A LOW COMPLEXITY AND LOW POWER SOC DESIGN ARCHITECTURE FOR ADAPTIVE MAI SUPPRESSION IN CDMA SYSTEMS

Yuanbin Guo[†] Joseph R. Cavallaro

Rice University
Department of Electrical and Computer Engineering
Houston TX 77005
{Yuanbin.Guo@nokia.com, cavallar@rice.edu}

Abstract

In this paper, we propose a reduced complexity and power efficient System-on-Chip (SoC) architecture for adaptive interference suppression in CDMA systems. The adaptive Parallel-Residue-Compensation architecture leads to significant performance gain over the conventional interference cancellation algorithms. The multi-code commonality is explored to avoid the direct Interference Cancellation (IC), which reduces the IC complexity from $\mathcal{O}(K^2N)$ to $\mathcal{O}(KN)$. The physical meaning of the *complete* versus *weighted* IC is applied to clip the weights above a certain threshold so as to reduce the VLSI circuit activity rate. Novel scalable SoC architectures based on simple combinational logic are proposed to eliminate dedicated multipliers with at least $10\times$ saving in hardware resource. A Catapult C High Level Synthesis methodology is apply to explore the VLSI design space extensively and achieve at least $4\times$ speedup. Multi-stage Convergence-Masking-Vector combined with clock gating is proposed to reduce the VLSI dynamic power consumption by up to 90%.

Keywords : interference cancellation, low power, CDMA, adaptive, SoC, VLSI.
This paper was presented in part at IEEE ISCAS in Vancouver, Canada, May, 2004.

[†]Author to whom all correspondence should be addressed, currently with Nokia Research Center, 6000 Connections Dr., Irving, TX, 75039; Tel: 469-767-5681.

I. Introduction

Multiple Access Interference (MAI) is one of the major limiting factors to system capacity in CDMA systems. In [1] [2], a Parallel Interference Cancellation (PIC) algorithm was developed to detect the multiple users simultaneously by *completely* cancelling the estimated MAI from all remaining users. Since it is much simpler than the Maximum-Likelihood (ML) multi-user detector, it has been well accepted as one of the most practical algorithms for real-time implementation. A multi-stage real-time VLSI architecture based on this algorithm was reported in [3]. Related implementation schemes are found in [4] and [5]. However, when the interference estimation is not accurate (e.g., when the system load is high or the receiver is in the early detection stages), cancelling the wrong estimate may even add more interference to the signal. This leads to the so-called "ping-pong" effect in the conventional PIC algorithms. In such situations, it is preferable not to cancel the entire estimated interference. Divsalar et al. [6] proposed a *partial* PIC (PPIC) algorithm by introducing a weight in each stage. The stage specific weights are found by a trial-and-error computer search for all the users with the only intuitive constraint $0 < w_1 < w_2 < \cdots < w_m < 1$, where w_m is the weight at stage m. A DSP prototype based on this improved algorithm was reported in [7]. However, because of the limited parallelism, the DSP-based prototype supported only a small number of users for relatively low speed systems.

The intuitive weights applied in [6] are far from optimal solution because it does not apply any optimization criteria in finding the weights. To seek better weights, adaptive PIC based on Minimum Mean Squared Error (MMSE) criteria was proposed in [8]. The weight for each user in each stage is computed by an adaptive Normalized Least-Mean-Square (NLMS) algorithm. However, the NLMS algorithm increases the system complexity considerably, which makes the real-time implementation very challenging. Although some VLSI architectures for DS/CDMA receiver can be found in [11], [12] and [13], related VLSI architecture work for the adaptive interference cancellation has not been reported yet in the literature. The extra complexity demands special treatments to meet the real-time requirement and hardware resource limit.

To achieve the goal of real-time implementation with efficient architecture, both algorithmic and architectural optimizations are explored in this paper. First, a symbol level Parallel-Residue-

Compensation (PRC) architecture is proposed to avoid the direct interference estimation for each individual user. The PRC reduces the complexity of the adaptive interference cancellation from $\mathcal{O}(K^2N)$ to $\mathcal{O}(KN)$ without loss in performance by utilizing the commonality among the multiple codes. The algorithm is further optimized extensively to reduce the redundant computations, avoid timing conflicts and share hardware resources for efficient real-time implementation.

