A Digital Signal Processor With Programmable Correlator Array Architecture for Third Generation Wireless Communication System

Chi-Kuang Chen, Po-Chih Tseng, Yung-Chi Chang, and Liang-Gee Chen, *Fellow, IEEE*

***Abstract*—In this paper, a digital signal processor (DSP) with programmable correlator array architecture is presented for third generation wireless communication system. The programmable correlator array can be reconfigured as a chip match filter, code group detector, scrambling code detector, and RAKE receiver with low power consideration. The architecture and instruction set of the proposed DSP are specially designed for several key operations of wireless communications, such as channel estimation for RAKE combining, Viterbi algorithm and finite-impulse response filtering. According to the performance evaluation results, the proposed DSP outperforms other previously presented communication digital signal processors in terms of several crucial operations of wireless applications. A chip of the proposed DSP was implemented using hybrid design method where the timing critical components were full-custom designed and the other parts were cell-based designed under TSMC 0.35-μm CMOS 1P4M technology. We believe that the proposed system architecture would be useful for upcoming 3G mobile terminal applications.**

***Index Terms*—Digital signal processor, programmable correlator array, third generation wireless communication system.**

I. Introduction

A. Third Generation Wireless Communication System

WIRELESS communications become more and more popular in these years. The requirement of data transmission through mobile radio interface also increases rapidly. Thus some previous communication standards for speech transmission such as GSM are extended to support data transmission [1]. Even though, most of the third generation mobile communication proposals suggest the wide-band code division multiple access (WCDMA) [2], [3] in order to support future multimedia transmission which requires much higher data rate and wider bandwidth.

Fig. 1 shows a brief illustration of the WCDMA system diagram. In the transmitter side, the baseband data is first convolutional encoded and spread by a pseudonoise (PN) sequence. The spread signal is then transmitted out through RF module to the air. In the receiver side, the signal received by RF module from the air is first de-spread by correlators and then convolutional decoded to obtain the transmitted data. In this paper, we propose a system architecture including a digital signal processor (DSP) with a programmable correlator array for the de-spreading and convolutional decoding of WCDMA receiver.

B. Correlator Array and DSP

In CDMA systems, correlators are needed to de-spread the received signal [4], [5]. The input data is spread by a PN sequence, and the receiver has to de-spread the received signal into original symbols by calculating the correlation of input data and the PN sequence. The receiver adjusts the timing offset to search the maximum correlation value. This process is very time-consuming, and thus some modern CDMA receivers usually use many correlators to perform parallel search [5]. Furthermore, in order to support higher data rate services, multicode DS-CDMA (MC-CDMA) [6] is suggested in ETSI UMTS [7] and TR45.5 CDMA2000. That is, one user uses several codes to transmit his data. As a result, correlator plays an important role in CDMA systems. Besides, in order to effectively reuse the same correlator array, it is quite important to design a reconfigurable correlator array architecture that can be used both in code acquisition [8] and code tracking [9]. The detailed issues of the reconfigurability will be discussed in Section II.

In addition to reconfigurability, low-power design must also be considered because correlator array usually consumes large chip area. There have been some existing methods to reduce the power consumption of correlator array, such as using sign-magnitude number system [5] or using the redundant codes to create zero terms [10], [11]. In this paper, we introduced a low-power tri-code correlator based on the later concept. From the tri-code correlator, the programmable correlator array architecture is then illustrated.

In wireless devices, the programmable digital signal processors are widely used as baseband signal processing engine to support necessary system flexibility and upgradability. In most CDMA systems, the DSP is used to process the symbol-rate data [12]. Because much higher data rate is required in third generation wireless communication system, a more powerful DSP is very desired for the acceleration of symbol-rate data processing to meet system requirement. For these purposes, we propose a powerful DSP with specially designed architecture and instruction set for third generation wireless communications.

Manuscript received May 26, 2000; revised December 14, 2001. This paper was recommended by Associate Editor X. Song.

C.-K. Chen is with the DSP group of VIVOTEK Inc., Taipei 106, Taiwan, R.O.C.

P.-C. Tseng, Y.-C. Chang, and L.-G. Chen are with the Graduate Institute of Electronics Engineering, Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan, R.O.C. (e-mail: lgchen@video.ee.ntu.edu.tw).

Publisher Item Identifier S 1057-7130(01)11374-1.

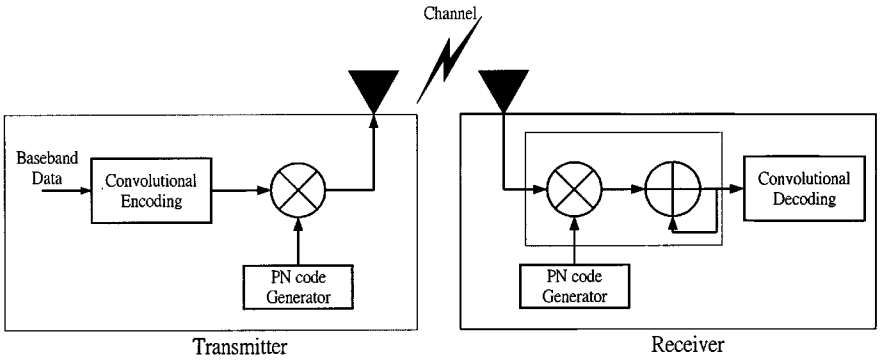

Fig. 1. Brief illustration of the WCDMA system diagram.

This paper is organized as followed, the programmable correlator array architecture is presented in Section II. Section III describes the detailed architecture of the proposed DSP. In Section IV, the performance evaluations of the proposed DSP and several previous arts are taken into consideration. The design flow and chip implementation issues of the DSP is discussed in Section V. Finally, a summary is given to conclude this paper.

II. Programmable Correlator Array

A. Tri-Code Correlator

Traditional correlator can only correlate the input signal with code 1 or -1. In order to improve the programmability and reduce power consumption, the tri-code correlator shown in Fig. 2 is developed.

The LSB of the CODE signal represents the real code of input data to correlate with. In hardware implementation, 0 represents that the code equals to 1 and 1 represents that the code equals to -1. The MSB of the CODE signal is gated to the clock signal of the accumulator. If the MSB of CODE is set to 1, this correlator behaves as a traditional correlator. Otherwise, this correlator will stop correlating the input signal. Thus, this is a tri-code correlator that can correlate with 1, -1, and 0. If two tri-code correlators are connected with some preprocessing of the two input codes as shown in (1), it becomes a dual-code correlator that can reduce power consumption [13].

$$\begin{cases} \mathrm{CODE}_1[0] = a \\ \mathrm{CODE}_1[1] = \sim (a \wedge b) \end{cases} \text{and} \begin{cases} \mathrm{CODE}_2[0] = a, \\ \mathrm{CODE}_2[1] = a \wedge b. \end{cases} \quad (1)$$

In (1), "a" and "b" are the two input codes. After correlated, the two outputs of the two correlators are added to get the correlation value for code "a" and subtracted to get the correlation value for code "b".

