A Low-Power Digit-Based Reconfigurable FIR Filter

Kuan-Hung Chen and Tzi-Dar Chiueh, *Senior Member, IEEE*

***Abstract*—In this brief, we present a digit-reconfigurable finite-impulse response (FIR) filter architecture with a very fine granularity. It provides a flexible yet compact and low-power solution to FIR filters with a wide range of precision and tap length. Based on the proposed architecture, an 8-digit reconfigurable FIR filter chip is implemented in a single-poly quadruple-metal 0.35-μm CMOS technology. Measurement results show that the fabricated chip operates up to 86 MHz when the filter draws 16.5 mW of power from a 2.5-V power supply.**

***Index Terms*—Canonical signed digit (CSD), finite-impulse response (FIR) digital filters, reconfigurable architectures.**

I. Introduction

FINITE-impulse response (FIR) filters play a crucial role in many signal processing applications in communication systems. A wide variety of tasks such as spectral shaping, matched filtering, interference cancellation, channel equalization, etc. can be performed with these filters. Hence, various architectures and implementation methods have been proposed to improve the performance of filters in terms of speed and complexity.

Recently, explosive proliferation in wired and wireless communication standards renders traditional FIR architectures less suitable for future communication needs. On the other hand, software radio [1]–[3] has gained much attention from the researchers worldwide due to a strong demand for reconfigurable communication systems capable of multi-standard operations. In light of this trend, programmability and reconfigurability need be taken into account in filter architecture design.

It is well known that the canonical signed digit (CSD) representation can be used to reduce the complexity of FIR digital filter implementation [4]–[6]. Encoding the filter coefficients using the CSD representation reduces the number of partial products and thus saves silicon area and power consumption in hardware implementation. Hence, this technique is popular for fixed-coefficient FIR filter implementation.

When applying the CSD representation to implementing programmable, rather than fixed-coefficient, FIR filters, it is natural to require that the number of programmable CSDs in a coefficient be fixed. However, for most filters, only a few taps require high-precision coefficients. Valuable hardware resources are wasted if all taps are implemented with the highest precision. To minimize hardware complexity, there have been several works on programmable FIR filter implementation and they all limit the number of allowable nonzero CSDs in every tap [7]–[9]. Unfortunately, this restriction can lower the coefficient precision and degrade the frequency response of the filter or it can induce a large overhead by assigning more CSDs than necessary to most taps. Another hardware-efficient implementation of programmable FIR filters with CSD coefficients has been presented in [10]. A 32-tap linear-phase filter, with two nonzero CSDs in each tap, is implemented. Additional nonzero CSDs can be allocated to specific filter taps, making it a reconfigurable FIR filter architecture. Nevertheless, some computational resources may still be unused and the critical path is quite long in some cases.

Due to a wide range in the filter coefficient precision for different applications, it is next to impossible to achieve reconfigurability without incurring overhead in hardware. As an example, a binary pseudorandom number (PN) code matched filter, which is an important block in CDMA receivers, requires only 1-bit coefficient precision while a pulse-shaping filter may require as high a coefficient precision as 16 bits. Since the tap complexity in these two cases can be very different, the tap-based reconfigurable FIR filter [10] is quite inefficient.

In this brief, we adopt the finest granularity for filter implementation and propose a reconfigurable FIR filter architecture with extreme flexibility. With this architecture, both the tap number and the number of nonzero digits in each tap can be arbitrarily assigned given that enough hardware resource is available. Techniques that yield an FIR filter with the minimum total number of CSD for a given frequency response specification can be found in [11], [12] and we have developed a software program based on [12]. The filter implemented with this architecture can be easily configured as a matched filter, a pulse-shaping filter, or other filters. Furthermore, the proposed FIR architecture also has scalability, modularity, and cascadability, making it amenable to VLSI implementation. Last but not least, the critical path delay in the proposed architecture stays quite invariant in different filter configurations.

The rest of the brief is organized as follows. In Section II, the reconfigurable FIR filter architecture is presented. Circuit design of a reconfigurable FIR filter chip is then presented in Section III. In Section IV, physical design and measurement results of the chip are given and discussed. Finally, Section V concludes this brief.