Power consumption is an essential consideration for both VLSI and DSP processor implementations, especially for mobile devices. SoC design architecture has many advantages over general-purpose DSP processors by providing higher parallelism and pipelining, lower power consumption and compacter size. Algorithmic transformations like pipelining and parallel processing can be used to reduce power consumption. On the other hand, shutting down some computation blocks leads to fewer instructions in DSP and fewer cycles in VLSI implementation. The power savings achieved in this manner can be significant but are very algorithm dependent. A proper addressing of *when* and *how* to shut down can result in substantial improvement in energy efficiency with no or little loss in performance.

To design power saving schemes for the SoC architecture, we explore the physical meaning of the weights in the adaptive interference cancellation algorithm. A conventional complete PIC is considered as a special case of adaptive scheme with weight '1' for all users. It is found that when the interference weight for one particular user in one stage is '1', it implicates that the symbol of the particular user is estimated *"almost"* correctly and the interference from that user is *"completely"* cancelled. We then investigate the inter-stage features of the user-specific weights. In the early stages, the NLMS algorithm will adjust the weights more significantly since the symbol detection is less accurate than later stages. But the weight tends to converge to '1' in later stages as the MSE converges. After the first stage, only a small portion of weights will diverge from the initial values. Thus, the distance of one user's weight from the initial value is used as an indicator of the accuracy of the symbol detection of that user. A Convergence-Masking-Vector (CMV) is generated by comparing each user's weight with a given threshold at each stage. The vector only contains flags (0 or 1) to indicate if the weight has converged or not.

The CMV is combined with clock gating as a dynamic power management function for the multi-stage components of VLSI architectures. If the CMV indicates a convergence, then there is

no need to update the weight for this user at all later stages and the corresponding components in later stages are shut down. Simulation shows that the active rate can be dropped to 60% after stage 1 and to 10% after stage 2 with a threshold of 90%, which leads to negligible performance loss. This gives 40% dynamic power savings in stage 1 and 90% savings after stage 2.

There exist many area/time tradeoffs in the SoC architecture [12]. The conventional VHDL-based design methodology is very time consuming and difficult to explore the design space extensively [15]. In this paper, a Catapult C High Level Synthesis (HLS) design methodology is proposed to explore design space extensively using layered parallelism and pipelining [9]. Special bit-ware VLSI units based on the simple Sumsub-Mux Unit (SMU) for the bottleneck design blocks are proposed to eliminate the use of dedicated ASIC multipliers. This reduces the hardware complexity dramatically and achieve the efficient tradeoffs in parallelism and pipelining architectures. The most area/time efficient VLSI architectures are implemented in an Field Programmable Gate Array (FPGA) prototyping system, giving at least $10\times$ saving in the hardware resource over a multiplier based design and at least $4\times$ speedup over the conventional area-constraint architecture.

The paper is organized as follows. In section II, we present the system model and conventional receivers. In section III, the symbol level adaptive PRC is presented followed by the power efficient CMV architecture. Section IV describes the SoC architecture design methodology, system partitioning and the derivation of the SMU combinational logic. The VLSI architecture for the dominant weight updating and PRC modules based on SMU blocks are presented in section V. In section VI, we provide the VLSI design space exploration results and the fixed-point bit error rate performance from emulations. The paper concludes in section VII.