B. Programmable Correlator Array

From Third Generation Partnership Project (3GPP) drafts, there should be two phases in de-modulator. The first phase is code acquisition [14], [15] and the second phase is code tracking. During the acquisition phase, three steps are performed in turn: First, a chip match filter is needed to match the primary synchronization channel in order to find out the slot position. Next, 17 correlators are needed to find out 16 consecutive slot numbers so that the code group information and frame boundary can be got by looking up code group table. Finally, 16 correlators are needed to identify the scrambling code if there are 16 codes in each group. After these three steps, code acquisition is completed and code-tracking phase can be started. During the tracking phase, a simple early–late delay lock loop will be used and a RAKE receiver is also needed to eliminate the multipath effect. If all above operations are needed to perform on a single architecture, then a reconfigurable correlator array that can be programmed as different architectures for different phases or steps must be considered.

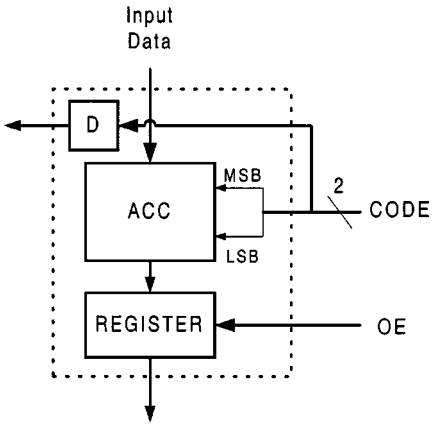

Fig. 2. Block diagram of tri-code correlator.

From the above analysis, a programmable correlator array architecture with low-power consideration is proposed to meet the two phases requirement as shown in Fig. 3, where the TCC in Fig. 3(a) is a tri-code correlator as mentioned before. This architecture looks like the correlator array architecture in [16], but the proposed one saves a lot of chip area because it doesn't need the data unit delay in [16]. According to the simulation result in EPIC TimeMill, our architecture can save about 20% power consumption when compared with [16].

In Fig. 3(b), the CODE signal is 17 by 2 bits wide. The 17 pairs of codes can be fed into 17 correlator bank elements (CBEs) separately by multiplexers. We will show how the architecture can be reconfigured to meet the acquisition phase and tracking phase requirements.

1) Slot Synchronization: When doing slot synchronization, the primary synchronization code is fed into correlator bank and

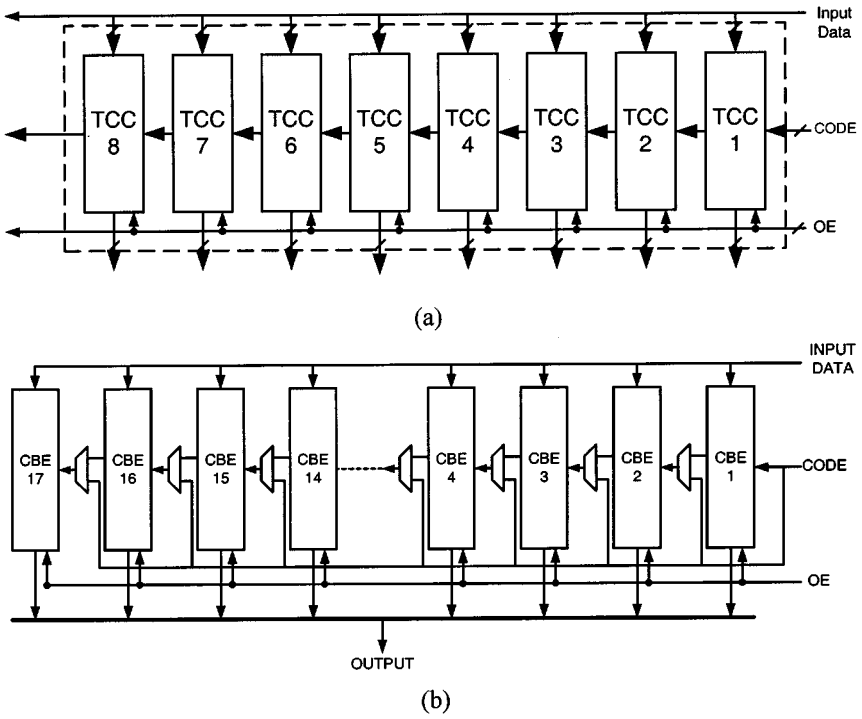

Fig. 3. Architecture of: (a) correlator bank element and (b) correlator bank.

the codes from previous stage CBE are fed into next stage CBE. By this approach, the correlator array can be used to substitute a chip match filter that consumes large chip area. But it may need more time to finish slot synchronization.

2) Frame Synchronization: When doing frame synchronization, 17 secondary synchronization codes are needed as CODE signal for 17 CBEs. These 17 codes also can be preprocessed by using (1) before fed into correlator bank in order to reduce power consumption. After correlation, the outputs are sent to DSP to do some postprocessing to get the actual correlation values.

3) RAKE Receiver: The architecture also can be configured as a RAKE receiver. The scrambling codes that have different delays can be combined as CODE signal. After correlation, the outputs are sent to DSP to do channel estimation and RAKE combining.

III. DSP

The proposed DSP is specially designed for symbol-rate I/Q channel data processing, such as channel estimation, RAKE combining, Viterbi algorithm, and finite-impulse response (FIR) filtering operation that is widely used in communication systems. The design issues for the proposed DSP are discussed in the following paragraphs.

A. WCDMA System Simulation

Before designing the DSP architecture, the simulation of WCDMA system is first considered. After the simulation, several key operations that can be well executed by the DSP can then be extracted. Fig. 4 shows the proposed complete system diagram of the WCDMA receiver that contains a correlator array, a digital signal processor, some code generators, and a

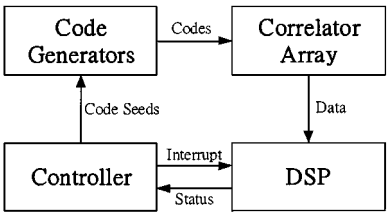

Fig. 4. Proposed WCDMA receiver system diagram.

system controller. The controller controls the tasks of baseband processing, providing the code seeds to the code generators and handshaking with the DSP. The code generators receive the code seeds and then generate the codes to correlator array. The processing in chip-rate is handled by the correlator array, and the symbol-rate processing is handled by the DSP.

After the correlator array architecture is designed, we use C language to simulate the WCDMA system baseband part based on 3GPP documents. The simulation program contains three major parts: transmitter, channel model, and receiver. The transmitter contains four physical channels: synchronization channel, primary common control physical channel, secondary common control physical channel, and dedicated physical channel. These are spread by its own channelization code that preserves orthogonality. After spread, they are summed and multiplied by the scrambling code. Afterwards, the signal summed with synchronization codes is fed into a pulse shaping filter. The channel model contains multipath effect, Rayleigh fading effect, AWGN, and carrier offset. The delay profile suggested in 3GPP document for four paths is used.