II. Reconfigurable FIR Filter Architecture

At first, note that an N-tap FIR filter can be formulated as

$$y[n] = \sum_{i=0}^{N-1} h_i \cdot x[n-i] \tag{1}$$

Manuscript received February 19, 2005; revised June 12, 2005. This work was supported in part by the National Science Council, Taiwan under Grant NSC 92-2220-E-002-002 and by MediaTek Inc. This paper was recommended by Associate Editor S. L. Netto.

K.-H. Chen is with the Department of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan, R.O.C. (e-mail: volley@analog.ee.ntu.edu.tw).

T.-D. Chiueh is with the Graduate Institute of Electronics Engineering and Department of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan, R.O.C. (e-mail: chiueh@cc.ee.ntu.edu.tw).

Digital Object Identifier 10.1109/TCSII.2006.875373

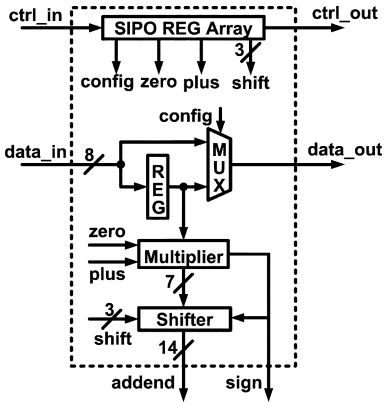

Fig. 1. Block diagram of DPU in the proposed reconfigurable filter.

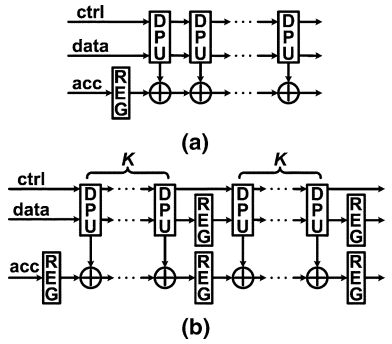

Fig. 2. Architecture of the proposed reconfigurable FIR filter. (a) General architecture. (b) Pipelined architecture.

where $x[n]$ and $y[n]$ are the input and output sequences of the filter, respectively, and h_i is the ith coefficient of the filter impulse response. If a coefficient, h_i, is expressed in the CSD format as $h_i = \sum_{k=0}^{M_i-1} d_{i,k} \cdot 2^{-p_k}$, we can rewrite (1) as

$$y[n] = \sum_{i=0}^{N-1} \sum_{k=0}^{M_i-1} d_{i,k} \cdot 2^{-p_k} \cdot x[n-i] \tag{2}$$

where $d_{i,k} \in \{-1, 0, 1\}$, $p_k \in \{0, \ldots, L\}$, and $L+1$ is the word length of the coefficients. If $h_i \neq 0$, then M_i is the number of nonzero digits in h_i. Otherwise, M_i is one and a "0" valued signed digit is assigned to this tap. A "0" valued signed digit does not contribute to the filter output but provide a unit delay in data path to delay the input data samples appropriately.

A. Digit Processing Unit (DPU)

Most FIR filters proposed before are implemented using a tap as the basic building block. A tap is designed to evaluate the term $h_i \cdot x[n-i]$ in (1) and then several taps constitute an FIR filter. After examining (2) carefully, one sees that if a basic building block is designed to evaluate the term, $d_{i,k} \cdot 2^{-p_k} \cdot x[n-i]$, then both the flexibility in the number of taps and the number of nonzero digits in each tap can be achieved.

To meet this requirement, a digit processing unit (DPU), as shown in Fig. 1, is designed. During the initialization, control signals can be serially shifted into a serial-in-parallel-out (SIPO) register (REG) array on the top. In each DPU, three control signals **plus**, **zero**, and **shift**, are derived from the corresponding digit in the ith tap coefficient $d_{i,k} \cdot 2^{-p_k}$. Two outputs **addend** and **sign** corresponding to the partial product $d_{i,k} \cdot 2^{-p_k} \cdot x[n-i]$ are evaluated by the multiplier and the shifter. Another control signal **config** controls the multiplexer in order to select either the buffered or the unbuffered input as $x[n-i]$ based on the filter configuration.

