論文 88-25-9-13

A VLSI Architecture of Systolic Array for FFT Computation

(고속 퓨리어 변환 연산용 VLSI 시스토릭 어레이 아키텍춰)

辛 卿 旭*, 崔 炳 允*, 李 文 基*

(Kyung Wook Shin, Byeong Yoon Choi and Moon Key Lee)

要 約

규칙적이고 반복적인 VLSI 구조를 갖는 fast Fourier transform(FFT) 연산용 2차원 시스토릭 어레이(systolic array) 아키텍춰를 소개한다.

제안된 시스토릭 어레이는 데이타 배열회로, 반쪽 버터플라이 연산회로, 제어회로 등으로 구성된 동일한 처리요소(processing element)들의 2차원 배열이며, 임의의 갯수까지 확장이 가능하다.

전체적인 FFT 연산과정은 데이타 입출력 파이프라인, 데이타 셔플링 그리고 버터플라이 연산 등으로 크게 나눌 수 있다. 처리요소 간의 데이타 이동은 병렬성, 파이프라인, 국부통신 등의 특성을 가지며, 따라서 효율적인 셔플링이 이루어진다. 본 논문에서는 기존의 버터플라이 연산대신 반쪽 버터플라이 연산을 제안하였으며, 이는 연산속도의 향상, 연산회로의 단순화, 데이타 셔플링 시간의 단축 등의 장점이 있으며, 또한 처리요소의 연산 참여율(PE Activity)이 기존 방식의 50%에서 100%로 향상 되었다. 반쪽 버터플라이 연산 칩을 3μm, CMOS 공정으로 제작하였으며, 20MHz 클락을 사용하였을 때 반쪽버터플라이 연산시간은 500nS이며, 1024개의 표본점에 대한 전체 FFT연산시간은 데이타의 입출력 시간을 포함하며 16.6μS이다.

현재, 한 개의 처리요소로 이루어진 칩을 2μm, 이중금속, CMOS 공정으로 웨이퍼 가공중에 있다.

Abstract

A two-dimensional systolic array for fast Fourier transform, which has a regular and recursive VLSI architecture is presented. The array is constructed with identical processing elements (PE) in mesh type, and due to its modularity, it can be expanded to an arbitrary size. A processing element consists of two data routing units, a butterfly arithmetic unit and a simple control unit.

The array computes FFT through three procedures; I/O pipelining, data shuffling and butterfly arithmetic. By utilizing parallelism, pipelining and local communication geometry during data movement, the two-dimensional systolic array eliminates global and irregular commutation problems, which have been a limiting factor in VLSI implementation of FFT processor. The systolic array executes a half butterfly arithmetic based on a distributed arithmetic that can carry out multiplication with only adders. Also, the systolic array provides 100% PE activity, i.e., none of the PEs are idle at any time.

A chip for half butterfly arithmetic, which consists of two BLC adders and registers, has been fabricated using a 3-um single metal P-well CMOS technology. With the half butterfly arithmetic execution time of about 500 ns which has been obtained by critical path delay simulation, total FFT execution time for 1024 points is estimated about 16.6 us at clock frequency of 20MHz. A one-PE chip expansible to any size of array is being fabricated using a 2-um, double metal, P-well CMOS process. The chip was layouted using standard cell library and macrocell of BLC adder with the aid of auto-routing software. It consists of around 6000 transistors and 68 I/O pads on 3.4 x 2.8mm^2 area. A built-in self-testing cirucit, BILBO (Built-In Logic Block Observation), was employed at the expense of 3% hardware overhead.

*正會員, 延世大學校 電子工學科

(Dept. of Elec. Eng., Yonsei Univ.)

接受日字 : 1988年 4月 18日

(※ The work described here is a result of MPC(Multi Project Chip) development project supported by the Ministry of Science and Technology of KOREA.)

I. Introduction

The VLSI implementations of FFT processors lead to long interconnection wires because of an inherent requirement for global communications in the FFT tree diagram, that consume large area and power. A concept of good algorithm is the most crucial factor in the design of complex VLSI chips, because the underlying algorithm determines the degree of concurrency and pipelining which eventually determines the performance of the chip [1]. Furthermore, algorithms with regular and local communication would have smaller chip area in comparison to algorithms with irregular, global communications. Besides, the amount of time spent not only on designing and debugging the chip, but also expanding the design, if required, can be reduced substantially. Many VLSI implementation approaches for FFT processors are available according to the desired degrees of parallelism and pipelining [2-9]. To circumvent the global communication problem, a systolic elevator concept was proposed by [10], but it is unfavorable because of a long permutation time. Another approach proposed by [11] is to permutate data on a two dimensional array. But, it's a major limitation that it requires both data ordering and reordering operations between FFT processos, and only a half of the processing elements (PE) is active on a butterfly computation, i.e., it provides only 50% of PE activity.