We first perform the floating-point simulation, and then the fixed-point simulation. From the simulation result, we analyze

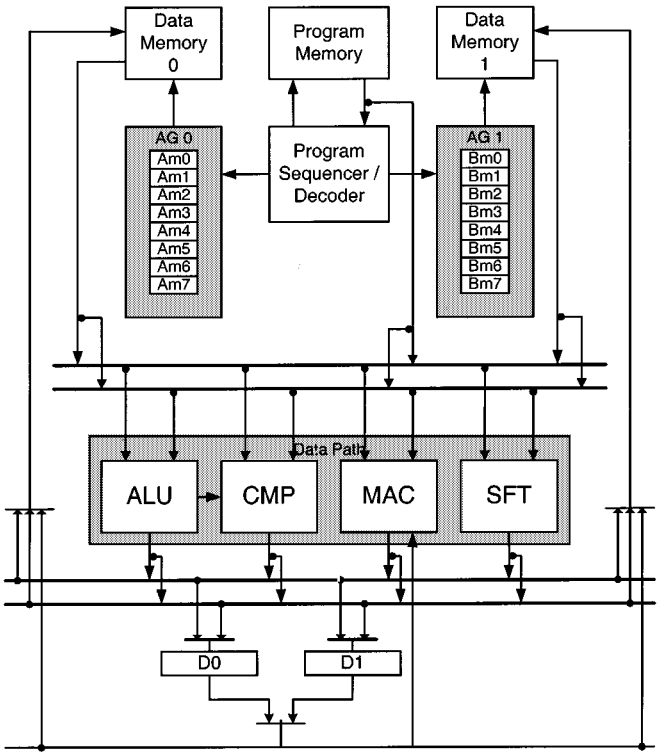

Fig. 5. Block diagram of proposed DSP.

that the error of 6-bit word length precision would be acceptable for the correlator output to the DSP. Besides, during the simulation, we also find out that there exists large amount of several key operations among each decoding phase. According to these results, we extract several key operations out, and then define special instructions and processor architecture for these operations. The details are discussed in the following sections.

B. DSP Architecture Overview

Fig. 5 shows the block diagram of the proposed DSP. The DSP uses the modified Harvard architecture with two data memories and one program memory, each has a 16-bit addressing space. The data memories use 16-bit word length and the program memory uses 28-bit word length, respectively. Each data memory has its own address generator (AG) which has eight index registers and modulo addressing mode.

The datapath contains four components: the arithmetic logic unit (ALU), the multiply-accumulate unit (MAC), the barrel shifter (SFT), and the comparator (CMP). The inputs of ALU, SFT, accumulator of MAC, and CMP are 40-bit wide. And the inputs of multiplier are 16-bit wide. The output of data path can be stored into two 40-bit registers (D0 and D1) or two data memories.

There are five pipeline stages in the DSP: instruction fetch, instruction decode, operand read, execution, and write back. One instruction can be executed in one clock cycle.

C. SubWord Parallel (SWP)

According to our simulation results of WCDMA system, 6-bit word length precision is enough for correlator output. Therefore, a normal 16-bit DSP datapath can be separated into two 8-bit data for I channel and Q channel, respectively. By this subword parallel (SWP) architecture, the symbol rate I/Q channel data processing can be effectively accelerated. Thus, the proposed DSP can support two 8-bit operations or one 16-bit operation at one time. In some special cases, it also supports two 16-bit operations when the inputs of datapath come from two 40-bit accumulators. The data format is shown in Fig. 6.

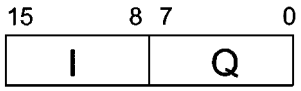

Fig. 6. Symbol rate data format of proposed DSP.

TABLE I
COMPUTATIONAL INSTRUCTION FORMAT

Operation	X	Y	D	
Operation	X	Y	A	B
Operation	X	D		

D. Instruction Set

The proposed instruction set can be classified into four kinds: computation, data movement, program flow, and special instructions.

1) Computational Instructions: These instructions contain datapath operations such as ALU, CMP, MAC, and SFT operations. The instruction format can be simplified as in Table I.

In Table I, *X* and *Y* means input operand that can be internal registers or data memory address pointers. If *X* or *Y* is a pointer, the data memory content of the pointed address is read out and fed into datapath. *D* means the output destination of datapath output, and it also can be register or data memory address pointer. *A* and *B* are also output destinations, but they only can be memory address pointers. The detailed instruction information of computational type is listed in the Tables V–VIII of the Appendix.

2) Data Movement Instructions: These instructions process the data movement in the DSP. It contains five formats of data movement: memory to register, register to register, register to memory, memory to I/O bus, and I/O bus to memory. The detailed instruction information is listed in Table IX of the Appendix.

3) Program Flow Instructions: These instructions involve the change of program counter value such as branch, looping, jump, and return. The detail instruction list is shown in Table X of the Appendix.

4) Special Instructions: These instructions are specially designed for wireless communications. We will discuss each in detail in the Section III-E. The special instruction list is shown in Table XI of the Appendix.

E. Special Instructions for Third Generation Wireless Communications

1) Channel Estimation for RAKE Combining: The main idea of channel estimation is to use the known pilot symbol to

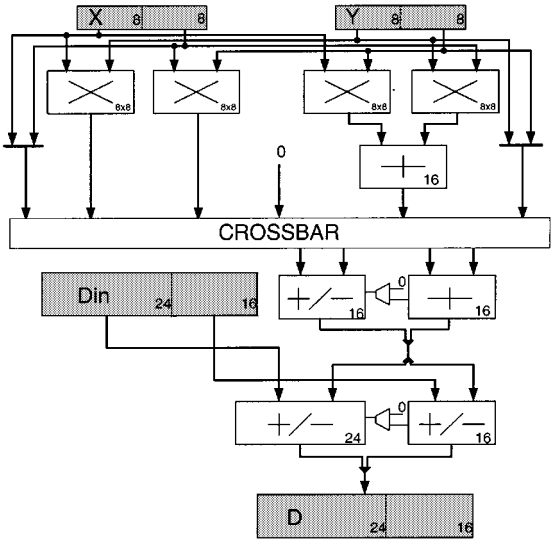

Fig. 7. Block diagram of MAC with SWP architecture.

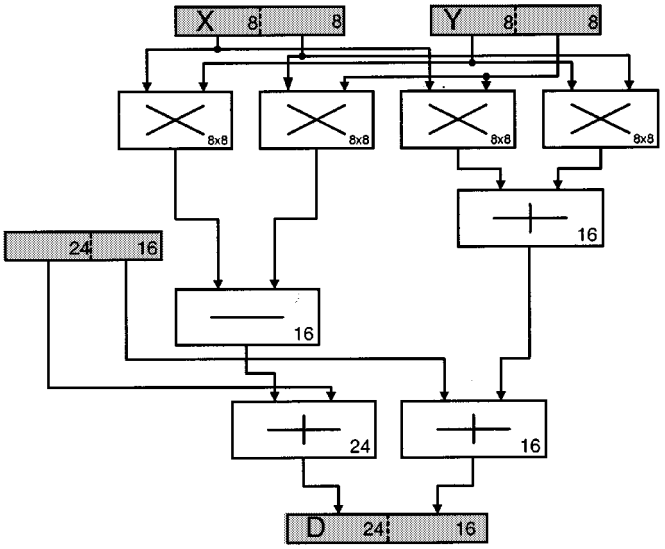

Fig. 8. Data flow of complex MUL/MAC operation.

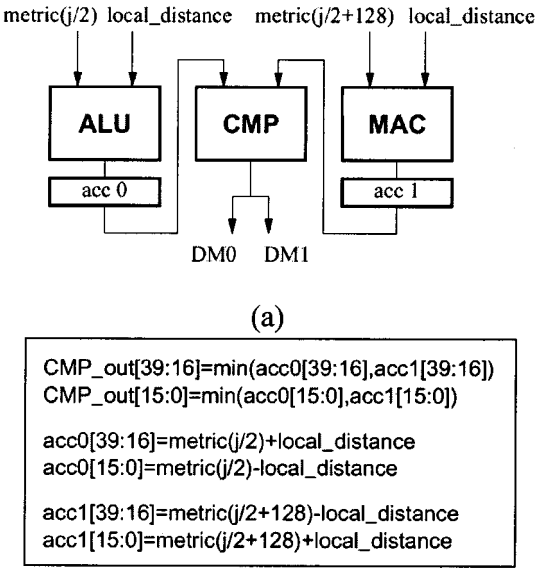

Fig. 9. Two ACS operations in one clock cycle. (a) Data flow of dual ACS operation. (b) Actual operations.