By cascading several DPUs, appropriately configuring the multiplexers in those DPUs, and summing up all the output of the DPUs and the accumulated sum as shown in Fig. 2(a), we can implement an FIR filter with variable number of CSDs in each tap. For the last digit of each tap, the multiplexer in the corresponding DPU selects the buffered data as the output. Table I shows the input data corresponding to each DPU in a 3-tap filter having 3, 3, and 2 CSDs in h_0, h_1, and h_2, respectively.

TABLE I
DATA COMPUTED IN DPUS WHEN CONFIGURED AS 3-TAP FILTER

Time	Tap 1			Tap 2			Tap 3	
Slot	DPU 1	DPU 2	DPU 3	DPU 4	DPU 5	DPU 6	DPU 7	DPU 8
0	x[0]	x[0]	x[0]	x[-1]	x[-1]	x[-1]	x[-2]	x[-2]
1	x[1]	x[1]	x[1]	x[0]	x[0]	x[0]	x[-1]	x[-1]
2	x[2]	x[2]	x[2]	x[1]	x[1]	x[1]	x[0]	x[0]

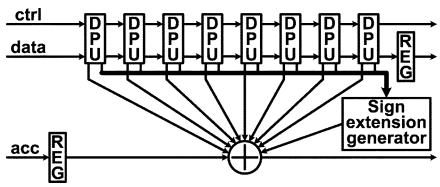

Fig. 3. Block diagram of the PE.

Even though the architecture we depicted in Fig. 2(a) can be implemented directly, usually, it is necessary to insert pipeline registers in the filter to achieve reasonable speed performance. The architecture that has one pipeline stage for K DPUs is illustrated in Fig. 2(b).

B. Block Diagram of a Reconfigurable FIR Chip

To illustrate the feasibility of the proposed reconfigurable FIR filter architecture, we designed a reconfigurable FIR chip with one pipeline stage for 8 DPUs. Detailed architecture for one pipeline stage is shown in Fig. 3 and it is referred to as a **processing element (PE)**. In order to reduce the area and the latency required for implementing the addition in the filtering process, eight adders are combined to form one single big adder with nine inputs. A special sign extension generator is introduced to deal with different precisions in the output of the DPU (15 bit) and the accumulated sum (24 bit). It generates the sum of sign extension bits of 8 DPUs' outputs based on the **sign** output of all **DPU**s. Finally, the **addend** output of 8 DPUs, the output of the sign extension generator, and the accumulated sum latched in the REG are summed together by an adder. From Fig. 1 and Fig. 3, we can see that the critical path is the path that goes through the DPU multiplier, the DPU shifter, and the adder generating the final sum. Filter configuration, however, affects only the delay in the data selecting path through DPUs, which will

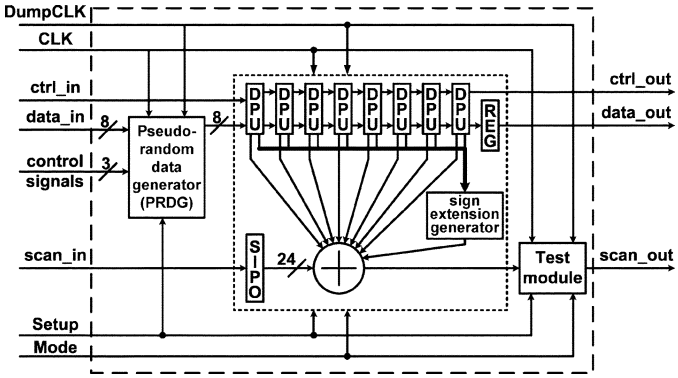

Fig. 4. Block diagram of the reconfigurable filter chip.

not along the critical path. Hence, the critical path is independent of the filter configuration. One PE is implemented in the chip so that it is capable of at most 8-digit FIR computation. It is also designed to be cascadable so that the FIR filter with more taps can be implemented.

