed methodology achieves higher parallel efficiency than the conventional systolic method.

There have been proposed various bit-serial multipliers, most of which are realized with the systolic architecture. The number of 8-bits serial multiplier requires 8 full adders in the conventional systolic realization. Though our approach includes the systolic case, we show the design case with 4 full adders as processor cells.

The algorithm description for a bit-serial multiplier $[x_j][y_i]$ for 8-bits serial multiplier is described as

for $i = 0$ *to* 7 *step* 1
for $j = 0$ *to* 7 *step* 1

$$s(i,j) = f_1(x(0,j), y(i,0), s(i+1,j-1), c(i-1,j))$$

$$c(i,j) = f_2(x(0,j), y(i,0), s(i+1,j-1), c(i-1,j)),$$

where $s(i+1,-1) = 0$, $c(-1,j) = 0$. The function f_1 corresponds to the partial sum of one bit full adder, while the function f_2 generates the carry of one bit full adder. From those expression, the dependence vectors of this algorithm are written as

$$d_{xs} = d_{xc} = (i,0)^T \quad d_{ys} = d_{yc} = (0,j)^T \tag{10}$$
$$d_{ss} = d_{sc} = (-1,1)^T \quad d_{cs} = d_{cc} = (1,0)^T. \tag{11}$$

A. Design Procedure

We can obtain the optimal schedule and resource allocation vectors as

$$T = (1,2) \quad O = (0,1), \quad exu = 4. \tag{12}$$

Fig.5 shows the execution ordering and the resource allocation. This schedule achieves the total execution within 22 steps. Four full adder executions are used in this multiplier. The operation ordering and the resource allocation determine the communication schemes of the variables. The communications corresponding to the dependence vectors d_{ss} and d_{sc} are expressed as $Td_{ss} = 1$, $Od_{ss} = 1$ and $Td_{sc} = 1$, $Od_{sc} = 1$. This scheme indicates that the partial sum $s(i,j)$ should be sent to the rightward processor cell in every time units.

While, the communications corresponding to the dependence vectors d_{cs} and d_{cc} are expressed as $Td_{cs} = 1$, $Od_{cs} = 0$ and $Td_{cc} = 1$, $Od_{cc} = 0$. This scheme indicates that the carry $c(i,j)$ should be sent to the same processor cell in every time units.

Both variables of the partial sums and the carrys are held in the processor cell array so that the advantages of the multiple data stream and local communication for array processing will be enjoyed.

The data format converters to convert the bit-serial input data $x(0,j)$ and $y(i,0)$ into the multiple data stream in the processor array are derived as shown in Fig.6 and Fig.7, respectively.

B. Final Circuit

The practical multiplier needs to perform signed multiplications, though we develop the multiplier without sign consideration. This is because we avoid the exceptional irregularity of the algorithm in order to enjoy the highly regular structure of the multiplication algorithm. However, it turns out that this structure can be kept intact by using the Hatamian algorithm for two's complement multiplication[5]. The sign modification with the algorithm