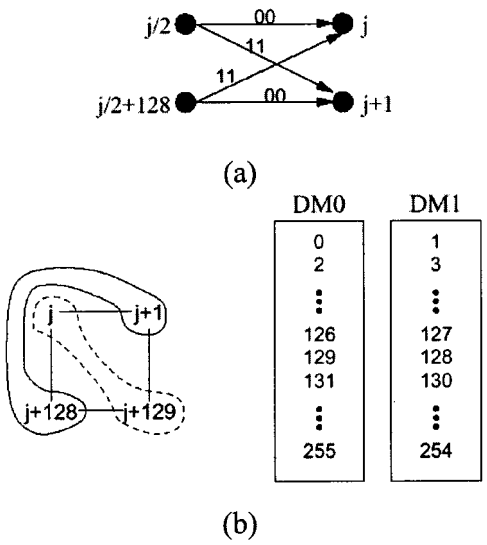

Fig. 10. Considerations for Viterbi algorithm. (a) Butterfly in the trellis diagram for $R = 1/2$ and $K = 9$. (b) Data arrangement of metric data in the memory banks.

obtain current mobile channel response. Then this information can be used for maximum-ratio-combining as the spirit of RAKE receiver [2]. In this manner, some complex arithmetic such as multiplication and multiplication-accumulation are needed. In a normal 16-bit DSP, it will take six cycles to complete a complex multiplication that contains four multiplications and two add/subtraction operations. Because the proposed DSP support either 8-bit or 16-bit data format, it has four 8-bit multipliers as shown in Fig. 7. Thus, it is sufficient to perform a complex multiplication/multiply-accumulation (complex MUL/MAC) in one instruction cycle. The results of complex MUL/MAC become two 16-bit words for real and imaginary part. These two words can be stored to two 16-bit data memories or be saved in a 40-bit accumulator. The data flow of complex MUL/MAC is shown in Fig. 8.

2) Viterbi Algorithm: Convolutional encoding is the most popular error correction coding method in wireless communication systems. Convolutional encoded data is decoded by using Viterbi algorithm on trellis diagram [17]. There are two steps in Viterbi decoding process as discussed in [18]. The first step is the metric update. In this step, the operation add-compare-select (ACS) is used. Because this step is most time-consuming in Viterbi decoding, TI C54x uses special instructions to speed up ACS operation. Even though, it still needs five instruction cycles to complete one butterfly calculation. We use SWP with single instruction streams multiple data streams (SIMD) [19] architecture to accelerate the ACS operation as shown in Fig. 9. The MAC bypasses the multipliers and behaves as a 24-bit substractor/adder and a 16-bit adder/substractor. The ALU behaves as a 24-bit adder/substractor and a 16-bit substractor/adder. At the same time, the comparator selects and saves the minimum distance of previous calculated path distances to data memories.

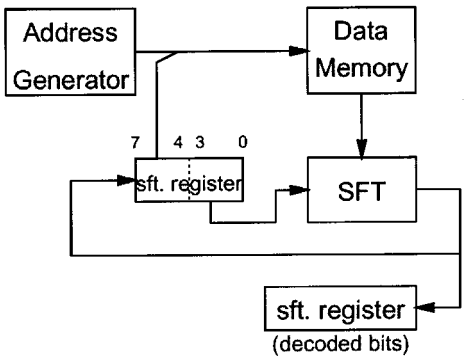

Fig. 11. Trace back circuit diagram which support $K = 5$ to 9.

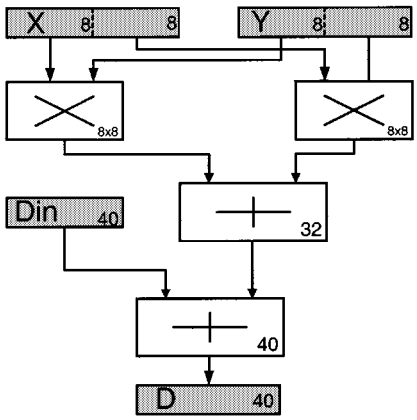

Fig. 12. Data flow of dual FIR filtering operation.

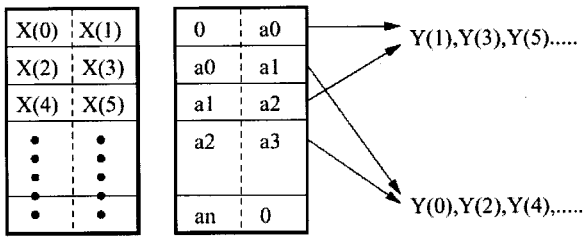

Fig. 13. Memory arrangement of FIR filtering operation.

Address	Instruction	Address	Instruction
0	SETCT 16,COUNT	13	SETCT 16,COUNT
1	LD 1,SRA	14	MV T,*Am0
2	LD 35,CB0	15	ST D0,Am2
3	LD 34,CB1	16	CLR D0
4	LD 1,*Bm0	17	DO 18
5	LD 1,SB0	18	FIR2 Am0,Bm0,D0
6	LD 1,IA0	19	SETCT 16,COUNT
7	LD 1,IA2	20	ADD SRA,T,T
8	LD 2,IB0	21	MV T,*Am0
9	LD 2,IB1	22	ST D0,Am2
10	LD 160,*Am2	23	CLR D0
11	DO 12	24	JUMP 11
12	FIR2 Am0,Bm0,D0		

Fig. 14. Example assembly code for a 33-tap FIR filter.

In other words, two ACS operations can be finished in one cycle if SWP and SIMD features are used at the same time. This is also the reason for separating CMP from ALU unit.

Obviously, two memory-read and two memory-write are needed per cycle when performing this dual-ACS operation. Thus, there may be some options for data memory type. At least one 2R2W port memory or two 1R1W port memories are needed. According to [20], when the port number for a SRAM equals to four, its size is about three to four times larger than two two-port SRAM. Thus, two 1R1W port memories are preferred. According to the indexing of the metric data shown in Fig. 10(a), its exclusion graph is drawn in the left of Fig. 10(b) based on only two 1R1W port data memories are available here. By this exclusion graph, the metric data must be arranged as shown in right-hand side of Fig. 10(b). In this way, the data flow is smooth for each trellis diagram stage. Without this consideration, it will need to use one 2R2W port SRAM as in [21] that consumes much more area than two 1R1W port SRAM, or need two cycles to write back the new metric data as TI C54x does.

When performing ACS operation, the two flags of CMP results are needed to be stored to a transition table for further trace back operation. Thus the two result bits are stored in a 16-bit shift register that shifts two bits per cycle. After the shift register is full, the content of shift register is moved to transition table in data memory.

The next step of Viterbi algorithm is trace back. Here we refer the architecture in [21] and support for constraint length (K) from 5 to 9. As shown in Fig. 11, the middle shift register is configurable from four bits to eight bits in order to support constraint length $K = 5$ to 9 and a special instruction called "CFGTRC" is executed to configure it before running trace back instruction "TRCBK." The four most significant bits of this shift register are used as four least significant bits of data memory address while constraint length equals to 9. The four least significant bits of this shift register are used as shift amount control for barrel shifter in order to shift right the transition data to correct state. As a result, the least significant bit of barrel shifter is the decoded data and is stored into a 16-bit shift register.