The reconfigurable FIR chip consists of one PE, one pseudorandom data generator (PRDG), and one test module (see Fig. 4). There are two clock signals: **CLK** and **DumpCLK**. **CLK** controls the operating speed of the filter, while **DumpCLK** is used to initialize the chip parameters and to output the results. Based on the filter configuration, chip parameters in each DPU are pre-calculated and then serially shifted into the chip with the **ctrl_in** signal. Since the chip will be I/O limited, scan-chain is used for both testing and normal partial sum communication between cascaded chips. So, the REG in the PE that is used to store the accumulated sum is replaced by a 24-bit SIPO REG array. The **scan_in** signal is used to store the compensation vector for the first chip or the accumulated sum of the previous chip into a 24-bit SIPO serially in the case when several chips are cascaded. A PRDG is designed to provide the input test patterns for full-speed testing. The data fed into the first DPU can be just the **data_in** signal or from the PRDG. Finally, a test module accumulates the 24-bit output of the adder in PE with a 32-bit carry-save adder. The result can be serially scanned out at a slower clock rate to verify the functionality of the chip.

III. Circuit Design

A. Multiplier and Shifter

The multiplier is used to multiply the input data $x[n-i]$ by $d_{i,k}$, which has three possible values: 1, 0, and -1. If $d_{i,k}$ is "0," the **zero** signal is "1" and force the multiplier output to be 0 regardless of the input. Otherwise, the zero signal is "0" and if the CSD coefficient is "1," the **plus** signal is "1" and the multiplier output is the same as the input. If the CSD coefficient is "-1," the **plus** signal is "0" and the multiplier output is set to the one's complement of the input data. The "1" in least significant bit (LSB) that needs to be added to form the two's complement negation is accumulated for all negative coefficient digits. This sum forms the compensation vector and is added by setting the initial accumulating value of the accumulator.

A shifter is used to multiply the term $d_{i,k} \cdot x[n-i]$ by 2^{-p_k}, where $p_k \in \{0, \ldots, 7\}$. The shifter performs an arithmetic left shift and expands the 7-bit multiplier output (excluding the MSB) into a 14-bit output as the **addend** signal by shifting the input left by $7 - p_k$ bits. Also, zeros are padded at LSB if the CSD coefficient is "1" or "0" and ones are padded if the CDS coefficient is "-1."

TABLE II
Sum of Sign Extension Bits in Nine Cases

Number of the non-negative **sign** signals	Sum of sign extension bits	Number of the non-negative **sign** signals	Sum of sign extension bits
0	**1111111**000	5	**1111111**101
1	**1111111**001	6	**1111111**110
2	**1111111**010	7	**1111111**111
3	**1111111**011	8	**0000000**000
4	**1111111**100		

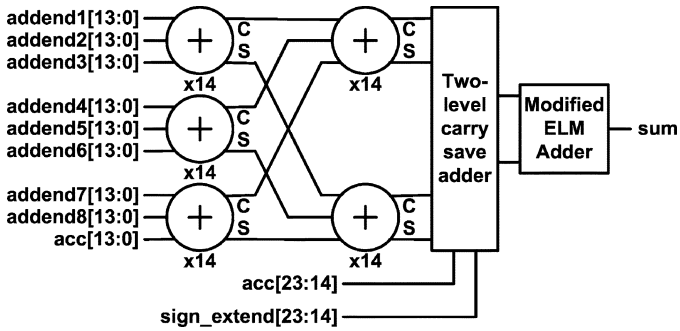

Fig. 5. Adder structure in the reconfigurable filter chip.

B. Sign Extension Generator

In our architecture, the accumulated sum is 24-bit wide but the term $d_{i,k} \cdot 2^{-p_k} \cdot x[n-i]$ calculated by the **DPU** is only 15-bit wide (14-bit **addend** and 1-bit **sign** signal). While summing them together, it is unwise to extend each of them to the word length of the accumulated sum. It is better to deal with the sign extension bits separately, so that the area and power consumption can be reduced.

A sign extension generator is designed to evaluate the sum of sign extension bits based on eight **sign** signals. The number of non-negative **sign** signals, i.e., "0" valued **sign** signals, and the sum of corresponding sign extension bits are shown in Table II. The boldfaced seven most significant bits can be determined by examining if there exists at least one "1" valued **sign** signal. It is also easy to see that the three LSBs are identical to the three LSBs of the binary representation of the number of non-negative **sign** signals. So, it can be implemented directly by complementing eight **sign** signals first, summing them together, and then taking the three LSBs of the sum.