This paper presents a VLSI architecture for FFT computation that has the following properties,

. regular and recursive hardware structure, i.e., array structure
. local communication geometry during data movement
. parallel, pipelined data shuffling and butterfly arithmetic
. expansible to any size FFT with simple change
. simple control flow
. none of the PEs in the array are idel at any time, i.e., 100% PE activity

By mapping the one-dimensional data stream onto the two dimensional systolic array, inter-process data shuffling operations can be carried out not only in a simple and systematic pattern, but also in a parallel, pipelined fashion. Also, by adopting half butterfly arithmetics (HBA) [12,13], data reshuffling operations that were indispensible in [5,7,11], is not needed for the proposed architecture. This HBA concept makes all of the PEs are active at all time, i.e., 100% the PE activity.

II. Fast Fourier Transform on a 2-Dimensional Systolic Array

N points, radix-2, decimation in time FFT is defined as, [14]

$$F_q(K) = F_{q-1}(K) + F_{q-1}(K+N/2^q)\cdot W^p \quad (1\text{ a})$$

$$F_q(K+N/2^q) = F_{q-1}(K) - F_{q-1}(K+N/2^q)\cdot W^p \quad (1\text{ b})$$

$$q = 0, 1, \ldots\ldots. \log_2 N$$

$$K = 0, \ldots\ldots\ldots N/2^q - 1$$

where $W^p = \exp(-j2\pi p/N)$ is the Twiddle factor, p is determined from the time index, q is the process number, and $F_0(K)$ denotes initial input data.

Instead of computing the conventional butterfly arithmetic of eq.(1) simultaneously in a PE, by executing each of eq. (1-a) and eq. (1-b) separately on different PEs, data reshuffling operations are not needed for the proposed architecture. This makes also the PE activity to be increased to 100% which is two fold improvement compared with conventional butterfly arithmetic.

The FFT computation on the two-dimensional systolic array is carried out through three procedures; I/O pipeline, data shuffling and HBA operation. After initial input data are loaded into the array, data shuffling and HBA operations are carried out in a parallel and pipelined manner. During inter-process data shuffling operation, data in each PE are moved to their destination PE with nearest-neighbor communication geometry. After the data shuffling operation has been completed, each PE computes HBA independently. Immediately after the whole FFT computation through entire processes has been completed, I/O pipeline procedure is performed, i.e., the output of the result and the input of new data are pipelined.

1. Data Shuffling

The FFT tree diagram, which is shown in Fig. (1)

for N=16, can be derived directly from eq.(1). As we can see in eq.(1) and Fig.(1), the data pair F_{q-1} (K) and F_{q-1} (K+N/2^q) should be shuffled between process q-1 and process q with their shuffling distance of N/2^q.

To do this data shuffling operation efficiently, the one dimensional data stream $\{F_q(k)\}$ are mapped onto a two-dimensional systolic array in row major order, which consists of 2^m x 2^n PEs forN = 2^{m+n} points FFT as shown in Fig.(2). The mapping relationship between data {Fq (k)} and processing element P (r,c) can be expressed as

$$P(r,c) \leftarrow F_q(k) \qquad (2)$$

$$k=2^n(r-1)+(c-1)$$

where, r= 1,2,3,.... 2^m
c= 1,2,3,.... 2^n

where, r and c are the row and column indices of an array of PEs, respectively. For N=16, the systolic array is composed of 4x4 PEs, and its data mapping is depicted in Fig.(3).

To perform the data shuffling operation on the systolic array, data in each PE move to their destination PE with nearest-neighbor communication geometry. Let's take the case where N=16 as an example. At time T=1 of 1st process, data in the 1st and 2nd row of the array move to the 2nd and 3rd row, respectively. At the same time, data in the 3rd and 4th row move to the 2nd and 3rd row, respectively. At T=2, data in the 2nd and 3rd row, which were moved from the 1st and 2nd row at T=1, are loaded into PEs in the 3rd and 4th row. On the other hand, data in the 2nd and 3rd row, which were moved from the 3rd and 4th row at T=1, are loaded into PEs in the 1st and 2nd row, respectively. As a result of the data shuffling operation for 1st process,data in the 1st and 2nd row are exchanged with those in the 3rd and 4th row, respectively. This data movement is completed in a parallel and pipelined fashion within only two cycles. Similarily, in 2nd process, the PEs in the 1st row exchange their data with PEs in the 2nd row, and the 3rd row with the 4th row, respectively. Also, in 3rd process, the PEs in the 1st and 2nd column exchange their data with PEs in the 3rd and 4th column, respectively. Also in 4th process, data movement is performed in a similar pattern. These data