3) FIR Filtering: FIR filtering are widely used operation in many communication systems [22]. Since the proposed MAC architecture support 8-bit data format operation, it can speed up FIR filtering operation by a factor of two easily by active the left two 8-bit multipliers, a 32-bit adder below the cross-bar, and the last stage 40-bit adder in MAC. The data flow of this dual FIR operation is shown in Fig. 12, and the memory arrangement of input data and coefficients is show in Fig. 13. Input data sequence is stored in one data memory in packed form, and the filter coefficients are stored in another data memory also in packed form. Fig. 14 shows the example assembly code for a 33-tap FIR filter. Although there is some overhead when doing addressing registers initialization (address 0–10), it can be ignored if the input data stored in data memory 0 (addressed by Am0) is large enough.

F. Other Features

During slot synchronization or frame synchronization procedure, we need to compare I channel and Q channel square sum of different time delay. Thus the operation $I^2 + Q^2$ is needed.

TABLE II
PERFORMANCE COMPARISON OF SEVERAL COMMUNICATION DSPS

	TI C54x (1 MAC)	TI C55 (2 MACs)	LODE (2 MACs)	MDSP-II (2 MACs)	Proposed (1 MAC)
GSM conv. (K = 5)	0.58 MIPS	0.276 MIPS	0.44 MIPS	0.8 MIPS	0.093 MIPS
IS-95 conv. (K = 9)	6.57 MIPS	3.13 MIPS	5.2 MIPS	9.01 MIPS	1.38 MIPS
3G conv. @ 384 kbps	263 MIPS	121 MIPS	208 MIPS	360 MIPS	55 MIPS
FIR operation (N-tap)	N cycles	N/2 cycles	N/2 cycles	N/2 cycles	N/2 cycles
Complex MUL/MAC	4 cycles	2 cycles	2 cycles	2 cycles	1 cycle

TABLE III
NORMALIZED PERFORMANCE COMPARISON OF SEVERAL COMMUNICATION DSPS

	TI C54x (1 MAC)	TI C55 (2 MACs)	LODE (2 MACs)	MDSP-II (2 MACs)	Proposed (1 MAC)
Viterbi algorithm	24%	53%	31%	18%	100%
FIR operation (N-tap)	58%	116%	116%	116%	100%
Complex MUL/MAC	29%	58%	58%	58%	100%

From the proposed MAC architecture, it can support this operation easily by modifying the input data of FIR instruction. The same data whose format is shown in Fig. 4 is fed into the two input ports of MAC, and the control signals of MAC are the same as those of the FIR instruction.

The proposed DSP has five interrupt vectors that are useful to get data from correlator array. Besides, the DSP has a general input-output port that can be used to configure the correlator array.

Zero-overhead looping instruction is also supported by the proposed DSP. The "DO #addr" instruction specify the loop ending program address and also set the next address of this instruction as loop starting address. The program will execute between the loop starting address and the loop ending address. Before using this instruction, the 16-bit register "COUNT" must be configured which stores the looping times. Every time the program counter meets the ending address, the COUNT value automatically decreases one until COUNT becomes zero. At the same time, the program sequencer set the next program address to the loop starting address. Thus, no any extra cycles will waste while executing looping program codes.

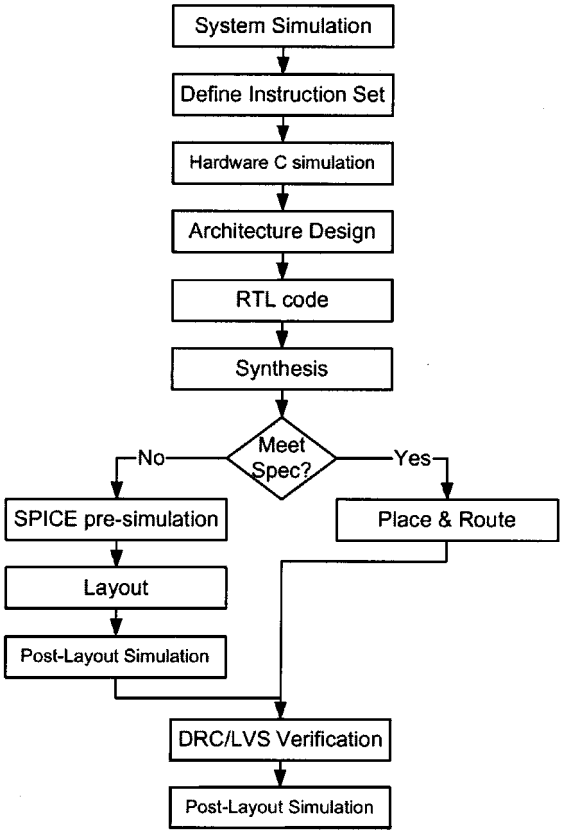

Fig. 15. Design flow of proposed DSP.

IV. PERFORMANCE EVALUATIONS

The proposed DSP has been compared with TI C54x [23] and C55 [24], LODE from Atmel [25], and MDSP-II [26] that are also designed for communication applications. Table II summaries the performance of these DSPs in terms of convolutional decoding, FIR filtering, and complex arithmetic.

The first three rows are the convolutional decoding at $R = 1/2$ for GSM, IS-95, and 3G WCDMA, respectively. Since convolutional encoded data is decoded by using the Viterbi algorithm as mentioned before, thus we calculate the required processor MIPS for the Viterbi algorithm in each communication standard. The required instruction cycles for single bit processing of the Viterbi algorithm including ACS operation and trace back operation are calculated first. Then the supported bit rate in each standard, such as 9.6 K bit per second (bps) in GSM and 384 K bps in WCDMA, is multiplied by the previous calculated result to obtain the total required processor MIPS. The fourth row uses the N-tap FIR filtering operation to measure the required instruction cycles in each DSP. The fifth row compares the required instruction cycles for one complex MUL/MAC operation performed in each DSP.

From Table II, we can find out that the proposed DSP has about four times to eight times performance at convolutional decoding and two times to four times performance at complex arithmetic. It also has the same performance with some DSP's that have two MAC units when performing FIR filtering operation.

Because the proposed MAC uses SWP architecture that contains four 8-bit multipliers, the MAC would be a little slower than a normal MAC that contains only one 16-bit multiplier. From our simulation result, the proposed MAC is slower than a normal MAC about 14%. Thus we can normalize the performance comparison result as shown in Table III.