C. Adder

The adder is used to sum eight 14-bit **addend** signals from DPUs, one 24-bit **acc** signal, and one 10-bit **sign_extend** signal from the sign extension generator. In Fig. 5, signals **addend1**$\sim$**addend8** correspond to eight **addend** signals and the signal **sign_extend** corresponds to the 10-bit signal from sign extension generator. The **acc** signal corresponds to the compensation vector or accumulated sum stored in the REG shown in Fig. 3. The **acc** signal is split into two parts where its fourteen LSBs and eight **addend** signals are compressed into four 14-bit signals by five 14-bit carry save adders in a

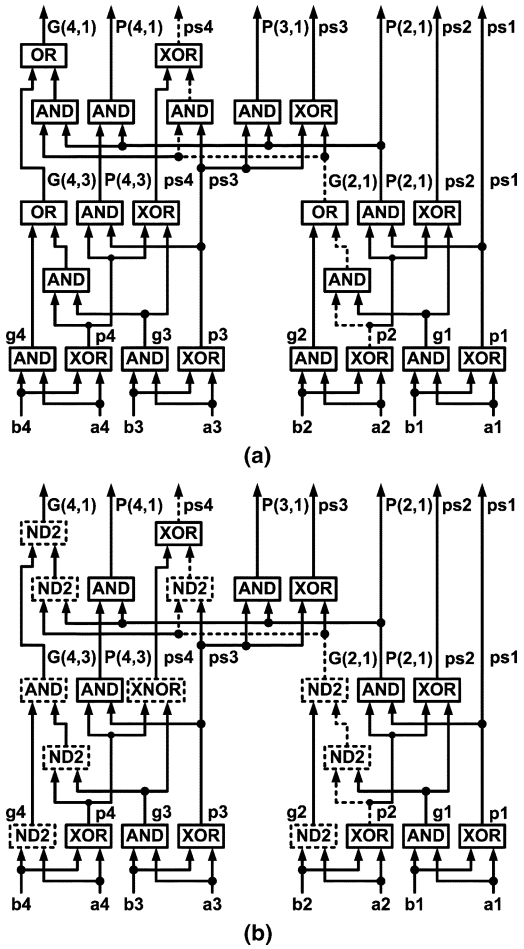

Fig. 6. Block diagrams of: (a) 4-bit ELM adder and (b) modified 4-bit ELM adder.

two-level arrangement. Its ten MSBs are then added with the **sign_extend** signal and above four 14-bit signals by a two-level carry save adder. Finally, a modified ELM adder with reduced critical path delay is used to compute the final sum.

The ELM adder is a kind of parallel adder first presented in [13]. Fig. 6(a) illustrates a 4-bit ELM adder and the critical path is indicated in dashed lines. First, we can replace the OR gate and its two input AND gates by three NAND gates as shown in Fig. 6(b). Also, as can be seen in Fig. 6(a), the critical path includes an XOR gate and an AND gate. We can further reduce the critical path delay by inverting the two inputs of the XOR gate as shown in Fig. 6(b). These modifications reduce not only the critical path delay but also the transistor count and thus give a faster and smaller 4-bit parallel adder.

IV. Implementation and Measurement

Detailed circuit design of the reconfigurable FIR filter chip was described in gate level using the hardware description language, Verilog. Layout was generated through the standard-cell-based design flow with our own cell library. The final layout is approximately 1.74 mm × 1.64 mm in a single-poly quadruple-metal 0.35-μm CMOS technology and contains 27035 transistors. The die photo of the fabricated reconfigurable FIR filter is shown in Fig. 7. There are 8 DPUs arranged in one row and three

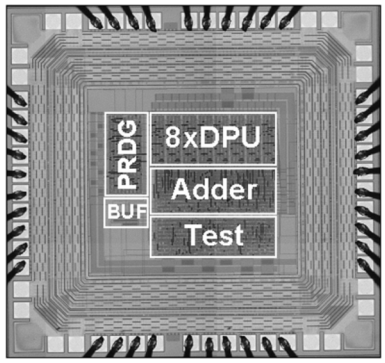

Fig. 7. Die photograph of the proposed chip.

TABLE III
Summary of the Reconfigurable FIR Filter Chip

Clock Frequency	86 MHz
Power Supply	2.5 V
Power Dissipation	16.5 mW (filter only)
Process Technology	0.35-μm 1P4M CMOS
Transistor Count	27035
Die Size	1.74 mm x 1.64 mm

other blocks: an adder, a PRDG, and a test module. The clock buffer and the reset buffer reside in the bottom-left corner.