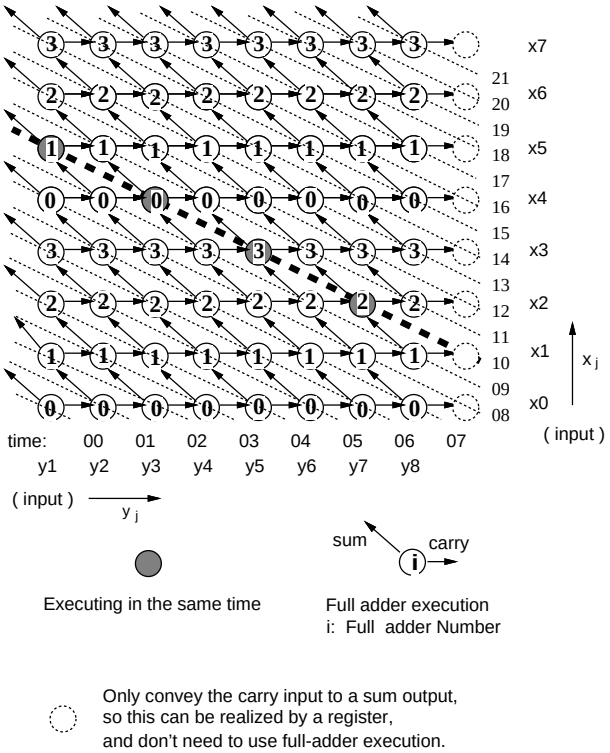

Fig. 5. Execution Ordering and Resouce Allocation of the Multiplier

leads to the original structure with the conditionally inverted terms and additions of two 1's. It is realized by slightly modifying the control circuit.

Fig.8 shows the final design of 8-bits two's complement multiplier. Designed multiplier consists of two input data format converters, as shown in Fig.6 and Fig.7. Since the lif-time analysis of the variable data $x(0, j)$, we obtain the largest lif-time overlapping number of 7. We can reduce the number of registers to 7 in the data converter for $x(0, j)$ as shown in Fig.8. The linear processor cell array consists of four full adder executions. We use the AND and EXOR gates to produce the partial product terms. The EXOR gates are used to realize the inverted terms

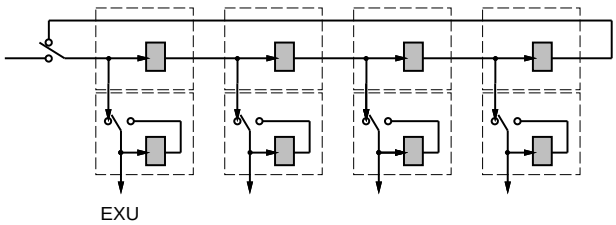

Fig. 6. The x_j's Data Format Converter

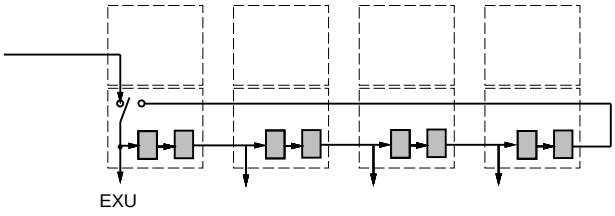

Fig. 7. The y_i's Data Format Converter

for the sign modification.

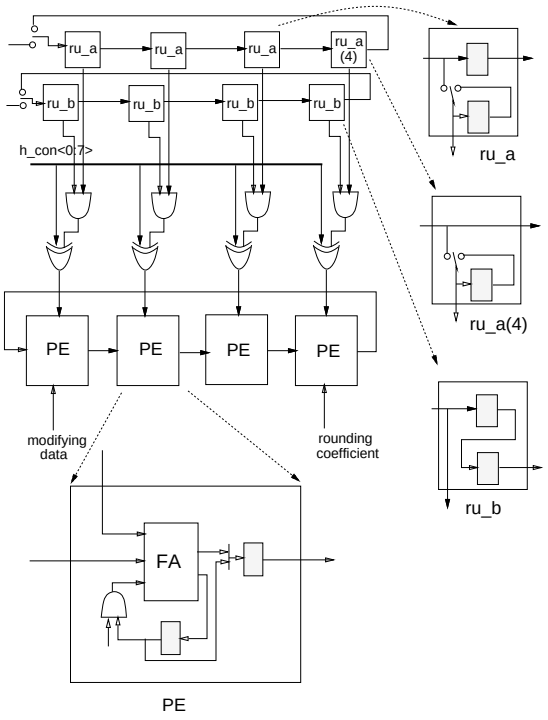

Fig. 8. Final design of the two's complement multiplier

C. Comparison

Comparisons between the $n \times n$-bits multipliers of Lyon [6], Smith [7], Bi [8] and the proposed one are summarized in Tab.I in terms of efficiency, performance and structural regularity.