II. System Model and Conventional Receiver

We consider the synchronous multi-code CDMA system using QPSK modulation scheme. The n^{th} symbol for the k^{th} user at the transmitter is mapped to constellation points using a group of bits $\{b_k^0, b_k^1\} \in \{0, 1\}$. The symbol output at the modulator is $s_k^{(n)} = \{[-2b_k^0(n) + 1] + [-2b_k^1(n) + 1]j\}/\sqrt{2}$ with equal probability, where $j = \sqrt{-1}$. In an AWGN channel, the received complex

base band signal at the i^{th} chip of the n^{th} symbol is expressed as

$$r^{(n)}(i) = \sum_{k=1}^{K} \alpha_k^{(n)} \sqrt{P_k^{(n)}} s_k^{(n)} c_k[i + (n-1)N] + z(i) \tag{1}$$

where $\alpha_k^{(n)}$ and $P_k^{(n)}$ are the complex channel amplitude and transmitted power for the k^{th} user, respectively. $c_k[i + (n-1)N] \in \{\pm 1\}$ is the i^{th} chip spreading code of the n^{th} symbol for the k^{th} user. N is the spreading factor and $K \in [1, N]$ is the number of active users. $z(i)$ is the complex additive Gaussian noise with double-sided spectrum density $\mathcal{N}_0/2$.

We focus on the n^{th} symbol and omit the symbol index for notation simplicity in the following. By collecting the N chip samples in one symbol duration into a vector, we form a signal vector as $\mathbf{r} = [r(0) \quad r(1) \quad \cdots \quad r(N-1)]$. A matched filter can be used to despread the received signal and to generate the soft detection of multiple users' symbols as $\tilde{\mathbf{S}}_{MF0} = \mathbf{r}\mathbf{C}^H/N$, where $\mathbf{C}$ is the spreading code matrix for all the users. $\mathbf{R} = [\mathbf{C} * \mathbf{C}^H]$ is the cross correlation matrix of the spreading codes where the superscript H denotes Hermitian conjugate. MAI appears when the cross correlation is not identity. The elements of $\tilde{\mathbf{S}}_{MF0}$, i.e., the k^{th} user's symbol detection is given by

$$\tilde{s}_k = \alpha_k \sqrt{P_k} s_k + \frac{1}{N} \sum_{\substack{j=1, \\ j \neq k}}^{K} \sum_{i=0}^{N-1} \alpha_j \sqrt{P_j} s_j c_j(i) c_k^*(i) + \frac{1}{N} \sum_{i=0}^{N-1} z(i) c_k^*(i). \tag{2}$$

The matched filter output is then corrected by the channel estimation phase and sent to a multi-user demodulator. At the demodulator, the estimated bits of the k^{th} user are detected as $\hat{b}_k^0 = \mathrm{sgn}\{\Re(\tilde{s}_k \cdot / \alpha_k \sqrt{P_k})\}$ and $\hat{b}_k^1 = \mathrm{sgn}\{\Im(\tilde{s}_k \cdot / \alpha_k \sqrt{P_k})\}$.

A. Complete Multistage PIC

A multi-user Parallel-Interference-Cancellation (PIC) algorithm was proposed in [1] for simultaneous detection of all users. By assuming the bit estimation of the $(m-1)^{th}$ stage as the transmitted bits for each user, it estimates the interference at the m^{th} stage for each user by recon-

structing the transmitted signal excluding itself as in

$$\hat{I}_k^{(m)}(i) = \frac{1}{N} \sum_{\substack{j=1 \\ j \neq k}}^{K} \sum_{i=0}^{N-1} \hat{\alpha}_j \hat{s}_j^{(m-1)} c_j(i) \tag{3}$$

where $\hat{s}_j^{(m-1)}$ is the modulator symbol output for user j at the $(m-1)^{th}$ stage by using the hard-decision bits of the $(m-1)^{th}$ stage. The estimated interference is subtracted completely from the received signal for each user. The corrected signal is despread and demodulated as (4) to generate more accurate estimation of the bits. This process is repeated in an iterative pattern for multiple stages.

$$\tilde{s}_k^{(m)} = \frac{1}{N} \sum_{i=0}^{N-1} [r(i) - \hat{I}_k^{(m)}(i)] c_k^*(i) \tag{4}$$