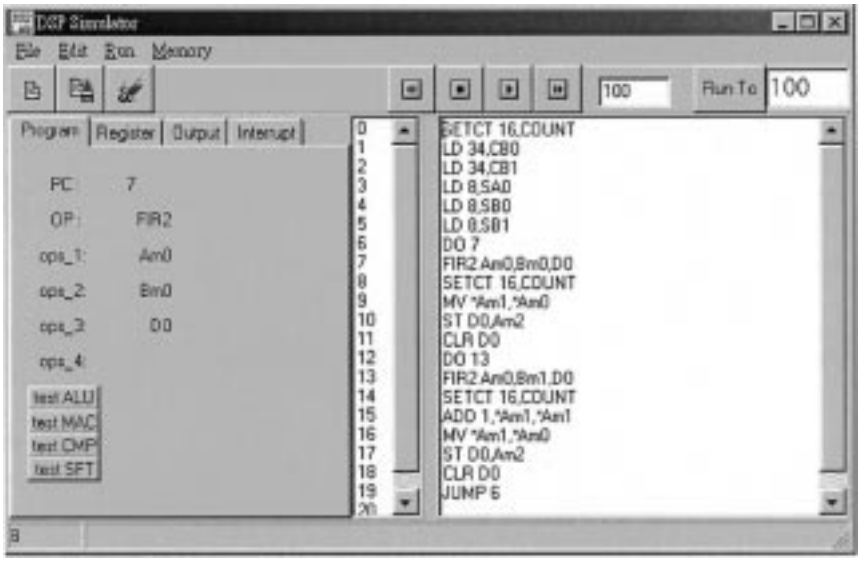

Fig. 16. DSP simulator interface.

V. Design Flow and Chip Implementation

As previous discussed, the WCDMA system simulation is first considered. After the simulation, several key operations that can be well executed by the digital signal processor can then be extracted. Then we target on these operations to define the instruction set and design the architecture for the DSP in order to accelerate these operations efficiently. The complete design flow is shown in Fig. 15. The design issues of system simulation, key operations extraction, and instruction set definition have been discussed in Section III, thus we focus on the remaining design issues in the Sections V-A–D.

A. Hardware C Simulation

After the instruction set is defined, a cycle-accurate simulator written in C language (hardware C) is developed to make sure the instruction set can work as we expect. Besides, when developing the simulator, we can also get some timing information for the proposed DSP architecture. If the architecture has any problem, it can be found early when running testing programs on the simulator and can be fixed easily. Without the help of this simulator, the total design time will increase a lot if we need to modify the DSP architecture when developing HDL design, this is because that design bugs are easily occurred during HDL design.

The simulator interface is shown in Fig. 16. Users can load DSP program file or write DSP program directly on the right-hand side of the simulator. The execution result can be export to file and some internal register or memory content can be shown directly on the simulator. As shown in Fig. 16 upper-right corner, the simulator can set the cycle number to be run or can run to the desired program address. Thus, it is very useful when developing and debugging application programs.

Besides, since the simulator is written in hardware C, it can be mapped to HDL design directly. The relation of hardware C and hardware pipeline is shown in Fig. 17, which shows the last stage in hardware pipeline should be executed first in C program.

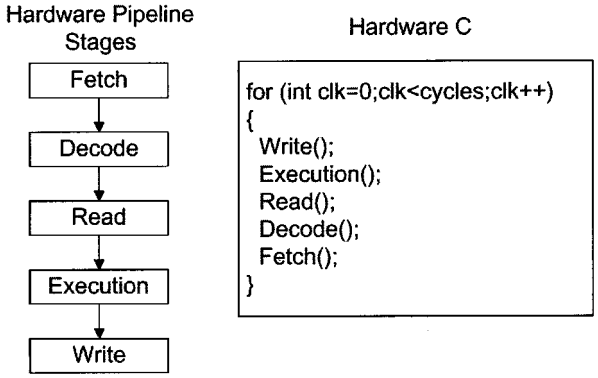

Fig. 17. Hardware C and hardware pipeline stages.

B. Cell-Based Design

After the instruction set is verified, we translate the hardware C to HDL design. Then SYNOPSYS design compiler is used to synthesize the gate level HDL using 0.35-μm cell library as our synthesis library. The synthesis report shows that the maximum clock frequency is up to 40 MHz and the critical path is the MAC unit.

C. Full-Custom Design

After synthesis, the timing critical multipliers and adders in DSP datapath are manually designed by full-custom design flow and the others are still designed by cell-based design flow. By this approach, the performance of the DSP can be improved about 50% compared to pure cell-based design version.

D. Chip Implementation

After the full-custom portion is completed, the automatic P&R is performed, and then the layout verification (DRC and LVS) is checked. Finally, we use EPIC TimeMill and PowerMill for postlayout simulation to obtain the chip features. Table IV summaries the chip features of the DSP. The chip layout is shown in Fig. 18.

TABLE IV
DSP CHIP FEATURES

Features	Proposed DSP
Pipeline	Five-stage pipeline
Max. Frequency	60MHz
Die Size	5.6 mm x 5.6 mm
Transistor Count	470K
On-Chip Memory	1K*28-bit(program), 2*2K*16-bit(data)
Technology	TSMC 0.35um 1P4M
Power Consumption	165mW@3.3V, 60MHz (Memories not included)

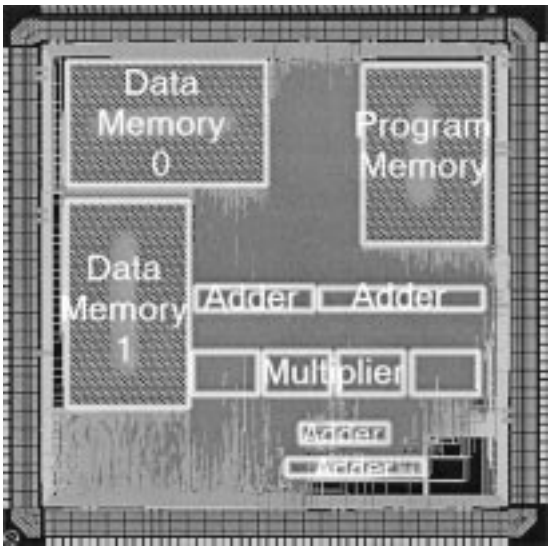

Fig. 18. DSP chip layout view.

TABLE V
ALU INSTRUCTION LIST

Instruction	Syntax	Description
ABD	X,Y,D	Absolute Distance
ABD16	X,Y,D ; X,Y,A,B	Absolute Distance (16-bit*2)
ABD8	a X,Y,D	Absolute Distance (8-bit*2)
ABS	X,D	D = abs(X)
ADD	X,Y,D	D=X+Y
ADD16	X,Y,D ; X,Y,A,B	DH=XH+YH , DL=XL+YL (16-bit*2)
ADD8	X,Y,D	DH=XH+YH , DL=XL+YL (8-bit*2)
SUB	X,Y,D	D=X-Y
SUB16	X,Y,D ; X,Y,A,B	DH=XH-YH , DL=XL-YL (16-bit*2)
SUB8	X,Y,D	DH=XH-YH , DL=XL-YL (8-bit*2)
ADDSUB8	X,Y,D	DH=XH+YH , DL=XL-YL (8-bit*2)
SUBADD8	X,Y,D	DH=XH-YH , DL=XL+YL (8-bit*2)
ADDSUB	X,Y,D ; X,Y,A,B	DH=X+Y , DL=X-Y
SUBADD	X,Y,D ; X,Y,A,B	DH=X-Y , DL=Y-X
ADDSUB16	X,Y,D ; X,Y,A,B	DH=XH+YH , DL=XL-YL (16-bit*2)
SUBADD16	X,Y,D ; X,Y,A,B	DH=XH-YH , DL=XL+YL (16-bit*2)
AND	X,Y,D	D=X and Y
OR	X,Y,D	D=X or Y
XOR	X,Y,D	D=X xor Y
NOT	X,D	D=not X
INV	X,D	D= -X
INV16	X,D; X,Y,A,B	DH= -XH , DL= -XL (16-bit*2)
INV8	X,D	DH= -XH , DL= -XL (8-bit*2)