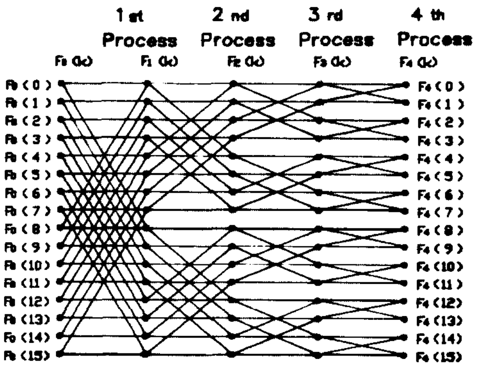

1st Process
2nd Process
3rd Process
4th Process
F0(k)
F1(k)
F2(k)
F3(k)
F4(k)
F0(0) F0(1) F0(2) F0(3) F0(4) F0(5) F0(6) F0(7) F0(8) F0(9) F0(10) F0(11) F0(12) F0(13) F0(14) F0(15)
F4(0) F4(1) F4(2) F4(3) F4(4) F4(5) F4(6) F4(7) F4(8) F4(9) F4(10) F4(11) F4(12) F4(13) F4(14) F4(15)

Fig.1. FFT tree diagram for N=16.

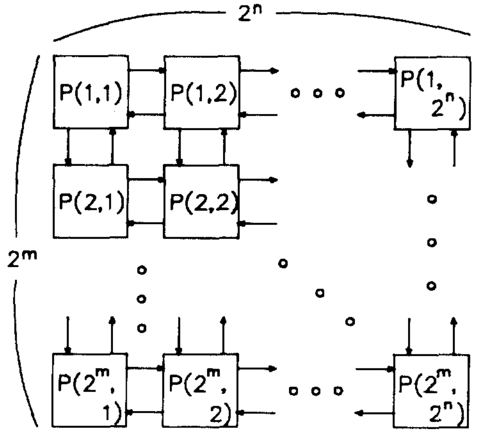


Fig.2. Systolic array for N=2^{m+n} points FFT.

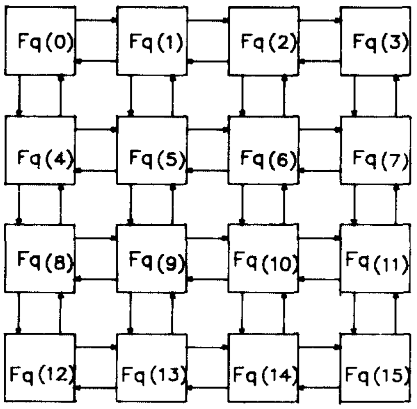


Fig.3. data mapping for N=16.

shuffling operations are shown in Fig. (4). As we can see from this example, the data movement is very simple and regular, thus it makes the control very simple.

Fig.(5) shows the data shuffling procedure for an arbitrary size, $N = 2^{m+n}$ From 1st process to mth process, row shuffling is carried out, and from (m+1)th process to (m+n)th process, column shuffling is performed.

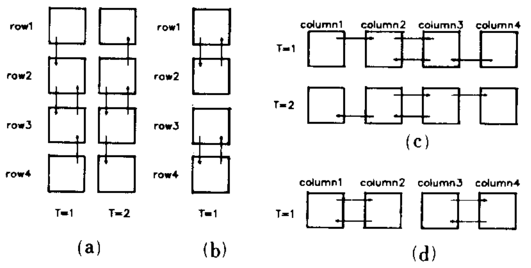


Fig.4. Data shuffling operation for N=16.
(a) 1st process (q=1).
(b) 2nd process (q=2).
(c) 3rd process (q=3).
(d) 4th process (q=4).

```
FOR $q \leftarrow 1$ TO m DO
  FOR $r \leftarrow 1$ TO $2^m$ DO IN PARALLEL
    IF ( ( $(r-1)$ MOD $2^{m-q+1}$ ) $< 2^{m-q}$ ) THEN
      $r' = r + 2^{m-q}$
    ELSE
      $r' = r - 2^{m-q}$
    ENDIF
    FOR $c \leftarrow 1$ TO $2^n$ DO IN PARALLEL
      $k = 2^n * (r-1) + (c-1)$
      MOVE DATA $F_{q-1}(k)$ TO $P(r',c)$
    ENDFOR
  ENDFOR
ENDFOR

FOR $q \leftarrow m+1$ TO $m+n$ DO
  FOR $c \leftarrow 1$ TO $2^n$ DO IN PARALLEL
    IF ( ( $(c-1)$ MOD $2^{m+n-q+1}$ ) $< 2^{m+n-q}$ ) THEN
      $c' = c + 2^{m+n-q}$
    ELSE
      $c' = c - 2^{m+n-q}$
    ENDIF
    FOR $r \leftarrow 1$ TO $2^m$ DO IN PARALLEL
      $k = 2^n * (r-1) + (c-1)$
      MOVE DATA $F_{q-1}(k)$ TO $P(r,c')$
    ENDFOR
  ENDFOR
ENDFOR
```