The fabricated chip has been tested and its function has been verified. The reconfigurable FIR filter chip can operate correctly at 86 MHz under a supply voltage of 2.5 V and the filter part of the chip consumes 16.5 mW at that speed. Table III summarizes the major features of the reconfigurable FIR filter chip.

V. Discussion

Based on FIR design results in [12], the number of signed powers-of-two (SPT) terms required for the proposed architecture is about 70% of that required by the architectures proposed in [7]–[9] when realizing an FIR filter with similar performance. Albeit 15% hardware overhead required for each SPT term to provide the reconfigurability, the proposed architecture still achieves 20% hardware savings given comparable speed performance. The architecture (transposed form) proposed in [10] can achieve higher speed but require higher complexity. Moreover, it results in longer critical path when there are more SPT terms in each tap. On the contrary, the critical path of the proposed architecture is independent of the filter configuration.

The measured power consumption of the reconfigurable FIR filter was compared to several previous filter implementations in the literature [14]–[16]. An equation introduced in [14] was used to normalize the performance of several FIR filters to the reconfigurable FIR filter chip by taking into account the number and the precision of multipliers used, the technology, the power supply voltage, and the operating frequency and is given by

$$P(\text{mult}) = \frac{\text{Total Power}}{\#\text{ mults}} \times \frac{8}{\#\text{ bits coeff}} \times \frac{8}{\#\text{ bits sample}} \times \left(\frac{2.5}{V_{dd}}\right)^2 \times \frac{0.35}{\text{Tech}} \times \frac{86}{\text{clk freq}}. \quad (3)$$

TABLE IV
COMPARISON OF PROPOSED CHIP WITH PREVIOUS PROGRAMMABLE FIR FILTERS

	Filter Description	Tech (μm)	Vdd (V)	Power (mW)	P(mult) (mW)
[14]	128-tap, 32 10×12 mult@50MHz	0.5	3.3	162	1.9
[15]	288-tap, 72 10×10 mult@72MHz	0.6	3.3	800	2.8
[16]	8-tap, 8 6×6 mult@550MHz	0.18	1.8	36	4.7
this work	8-digit, 2.88 8×8 mult@86MHz	0.35	2.5	16.5	5.7
this work (postsim)	32-digit, 11.52 12×8 mult@100MHz	0.35	2.5	80	4.0

This formula assumes linear scaling and favors older technologies due to the wire fringing field capacitance. The results of this rough comparison are listed in Table IV. The corresponding precision of multipliers in this filter is 8×8. The number of multiplications, # mults, of this work, 2.88, is derived by dividing 8 (the number of DPUs in the chip) by the average number of nonzero CSDs required to represent an 8-bit wide coefficient.

As shown in Table IV, the reconfigurable FIR filter chip consumes comparable power as [16] but three times as much as [14] and twice as much as [15]. The difference is partially due to the fact that (3) favors older technology as explained in [14]. Moreover, the normalized power of [14] is greatly reduced by using carry-save representation in the output, which can be easily adapted to the proposed architecture to reduce its power consumption. To make a fair comparison, we designed a 32-DPU circuit using the proposed architecture and the carry-save representation for outputs. The precision of the multiplier in the DPU is 12×8. A layout was generated through the same flow using the same library which has been used in the design of the reported chip. Post-layout simulation results show that its normalized power consumption is only 70% of that of the 8-DPU filter. So, if we implement an FIR filter based on the proposed architecture with 32 DPUs or more and utilize the carry-save representation in the outputs, it can achieve comparable normalized power consumption as the three filters in [14]–[16] but with a degree of flexibility unachievable by them. In conclusion, the proposed digit-based reconfigurable FIR architecture provides a flexible yet compact and low-power solution to FIR filters with a wide range of precision and tap length, often needed in most communication signal processing systems. For example, the filter implemented with the proposed architecture can be easily configured as a 128-chip binary PN code matched filter or a 16-tap raised cosine filter with 16-bit coefficient precision if 128 DPUs are available.

VI. CONCLUSION

In this brief, a reconfigurable FIR filter architecture is proposed. The design concepts of the architecture and the circuit are presented. Testing results show that the fabricated reconfigurable filter draws only 16.5 mW from a 2.5-V power supply while running at 86 MHz.