TABLE VI
CMP INSTRUCTION LIST

Instruction	Syntax	Description
CMPL	X,Y,D ; X,Y,A,B	D=MAX(X,Y) , modify flag
CMPS	X,Y,D ; X,Y,A,B	D=MIN(X,Y) , modify flag
CMP16L	X,Y,D ; X,Y,A,B	DH=MAX(XH,YH) , DL=MAX(XL,XH) (16-bit*2)
CMP16S	X,Y,D ; X,Y,A,B	DH=MIN(XH,YH) , DL=MIN(XL,XH) (16-bit*2)
CMP8L	X,Y,D	DH=MAX(XH,YH) , DL=MAX(XL,XH) (8-bit*2)
CMP8S	X,Y,D	DH=MIN(XH,YH) , DL=MIN(XL,XH) (8-bit*2)

TABLE VII
MAC INSTRUCTION LIST

Instruction	Syntax	Description
MUL8 / MUL8U	X,Y,D ; X,Y,A,B	DL=XL*YL , DH=XH*YH
MAC8 / MAC8U	X,Y,D	DL=DL+XL*YL , DH=DH+XH*YH
CMUL / CMULU	X,Y,D ; X,Y,A,B	DH=XH*YH-XL*YL, DL=XH*YL+XL*YH
CMAC /CMACU	X,Y,D	DH=DH+XH*YH-XL*YL, DL=DL+XH*YL+XL*YH
CMSUB / CMSUBU	X,Y,D	DH=DH-XH*YH+XL*YL, DL=DL-XH*YL-XL*YH
MUL / MULU	X,Y,D ; X,Y,A,B	D=X*Y
MAC / MACU	X,Y,D	D=D+X*Y
MSUB / MSUBU	X,Y,D	D=D-X*Y

TABLE VIII
SFT INSTRUCTION LIST

Instruction	Syntax	Description
LSFT	X,D	logical shift X to left S bits
ASFT	X,D	arithmetical shift X to left S bits
ROT	X,D	Rotate X to left S bits
LSFT16	X,D ; X,Y,A,B	logical shift XL and XH to left S bits
ASFT16	X,D ; X,Y,A,B	arithmetical shift XL and XH to left S bits
ROT16	X,D ; X,Y,A,B	rotate XL and XH to left S bits

VI. CONCLUSION

In this paper, an enhanced DSP with programmable correlator array architecture is proposed for the third generation wireless communication systems. The low power programmable correlator array can be reconfigured as chip match filter, code group detector, scrambling code detector, and RAKE receiver. The proposed DSP is designed with several special instructions to accelerate the operations for symbol-rate data processing such as channel estimation, Viterbi algorithm, and FIR filtering. Since the processor MIPS needed for these key operations are effectively reduced, we believe that the proposed architecture would be useful in upcoming 3G systems that support much higher data rate then 2G systems.

APPENDIX

See Tables V–XI.

TABLE IX
DATA MOVEMENT INSTRUCTION LIST

Instruction	Syntax	Description
DLD	Am,Bm,X,Y	load memory content ad address Am and Bm to X and Y
DLD	Am,Bm,*Am1,*Bm1	Am=Mem(Am1), Bm=Mem(Bm1)
LD	Am,X	load memory content at address Am to X
LD	#imm,X	X=#imm (16-bit immediate value)
DST	DST X,Y,Am,Bm	save X and Y to memory at address Am and Bm
ST	ST X,Am	save X to memory at address Am
STI	*Am,Bm	Mem(Bm)=Am
MV	X,Y	move X to Y
IN	#addr,Am	Mem(Am)=IO_data, IO_address=#addr
IN	Am,Bm	Mem(Bm)=IO_data, IO_address=Am
OUT	Am,Bm	IO_data=Mem(Am), IO_address=Bm
OUT	Am,#addr	IO_data=Mem(Am), IO_address=#addr

TABLE X
PROGRAM FLOW INSTRUCTION LIST

Instruction Syntax		Description
JUMP	#addr	Jump to PC=#addr
CALL	#addr	CALL sub-routine at PC=#addr
RET		Return to the PC before CALL
RETI		Return from interrupt
DO	#addr	Do the following instructions until PC=#addr, then return to the next instruction of DO, repeat COUNT times.
SETCT	#ck,COUNT	COUNT=#ck
TEST	X,#num	if X's #num'th bit==1, excute next instruction, else ignore the next instruction.
TESTZ	X,#num	if X's #num'th bit==1, excute next instruction, else ignore the next instruction.

TABLE XI
SPECIAL INSTRUCTION LIST

Instruction	Syntax	Description
ACS	Am,Bm,Am1,Bm1,T	Mem(Am1)=MAX(D0H,D1H) Mem(Bm1)=MAX(D0L,D1L) D0H=Mem(Am)+TH, D0L=Mem(Bm)-TH D1H=Mem(Bm)-TH, D0L=Mem(Am)+TH save compare result bits to shift register SR0 T=0 : TH, T=1 : TL
ACSB	Am,Bm,Am1,Bm1,T	D0H=Mem(Am)-TH, D0L=Mem(Bm)+TH D1H=Mem(Bm)+TH, D0L=Mem(Am)-TH others same as ACS
CFGTRC	Am,#num	SRA=Mem(Am), SRA set to #num+4 bits
TRCBK		Traceback, save decoded bit to SR0
PACK	X,Y,D	D={XL, YL}
FIR2	X,Y,D	D=D + XL*YL + XH*YH
SQS	X,D	D=D + XL*XL + XH*XH
SQSA	X,D	D=XL*XL + XH*XH
SAT	D	Saturate D
CLR	D	Clear D

REFERENCES

[1] "TS02.60 v5.0.0—Digital Cellular Telecommunication System (Phase 2+)," General packet radio service (GPRS) ETSI, 1997.

[2] R. Prasad, "An overview of CDMA evolution toward wideband CDMA," *IEEE Commun. Surveys*, vol. 1, pp. 2–29, Jan. 1998.

[3] E. Dahlman, P. Beming, J. Knutsson, F. Ovesjö, M. Persson, and C. Roobol, "WCDMA—The radio interface for future mobile multimedia communications," *IEEE Trans. Veh. Technol.*, vol. 47, pp. 1105–1118, Nov. 1998.

[4] S. Sheng and R. Rodersen, *Low-Power CMOS Wireless Communications—A Wideband CDMA System Design*. Norwell, MA: Kluwer, 1998.

[5] W. Namgoong and T. Men, "Power consumption of parallel spread spectrum correlator architectures," in *Proc. IEEE Low Power Electronics and Design Conf.*, 1998, pp. 133–135.

[6] T. Dohi, Y. Okumura, A. Higashi, K. Ohno, and F. Adachi, "Experiments on coherent multicode DS-CDMA," in *Proc. IEEE Vehicular Technology Conf.*, 1996, pp. 889–893.

[7] "UMTS Terrestrial Radio Access: Concept Evaluation (UMTS 30.06) Tech. Rep.," ETSI, 1997.

[8] C. Deng and C. Chien, "A low energy architecture for fast pn acquisition," in *Proc. IEEE Int. Symp. Low Power Electronics and Design*, 1999, pp. 42–47.

[9] C.-C. Fan and Z. Tsai, "A differentially coherent delay-locked loop for spread-spectrum tracking receivers," *IEEE Commun. Lett.*, vol. 3, pp. 282–284, Oct. 1999.