Fig.5. Data shuffling procedure for $N=2^{m+n}$

2. *Half Butterfly Arithmetic (HBA)*

For simplicity, eq.(1) can be written symbolically using an operator &,

$$HBA = A \,\&\, B \cdot W \quad (3)$$

where, the operator & represents either '+' (Addition) or '-' (subtraction). Eq.(3) defines two HBA operations, i.e., HBA+ if the operator & is '+', and HBA- if the operator & is '-'. Fig.(6) shows the HBA operation and conventional butterfly arithmetic. The HBA provides simpler hardware and faster execution than conventional butterfly arithmetic operation.

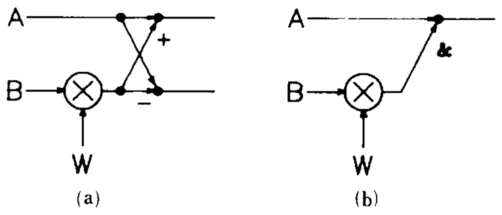


Fig.6. Comparison for two butterfly arithmeics.
(a) conventional butterfly.
(b) half butterfly.

The HBA procedure for 2^{m+n} points transform is shown in Fig.(7). The HBA type is determined by both process q and the row or column index of PE. The PEs originally having F_{q-1} (K) execute HBA+ to obtain Fq (K), and the PEs originally having F_{q-1} $(K+N/2^q)$ execute HBA- to obtain F_q $(K+N/2^q)$. As an example of the HBA, let's take the case where N=16 as before. In 1st process, the PEs in the 1st and 2nd row execute HBA+, and the PEs in the 3rd and 4th row execute HBA-. Similrily, in 2nd process, the PEs in the 1st and 3rd row execute HBA+, and the PEs in the 2nd and 4th row execute HBA-. Fig.(8) depicts these HBA operations for each process. Therefore, the half of PEs in the array executes HBA+, while the other half exeuctes HBA- in parallel. These HBA operations eliminate the data reshuffling operation, which would be indispensible, if conventional butterfly arithmetic was adopted on the two dimensional array. Also it makes the PE activity to be improved to 100% in comparison with conventional butterfly arithmetic.

```
FOR q ← 1 TO m DO
  DO IN PARALLEL
    IF ( ( (r−1) MOD 2^{m−q+1} ) < 2^{m−q} ) THEN
      EACH PE COMPUTE HBA+
    ELSE
      EACH PE COMPUTE HBA−
    ENDIF
  ENDDO
ENDFOR

FOR q ← 1 m+1 TO m+n DO
  DO IN PARALLEL
    IF ( ( (c−1) MOD 2^{m+n−q+1} ) < 2^{m+n−q} ) THEN
      EACH PE COMPUTE HBA+
    ELSE
      EACH PE COMPUTE HBA−
    ENDIF
  ENDDO
ENDFOR
```

Fig.7. HBA procedure for $N=2^{m+n}$.

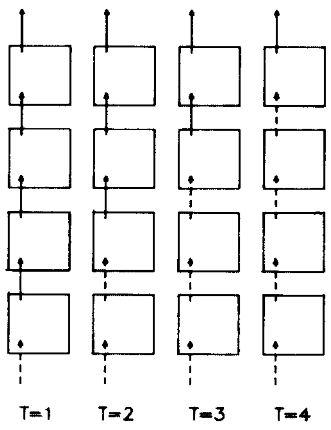


Fig.9. I/O pipelining for N=16.