Chip area is estimated in terms of the number of dual input AND gates under the assumption that each gate occupies a unit area. For example, a delay element and a full adder occupy a chip area equivalent to 6 and 8 unit areas, respectively. Because the total chip area is determined by the chip area of all the stages, we see that the proposed design requires considerably less chip area

	Lyon[6]	*Smith*[7]	*Bi*[8]	*Proposed*
chip area per stage	64	36	160	62
No. of stages	n	n	n	$\frac{n}{2}$
No. of cycles per product	n	$2n$	n	$2n$
Latency (cycles)	$2n+1$	2	$2n+4$	$2n$
Parallel efficiency	0.348	0.5	0.348	0.696
Computation time	46	32	46	46
Real time	*yes*	*no*	*yes*	*yes*
Structure regularity	*systolic*	$semi-systolic$	*systolic*	$semi-systolic$

TABLE I
PERFORMANCE COMPARISONS

than the other three designs do.

Parallel efficiency means the processor cell utilization efficiency of one multiply execution. It is clear that the proposed design is effective in processor cell utilization efficiency.

Overall, it's concluded that the new design based on the proposed architecture is shown to be attractive in term of the following features: real-time operation, flexible processing precision and structural regularity, small chip area and high processor cell utilization efficiency.

VI. CONCLUSION

We present the novel architectures of bit-serial datapath synthesis based on Memory Shared Processor Array technique for the regular bit-level algorithms. The design procedure to derive the MSPA bit-level architecture requires the design of the data format converters. We succeeded to derive the systematic design to generate data format converters with the minimum number of registers and the simpler control scheme.

As a synthesis example, we present a new bit-serial multiplier with the smaller number of logic gates than conventional bit-serial multipliers. The typical feature of our multiplier is to select the number of the processor cells. In our bit-serial multiplier, we selected full adders of half the number of bits. We can select any number of full adders with our design methodology. If we adopt the same number of full adders as the number of data bits, we can derive the conventional systolic multiplier. If we decrease the number of full adders, the multiplier with more complex control scheme will be derived. In other words, the ratio of the hardware complexity between the datapath and the control circuits can be optimized by the selection of the number of processor cells.

ACKNOWLEDGEMENTS

This work has been engaged as a project in CAD21 Research Body of Tokyo Institute of Technology. We wish to thank all the members of CAD21 for their suggestions and cooperations.

REFERENCES

[1] H. Kunieda and K. Hagiwara. "Effective processor array architecture with shared memory". In *IEEE, APCCAS'94*, pp.113-118, 1994

[2] D.M. Grant and P.B. Denyer. "Address generation for array access based on modulus m counters". In *EDAC. Proceedings of the European Conference on Design Automation*, pp.118–122, Feb. 1991.

[3] P.E.R. Lippens, J.L. van Meerbergen, A. van der Werf, W.F.J. Verhaegh, and B.T. McSweeney. "Memory synthesis for high speed DSP applications". In *Proceedings of the IEEE 1991 Custom Integrated Circuits Conference*, pp.11.7/1–4, May 1991.

[4] M. Hatanmian and G. l. Cash. "A 70-MHz 8-bit×8-bit Parallel Piplelined Multiplier in 2.5-μm CMOS". In *IEEE Journal of Solid-State Circuits*, V0l. SC-21, No. 4, pp.505-513, August 1986

[5] LYON,R.E. "Two's complement pipleline multipliers ". In *IEEE Transactions On Communications*, COM-24, pp.418-425, April 1976

[6] SMITH, S.G. "Serial/parallel automultiplier". In *Electron. Letters*, Vol.23, No.8, pp.413-415, 1987

[7] B. Bi and E. B. Jones. "High-performance bit-serial adders and multipliers". In *IEEE Proc. Circuits, Devices and Systems*, Vol.139, No.1, pp.109-113, Feb, 1992

[8] K. K. Parhi. "Systematic systhesis of DSP data format converters using life-time analysis and forward-backward register allocation". In *IEEE Transactions On Circuits And Systems-II:Analog And Digital Signal Processing*, Vol.39, No.7, pp.423-440, 1992