B. Partial PIC Receiver

It is pointed out in [6] that if the estimation of the early stages is not accurate enough, the complete PIC even adds more interference to the signal. To achieve more accurate interference cancellation, a partial weight is introduced for each stage. The weights are chosen based on the intuition that the estimation from the earlier stages is less accurate than later stages and less interference should be cancelled. The intuitive weights are found by a trial-and-error computer search with the only constraint that $w_1 < w_2 < \cdots < w_m$. The more accurate signal used for demodulation of each user is generated by adding the partially interference cancelled signal and a weighted soft input signal of the previous stage as in (5).

$$\tilde{r}_k^{(m)} = w^{(m)} [r(i) - \hat{I}_k^{(m)}(i)] + [1 - w^{(m)}] \tilde{s}_k^{(m-1)} c_k(i) \tag{5}$$

$$\tilde{s}_k^{(m)} = \frac{1}{N} \sum_{i=0}^{N-1} \tilde{r}_k^{(m)}(i) c_k^*(i) \tag{6}$$

III. Low Power Adaptive PRC

Despite the performance gain of PPIC over the complete PIC, the intuitive weighting scheme is far from optimal solution. For better accuracy, it is preferable to choose individual weights for

each user depending on the accuracy of the symbol detection. To achieve this, a set of weights is introduced in [8] for each user in each stage. By defining a cost function in terms of the squared Euclidean distance between the received signal $r(i)$ and the weighted sum of all users' estimated signal, the optimal weights are given by minimizing the MSE of the cost function as $\mathbf{w}_{opt}^{(m)} = \arg\min_{w_k^{(m)}} E[|r(i) - \hat{r}_w^{(m)}(i)|^2]$ where the weighted sum of all users' hard-decision symbols at the m^{th} stage is given by

$$\hat{r}_w^{(m)}(i) = \sum_{k=1}^{K} w_k^{(m)}[c_k(i)\hat{s}_k^{(m-1)}] = \mathbf{w}^{(m)}\hat{\mathbf{\Omega}}^{(m-1)}(i). \tag{7}$$

Here $\mathbf{w}^{(m)} = [w_1^{(m)} \quad w_2^{(m)} \quad \cdots \quad w_K^{(m)}]$ is the weighting vector for the m^{th} stage and $\hat{\mathbf{\Omega}}^{(m-1)}(i) = [c_1(i)\hat{s}_1^{(m-1)} \quad c_2(i)\hat{s}_2^{(m-1)} \quad \cdots \quad c_K(i)\hat{s}_K^{(m-1)}]^T$is the output vector of the multi-user spreader in the signal regenerator of the PIC. Define the residual error between the desired response and its estimate in the m^{th} stage as $\epsilon^{(m)}(i) = r(i) - \hat{r}_w^{(m)}(i)$, the MMSE optimization is solved by the Normalized Least-Mean-Square (NLMS) algorithm in an iterative update equation operated in the bit interval on chip rate as

$$\begin{aligned} &\mathbf{w}^{(m)}(i+1) = \mathbf{w}^{(m)}(i) + \frac{\mu}{\| \hat{\mathbf{\Omega}}^{(m)}(i) \|^2}[\hat{\mathbf{\Omega}}^{(m-1)}(i)]^*\epsilon^{(m)}(i)] \\ &\mathbf{w}_{opt}^{(m)} = \mathbf{w}^{(m)}(N-1) \end{aligned} \tag{8}$$

where μ is the step size and $\hat{\mathbf{\Omega}}^{(m-1)}$ is the input vector to the NLMS algorithm. The interference for each user in the adaptive PIC is estimated in a direct form for all the K users as $\hat{I}_k^{(m)}(i) = \sum_{\substack{j=1 \\ j\neq k}}^{K} w_j^{(m)}(N-1)[c_j(i)\hat{s}_j^{(m-1)}]$. The more accurate chip-level signal is generated for each user as $\tilde{\gamma}_k^{(m)}(i) = r(i) - \hat{I}_k^{(m)}(i)$ and the more accurate symbols are detected as $\tilde{s}_k^{(m)} = \frac{1}{N}\sum_{i=0}^{N-1} \tilde{\gamma}_k^{(m)}(i)c_k^*(i)$.