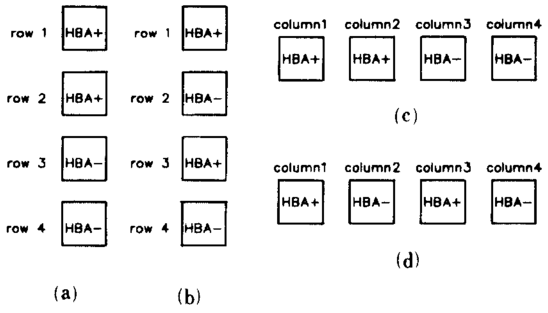


Fig.8. HBA computation for N=16.
(a) 1st Process (q=1).
(b) 2nd Process (q=2).
(c) 3rd Process (q=3).
(d) 4th Process (q=4).

```
DO IN PARALLEL
  LOAD DATA $F_{m+n}(k)$ INTO ROUTING REGISTER A
ENDDO
FOR i ← $2^m$ TO 1 DO
  FOR r ← 1 TO $2^m$ DO IN PARALLEL
    r' = r − 1
    FOR c ← 1 TO $2^n$ DO IN PARALLE
      IF ( (r−i) < 0 ) THEN
        DATA $F_{m+n}(k)$ IN P(r,c) MOVE TO P(r',c)
      ELSE
        NEW INPUT DATA $F_0(k)$ IN P(r,c) MOVE TO P(r',c)
      ENDIF
    ENDFOR
  ENDFOR
ENDFOR
DO IN PARALLEL
  LOAD NEW INPUT DATA $F_0(k)$ INTO ACCUMULATOR IN HBAU
ENDDO
```

Fig.10. I/O pipelining procedure for $N=2^{m+n}$.

3. *I/O Pipelining*

Immediately after the whole FFT computation has been completed, final output data are moved to their north-neighbor PEs. At the same time, newly inputed data which will be used in the another FFT computation, are loaded into PEs in row major order. Fig.(9) illustrates this I/O pipelining for N=16. And, Fig.(10) shows the I/O pipelining procedure for the general case, $N=2^{m+n}$ It is illustrated in Fig.(10) that the output of final data and the input of new data are pipelined through m clocks.

III. PE Architecture

The proposed 2-D systolic array consists of 2^m x 2^n identical processing elements for 2^{m+n} point FFT. As depicted in Fig.(11), a PE is composed of data routing unit, half butterfly arithmetic unit and control logic unit.

1. *Data Routing Unit (DRU)*

DRU consists of two multiplexers and one data register as shown in Fig.(12). To perform data shuffling operation, the data obtained by HBA, which is to be used for the next process, is loaded from HBAU into registers in DRU. On the other hand, the shuffled data that will be used in HBA, is loaded from DRU into registers in HBAU.

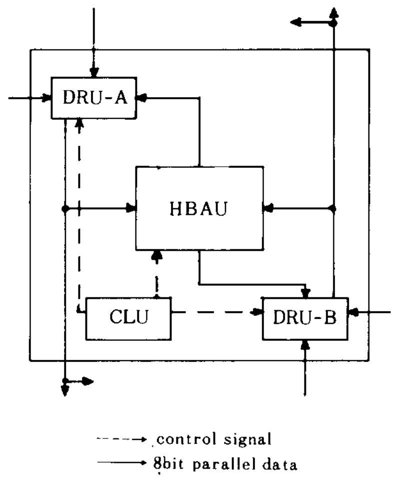


Fig.11. PE architecture.

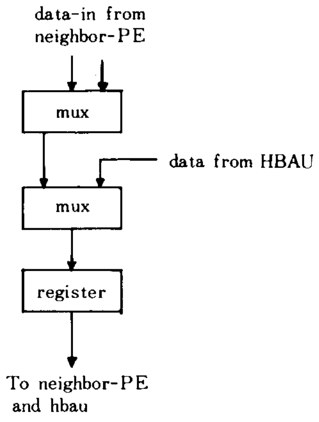


Fig.12. Data routing unit.

During row shuffling, DRU-A transfers data from its north-neighbor PE to its south-neighbor PE, while DRU-B receives data from its south-neighbor PE and sends it to its north-neighbor PE. Also, during column shuffling, DRU-A receives data from its west-neighbor PE and sends it to its east-neighbor PE, while DRU-B transfers data from its east-neighbor PE to its west-neighbor PE.

VLSI implementation of most bit-parallel systolic algorithm suffers from impractical pin counts. To overcome this problem, data are moved 8 bits at a time during data shuffling and I/O pipelining operations.

2. Half Butterfly Arithmetic Unit (H^rAU)

HBAU executes one of the HBA operations, either HBA+ or HBA-. In eq.(3), data A, B and Twiddle factor W are all complex numbers, thus we can rewrite it as follow,

$$HBA = [Ar \,\&\, (BrWr - BiWi)] + j\,[Ai \,\&\, (BrWi + BiWr)] \quad (4)$$

$$\text{where, } j = \sqrt{-1}$$

From eq.(4), we can see that four multipliers and four adders are required to compute directly the HBA. Distributed arithmetic [6,13,15] was adopted in HBA to perform multiplierless arithmetic and to reduce chip area.