ACKNOWLEDGMENT

The authors greatly appreciate the Chip Implementation Center (CIC) of the National Science Council, Taiwan, R.O.C., for the fabrication and measurement of the proposed chip.

REFERENCES

[1] J. Mitola, "The software radio architecture," *IEEE Commun. Mag.*, vol. 33, pp. 26–38, May 1995.
[2] M. Isohookana, T. Kokkonen, P. Leppanen, A. Nykanen, J. Pyhtila, J. Reinila, J. Sillanpaa, and V. Tapio, "Software radio-an alternative for the future in wireless personal and multimedia communications," in *Proc. IEEE Int. Conf. Personal Wireless Commun.*, 1999, pp. 364–368.
[3] E. Buracchini, "The software radio concept," *IEEE Commun. Mag.*, vol. 38, pp. 138–143, Sep. 2000.
[4] R. M. Hewlitt and E. S. Swartzlantler, Jr., "Canonical signed digit representation for FIR digital filters," in *Proc. IEEE Worksh. Signal Process. Syst.*, 2000, pp. 416–426.
[5] M. Tamada and A. Nishihara, "High-speed FIR digital filter with CSD coefficients implemented on FPGA," in *Proc. ASP DAC'01*, 2001, pp. 7–8.
[6] Y. M. Hasan, L. J. Karem, M. Falkinburg, A. Helwig, and M. Ronning, "Canonic signed digit Chebyshev FIR filter design," *IEEE Signal Process. Lett.*, vol. 8, no. 6, pp. 167–169, Jun. 2001.
[7] T. Zhangwen, Z. Zhanpeng, Z. Jie, and M. Hao, "A high-speed, programmable, CSD coefficient FIR filter," in *Proc. 4th Int. Conf. ASIC*, 2001, pp. 397–400.
[8] K. T. Hong, S. D. Yi, and K. M. Chung, "A high-speed programmable FIR digital filter using switching arrays," in *Proc. IEEE Asia Pacific Conf. Circuits Syst.*, 1996, pp. 492–495.
[9] T. Rissa, R. Uusikartano, and J. Niittylahti, "Adaptive FIR filter architectures for run-time reconfigurable FPGAs," in *Proc. 2002 IEEE Int. Conf. Field Programm. Technol.*, Dec. 2002, pp. 52–59.
[10] K. Y. Khoo, A. Kwentus, and A. N. Willson, Jr., "A programmable FIR digital filter using CSD coefficients," *IEEE J. Solid-State Circuits*, vol. 31, no. 6, pp. 869–874, Jun. 1996.
[11] D. Li, Y. C. Lim, Y. Lian, and J. Song, "A polynomial-time algorithm for designing FIR filters with power-of-two coefficients," *IEEE Trans. Signal Process.*, vol. 50, no. 8, pp. 1935–1941, Aug. 2002.
[12] C. L. Chen and A. N. Wilson, Jr., "A trellis search algorithm for the design of FIR filters with signed-powers-of-two coefficients," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 46, no. 1, pp. 29–39, Jan. 1999.
[13] T. P. Kelliher, R. M. Owens, M. J. Irwin, and T. -T. Hwang, "ELM-A fast addition algorithm discovered by a program," *IEEE Trans. Comput.*, vol. 41, no. 9, pp. 1181–1184, Sep. 1992.
[14] C. J. Nicol, P. Larsson, K. Azadet, and J. H. O'Neill, "A low power 128-tap digital adaptive equalizer for broadband modems," *IEEE J. Solid-State Circuits*, vol. 32, no. 11, pp. 1777–1789, Nov. 1997.
[15] J. R. Choi, L. H. Jang, S. W. Jung, and J. H. Choi, "Structured design of a 288-tap FIR filter by optimized partial product tree compression," *IEEE J. Solid-State Circuits*, vol. 32, no. 3, pp. 468–476, Mar. 1997.
[16] R. B. Staszewski, K. Muhammad, and P. Balsara, "A 550 Msample/s 8-Tap FIR digital filter for magnetic recording read channel," *IEEE J. Solid-State Circuits*, vol. 35, pt. 8, pp. 1205–1210, Aug. 2000.