[10] L. Liu, W. Lin, and C. Wang, "A pipelined digital differential matched filter fpga implementation and vlsi design," in *Proc. IEEE Custom Integrated Circuits Conf.*, 1996, pp. 75–78.

[11] C.-W. Ku, F.-Y. Kuo, C.-K. Chen, and L.-G. Chen, "Low power strategy about correlator array for CDMA baseband processor," in *Proc. IEEE Workshop Signal Processing Systems*, 1999, pp. 513–522.

[12] D.-S. Lyu, J.-S. Kim, K.-M. Ha, J. H. Lee, J.-M. Kim, Y.-G. Jeong, J. S. Ha, and N.-J. Park, "Implementation of wideband CDMA modem using processors and programmable logic devices," in *Proc. IEEE Int. Symp. Circuits and System*, 1997, pp. 2613–2616.

[13] C.-W. Ku, F.-Y. Kuo, C.-K. Chen, and L.-G. Chen, "Low power strategy about correlator array for CDMA baseband processor," in *Proc. IEEE Workshop Signal Processing Systems*, 1999, pp. 513–522.

[14] S. H. Ahn, J. T. Kim, and Y. H. Lee, "Efficient implementation of parallel correlators for code acquisition in DS-CDMA systems," in *Proc. IEEE Int. Symp. Circuits and System*, vol. 4, 1999, pp. 576–579.

[15] S. Sriram, K. Brown, and A. Dabak, "Low-power correlator architectures for wideband CDMA code acquisition," in *Conf. Rec. 33rd Asilomar Conf. Signals, Systems, and Computers*, vol. 1, 1999, pp. 125–129.

[16] C.-W. Ku and F.-Y. Kuo, "Software radio based re-configurable correlator/FIR filter for CDMA/TDMA receiver," in *Proc. IEEE Int. Symp. Circuits and System*, 2000.

[17] G. D. Forney, "The Viterbi algorithm," *Proc. IEEE*, pp. 268–278, 1973.

[18] H. Hendrix, "Viterbi decoding techniques in the TMS320C54x family application report," Texas Instruments Incorporated, 1996.

[19] J. Du, G. Warner, E. Vallow, and T. Hollenbach, "High-performance DSP's," *IEEE Trans. Signal Processing*, pp. 16–26, Mar. 2000.

[20] S. Dutta, K. J. O'Connor, W. Wolf, and A. Wolfe, "A design study of a 0.25-μm video signal processor," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, pp. 501–519, Aug. 1998.

[21] T. Ishikawa, H. Suzuki, N. Minamida, R. Yamanaka, M. Okamoto, H. Kabuo, and H. Taki, "W-CDMA hardware-related issues," in *Proc. Int. Conf. Communication Technology*, 1998, pp. S22-05-1–S22-05-5.

[22] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*. Englewood Cliffs, NJ: Prentice Hall, 1989.

[23] W. Lee, P. E. Landman, BrockBarton, S. Abiko, H. Takahashi, H. Mizuno, S. Muramatsu, K. Tashiro, M. Fusumada, L. Pham, F. Boutaud, E. Ego, G. Gallo, H. Tran, C. Lemonds, A. Shih, M. Nandakumar, R. H. Eklund, and I.-C. Chen, "A 1-V programmable DSP for wireless communications," *IEEE J. Solid-State Circuits*, vol. 32, pp. 1766–1776, Nov. 1997.

[24] "TMS320C55x DSP core technical CPU report," Texas Instruments Incorporated, Dallas, TX, 2000.

[25] I. Verbauwhede and M. Touriguian, "A low power DSP engine for wireless communications," *J. VLSI Signal Processing*, vol. 18, 1998.

[26] B.-W. Kim, J.-H. Yang, C.-S. Hwang, Y.-S. Kwon, K.-M. Lee, I.-H. Kim, Y.-H. Lee, and C.-M. Kyung, "MDSP-II: A 16-bit DSP with mobile communication accelerator," *IEEE J. Solid-State Circuits*, vol. 34, pp. 397–404, Mar. 1999.

Chi-Kuang Chen was born in Taipei, Taiwan, R.O.C., in 1976. He received the B.S. and M.S. degrees in electrical engineering from National Taiwan University, Taipei, Taiwan, R.O.C., in 1998 and 2000, respectively.

In 2000, he joined the DSP group of VIVOTEK Inc., and is currently engaged in audio codec programming on DSP processors.

Po-Chih Tseng was born in Tao-Yuan, Taiwan, R.O.C., in 1977. He received the B.S. degree in electrical and control engineering from National Chiao Tung University, Hsinchu, Taiwan, R.O.C., in 1999 and the M.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, R.O.C., in 2001. He currently is pursuing the Ph.D. degree at the Graduate Institute of Electronics Engineering, National Taiwan University.

His research interests include digital signal processor design, architecture design for image and video coding system, and reconfigurable computing for multimedia applications.

Yung-Chi Chang was born in Kaohsiung, Taiwan, R.O.C., in 1975. He received the B.S. and M.S. degrees from the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, R.O.C., in 1998 and 2000, respectively. He is currently working toward the Ph.D. degree in the same department.

His research interests include video streaming, multimedia communication, and VLSI architecture for image/video processing.

Liang-Gee Chen (S'83–M'86–SM'94–F'01) was born in Yun-Lin, Taiwan, R.O.C., in 1956. He received the B.S., M.S., and Ph.D. degrees in Electrical Engineering from National Cheng Kung University, Tainen, Taiwan, R.O.C., in 1979, 1981, and 1986, respectively.

From 1981 to 1986, he was an Instructor in the Department of Electrical Engineering, National Cheng Kung University and from 1986 to 1988, he was an Associate Professor with the same university. From 1987 to 1988, he served in the military and was an Associate Professor in the Institute of Resource Management, Defense Management College. In 1988, he joined the Department of Electrical Engineering, National Taiwan University. From 1993 to 1994, he was a Visiting Consultant at the DSP Research Department, AT&T Bell Laboratories, Murray Hill, NJ. In 1997, he was a Visiting Scholar at the Department of Electrical Engineering, University of Washington, Seattle. Currently, he is a Professor at National Taiwan University. His current research interests are DSP architecture design, video processor design, and video coding system.

Dr. Chen is a member of the Phi Tan Phi. He received the Best Paper Award from the ROC Computer Society, in 1990 and 1994. From 1991 to 1999, he received Long-Term (Acer) Paper Awards annually. In 1992, he received the Best Paper Award of the 1992 Asia–Pacific Conference on Circuits and Systems in VLSI design track. In 1993, he received the Annual Paper Award of the Chinese Engineer Society. In 1996, he received the Outstanding Research Award from NSC, and the Dragon Excellence Award for Acer. He was nominated the IEEE Circuits and Systems Distinguished Lecturer from 2001 to 2002. He was the general chairman of the 7th VLSI Design/CAD Symposium and the 1999 IEEE Workshop on Signal Processing Systems: Design and Implementation. He has been an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY , since 1996, and the IEEE TRANSACTIONS ON VLSI SYSTEMS, since 1999. He has been the Associate Editor of the *Journal of Circuits, Systems, and Signal Processing*, since 1999. He served as the Guest Editor of *The Journal of VLSI Signal Processing-Systems for Signal, Image, and Video Technology*. He is currently an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS —II: ANALOG AND DIGITAL SIGNAL PROCESSING.