A complex multiplication in eq.(4) can be expressed as

$$P = B \cdot W$$
$$= (Br + jBi)\,(Wr + jWi)$$
$$= Pr + jPi \quad (5)$$

By expressing each component of B as a fractional N-bits number in 2's complement notation (Brn, Bin), and substituting them into eq.(5), we obtain

$$Pr = \frac{1}{2}\,(Wr - Wi)2^{-(n-1)} + \sum_{n=o}^{N-1} Qrn\,(Brn,Bin)2^{-n}$$

$$Pi = \frac{1}{2}\,(Wr + Wi)2^{-(n-1)} + \sum_{n=o}^{N-1} Q\,in\,(brn,Bin)2^{-n} \quad (6)$$

The values of the Qrn and Qin are determined by Brn and Bin. From Eq.(6), it is clear that complex multiplication by distributed arithmetic requires (Wr + Wi)/2 and (Wr-Wi)/2 rather than Wr and Wi.

Also, eq.(4) can be rewritten as

$$HBAr = Ar \,\&\, Pr$$
$$HBAi = Ai \,\&\, Pi \quad (7)$$

To perform the & operation in eq.(7), i.e., addition and subtraction, the same adders that used in distributed arithmetic can be reused.

Therefore, the HBA can be executed efficiently with only two adders, which occupy a smaller chip area than multiplier.

HBAU consists of two data registers(A,B), two Twiddle factor registers, accumulators (Acc), adders and some control logic. BLC (Binary Lookahead Carry) adder was used because of its regular layout and high arithmetic speed. Accumulators are used not only to accumulate intermediate results of HBA, but also to store data during shuffling operations.

Two Twiddle factor registers are used to store coefficient $(W_r + W_i)/2$ and $(W_r - W_i)/2$. Their initial values are set to 0.5 for 1st process. Also, the Twiddle factors that will be used in the following process are transferred from external memory into registers in DRU during HBA operation. Immediately after HBA has been completed, these new Twiddle factors are loaded into Twiddle factor registers in HBAU.

Overflow in the fixed register size within HBAU is prevented to adopt an automatic cell scaling (ASC) technique, which scales down every data by a factor of 2. This ASC is simple and fast at the expense of accuracy. Also, to compromise the area/precision criteria, data and coefficients have 16 bit complex 2's complement fixed point format.

Control Logic Unit (CLU) receives signals from external array controller, and generates various control signals which are used in DRU and HBAU.

3. *Testability*

With the increase of chip complexity, the design for testability issue has been growing in importance [16]. Various techniques of design for testability have been surveyed in [17]. Among many design for testability techniques, we employed built-in self-test (BIST) in PE design because of its low hardware costs, simple implementation and excellent test coverage.

A BIST utilizes on-chip pseudorandom test pattern generation and on-chip signature analysis, which are performed using registers in HBAU. In the test mode, data registers in HBAU and two Twiddle factor registers are used as a pseudorandom test pattern generator, and accumulators in HBA are used as a signature analyzer. These registers are configured into linear-feedback shift registers (LFSR) during the self-testing operation. Initial test patterns are supplied through DRU from outside. After initial test patterns are loaded into the test pattern generating registers, self-testing operation is initiated by an external test enable signal. During the test operation, two 16 bits LFSRs generate pseudorandom test patterns up to 2^{16}-1 while the output LFSR compresses test responses to a 9 bit signature until test-end-detector detects a test stop signal. After test operation is completed, the final compressed contents of output LFSR, called signature, are compared with predetermined good signature by an on-chip comparator. According to the comparison result, a GO or NO-GO signal is generated, which specifies whether the tested circuit is good or faulty, respectively.

Our built-in self-testing circuit takes up only 3% of hardware overhead together with two additional I/O pins, which are for the test enable signal and the GO/NO-GO output signal.

IV. Implementations and Performance Simulation

The HBAU chip, which is composed of two BLC (Binary Lookahead Carry) adders, data and coefficients registers, was fabricated using a 3 um single metal P-well CMOS process. It contains about 7,000 transistors and measures about 7.3 x 8.0 mm^2 including 63 I/O pads. Fig.(13) is a microphotograph of the HBAU chip.

Also, An one-PE chip was designed using standard cell library and BLC adder macrocell with aid of auto-routing software. It consists of about

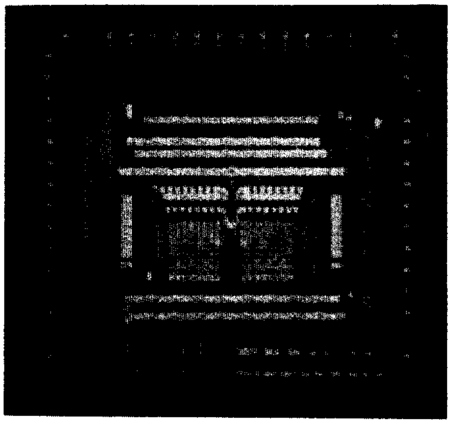


Fig.13. A microphotograph of the HBAU chip.

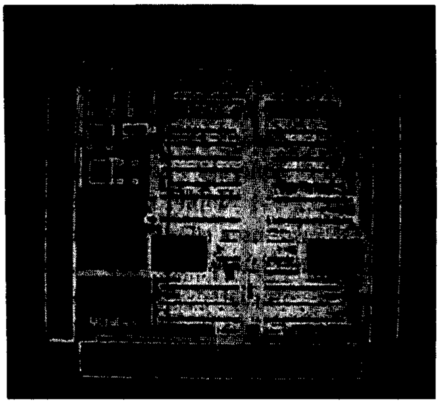


Fig.14. Layout of one-PE chip.

6000 transistors and 68 I/O pads in a 3.4 x $2.8mm^2$ area. Its layout is shown in Fig.(14). Currently, the chip is being fabricated using a 2 um, double metal, p-well CMOS process. Built-in self-testing circuits were employed at the expense of 3% hardware overhead.

The total execution time T_{FFT} for 2^{m+n} point can be expressed by

$$T_{FFT} = T_{I/o} + T_B + T_s \qquad (8)$$

where
$T_{I/o}$ = total data I/o time
TB = total butterfly arithmetic time
T_s = total data shuffling time

Data I/O, butterfly arithmetic and shuffling time in the above equation can be calculated with,

$$T_{I/o} = 2T\,(2^m + 2) \qquad (9a)$$

$$T_s = 2T\,(\sum_{i=1}^{m} (2^{m-i} + 2) + \sum_{j=1}^{n} (2^{n-j} + 2)) \qquad (9b)$$

$$T_B = (m+n)\,T_{HB} \qquad (9c)$$

where : T : clock period

T_{HB} : half butterfly arithmetic time

The coefficient 2 in eq. (9-a) and (9-b) represents the partitioned transfer of the 16 bit data into 8 bits at a time during I/O pipeline and data shuffling operation.

The computation time of a half butterfly arithmetic is simulated to be about 500ns at a clock frequency of 20 MHz chosen by critical path delay. Based on this result, total FFT execution time for a 1024 point is estimated about 16.6 us by eq(9). In comparison, the FFT computation on this architecture is much faster than other DSP processors, [18,19], thus it is expected that the PE discussed in this paper will result in significant improvement in high speed real time processing.

V. Conclusion

In the development of complex VLSI chip, the optimality of the underlying algorithm determines the degree of concurrency and pipelining which eventually determines the performance of the chip. The basis of the VLSI system design philosophy encompasses not only the regularity and locality in communications but also the modularity in hardware structure.

This paper describes a two-dimensional systolic array for FFT, which is based on half butterfly arithmetic and efficient data shuffling operations. A new arithmetic method that employs the half butterfly arithmetic reduces data shuffling time by 2, and makes hardware simpler than that of conventional butterfly arithmetic. Also, the PE activity is improved to 100%.

Since the proposed VLSI architecture features regular and recursive structure, and offers a highly local and regular communication geometry, it is suitable for VLSI and WSI (Wafer Scale Integration) implementation. Also due to its expandability, a processor performing an arbitrary size FFT can be constructed by simply increasing the array size.

A result of fabrication of the HBA test chip, the HBA operation can be executed in about 0.5us with a 20MHz clock frequency. From the performance simulation, it is expected that this architecture can be applied to the high speed real time signal processing. Currently, an one-PE chip is being fabricated using a 2 um, CMOS technology.

In future, it will be possible to realize a wafer scale FFT processor that has the proposed regular and recursive architecture, if fabrication technology is available.

References

[1] H.C. Yung, and C.R. Allen, "VLSI Architecture for digital signal processing," Technical report, Univ. of Newcastle Upon Tyne.

[2] C.D. Thompson, "Fourier Transform in VLSI," *IEEE Trans. on Computer*, vol. C-32, 1983.

[3] E.H. Wold, and A.M. Despain, "Pipeline and parallel-pipeline FFT Processors for VLSI implementations," *IEEE Trans. on Computer*, vol. C-33, no. 5, May 1984.

[4] J.A. Jonston, "Parallel pipelined Fast Fourier Transformer," *IEE Proc. F*, Oct. 1983.

[5] M.K. Lee, "Multiproject Chip Development '85," Ministry of Science and Technology, Korea, July 1986.

[6] M.K. Lee, "Multiproject Chip Development '86", Ministry of Science and Technology, Korea, July 1987.

[7] K.W. Shin, "A Systolic Array for FFT computation," M.S. Thesis, Yonsei Univ. Korea, 1986.

[8] B.Y. Choi, "A 2 dimensional systolic FFT processor," M.S. Thesis, Yonsei Univ. Korea, 1987.

[9] B.Y. Choi, K.W. Shin, and M.K. Lee, "2 dimensional systolic FFT processor," *Proceedings of KIEE*, vol. 9, no. 2, Korea, 1986.

[10] T. Willey, T.S. Durrani, and R. Chapman, "An FFT systolic processor and its applications," *Proceeding of the IEEE ICASSP-84*, vol. 2, Mar. 1984.

[11] H.S. Lee, H. Mori, and H. Aiso, "Parallel processing FFT for VLSI implementation", *The Trans. of the IECE of Japan*, vol. E68, no.5, May. 1985.

[12] B.Y. Choi, B.H. Kang, J.K. Lee, K.W. Shin, B.R. Kim, M.K. Lee., "VLSI implementation of two dimensional FFT algorithm on systolic array", *Proceeding of TENCON '87-IEEE Region 10 Conference*, Korea, Aug. 1987.

[13] J.K. Lee, B.Y. Choi, K.W. Shin, C.B. Kim, and M.K. Lee, "DIT radix-2 Half Butterfly Arithmetic for systolic array architecture," *Proceedings of KIEE*, vol. 9, no. 2, Korea, 1986.

[14] J.W. Cooley, and J.W. Tukey, "An algorithm for the machine calculation of complex Fourier series," Math. Comput., 1965.

[15] S.A. White, "A simple FFT butterfly arithmetic unit," IEEE Trans on CAS, Apr. 1981.

[16] T.W. Williams, "VLSI Testing," North-Holland press, 1986.

[17] T.W. Williams, and K.P. Parker, "Design for testability-A survey", *Proceedings of the IEEE*, vol. 71, no. 1, Jan. 1983.

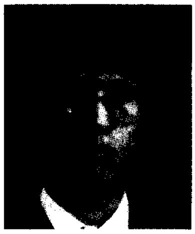
[18] R.W. Linderman, et. al, "CUSP:a 2-um CMOS digital signal processor," *IEEE J. of solid state circuits*, vol. 20, no. 3, Jun. 1985.

[19] J. Fox, G. Surace, P.A. Thomas, "A self-testing 2-um CMOS chip set for FFT applications," *IEEE J. of solid state circuits*, vol. 22, no. 1, Feb. 1987. *

Acknowledgment

The authors wish to thank to the Ministry of Science and Technology of korea for financial support of MPC project.

著 者 紹 介



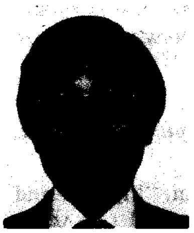
辛 卿 旭(正會員)

1961年 10月 26日生. 1984年 2月 한국항공대학교 전자공학과 학사학위 취득. 1986年 2月 연세대학교 대학원 전자공학과 석사학위 취득. 1986年 3月~현재 연세대학교 대학원 전자공학과 박사과정 재학중. 주관심분야는 VLSI design. Array Architecture, Parallel Processing, DSP칩 설계 등임.

◆

崔 炳 允(正會員)

1963年 12月 22日生. 1985年 2月 연세대학교 전자공학과 졸업, 학사학위 취득. 1987年 2月 연세대학교 대학원 전자공학과 석사학위취득. 1987年 3月~현재 연세대학교 대학원 전자공학과 박사과정 재학중. 주관심분야는 디지탈 신호처리를 위한 Systolic array, Wavefront array, Processor, Neural Network, VLSI Circuit Design 및 Logic Optimizer 등임.



李 文 基(正會員)

1965年 연세대학교 전기공학과 졸업. 1973年 연세대학교 전자공학과 공학박사학위 취득. 1970年 3月~1976年 8月 경희대학교 전자공학과 교수. 1976年 8月~1980年 7月 Oklahoma Univ. Microelectronics Research Lab. 연구원. 1980年 8月 Ph.D of Oklahoma Univ. 1980年 9月~1982年 8月 한국전자통신연구소 반도체부 책임연구원. 1982年~현재 연세대학교 전자공학과 교수. 주관심분야는 VLSI design 및 CAD 그리고 실리콘 압력센서 분야 등임.