A. Adaptive Parallel Residue Compensation

Since the computational complexity determines the cost of necessary hardware resources such as the number of functional units, it is one of the most important considerations in the implementation of PIC schemes. The complexity of direct form PIC in one chip for K users is $4K(K-1)$ real multiplications, $2K(K-1)$ real additions and $2K$ subtractions. Moreover, there is one"if" state-

ment which is mapped to a hardware comparator for each user loop. This makes the loop structure irregular and not very suitable for pipelining. Considering the regularity of the computations for all users, we change the order of *"interference estimation"* and *"interference cancellation"*. Instead, the new architecture has the following steps:

1. "Weighted-Sum-Chip Function": by summing up all users' weighted signal together, we get weighted estimation of the received signal in chip rate samples as

$$\hat{r}_{w,opt}^{(m)}(i) = \sum_{k=1}^{K} w_k^{(m)}(N-1)[c_k(i)\hat{s}_k^{(m-1)}]. \tag{9}$$

2. "Residue Error Generation": a common residual signal for all users is generated by a single subtraction from the original signal as

$$\epsilon^{(m)}(i) = r(i) - \hat{r}_{w,opt}^{(m)}(i). \tag{10}$$

3. "Parallel Residual Computation": in the final step, this residual error is compensated to each user to get the interference-cancelled chip signal as

$$\tilde{\gamma}_k^{(m)} = \epsilon^{(m)}(i) + w_k^{(m)}(N-1)[c_k(i)\hat{s}_k^{(m-1)}]. \tag{11}$$

4. The afore-mentioned procedure constructs a "Chip-Level" PRC (CL-PRC) structure if the multi-user "chip-matched filter" is carried out on the corrected chip signals directly as described. However, by jointly considering the matched filter and the residue compensation step in (9), (10) and (11), the 0^{th} stage multi-user matched filter output can be utilized to generate the "symbol-Level" PRC (SL-PRC) architecture. The "spreading" and then "matched filter" procedure for the weighted symbols of each user is redundant in chip level. We only need to do matched filtering for the weighted-sum chips as in (12). The soft-decision matched filter output of the corrected signal is finally generated in the symbol level as (13). The optimally weighted symbol in (9) can be computed as $w_s(k) = w_k^{(m)}(N-1)\hat{s}_k^{(m-1)}$ and stored in registers or memory arrays before the spreading in (9).

$$\hat{\xi}_{w,MF}(k) = \frac{1}{N}\sum_{i=0}^{N-1} \hat{r}_{w,opt}^{(m)}(i)c_k^*(i). \tag{12}$$

$$\tilde{s}_k^{(m)} = \tilde{\mathbf{S}}_{MF0}(k) - \hat{\xi}_{w,opt}(k) + w_s(k). \tag{13}$$

The complexities of one stage PRC and matched filters for the three different schemes are summarized here. It is seen that the interference cancellation complexity is reduced from the order of $\mathcal{O}(K^2 * N)$ in direct interference cancellation to $\mathcal{O}(K * N)$ in the proposed PRC architecture, which is linear to the number of users. Overall, the symbol-level PRC has the minimum complexity. Although it is similar to the chip-level PRC, the loop chain for chip index is more compact and regular for scheduling the pipelined and parallel architecture than the chip level processing so as to generate faster VLSI design.

B. Algorithmic Optimization for Low Power VLSI Architecture

Low power consumption is an important factor in lowering system cost [17]-[22]. Average power determines the battery life while peak power affects the reliability. The source of power consumptions in CMOS technology includes the switching current (dynamic power), short circuit current and leakage currents. The average power consumption of a CMOS gate due to the switching component is given by $P = \alpha_s C_L \cdot V_{dd}^2 \cdot f_{clk}$, where f_{clk} is the system clock frequency, V_{dd} is the supply voltage, C_L is the load capacitance and α_s is the switching activity (the probability of $0 \rightarrow 1$ transition during a clock-cycle). Power saving can be accomplished either by shutting off the clock $(f_{clk} = 0)$ or in certain cases by shutting off the power supply $(V_{dd} = 0)$ [17]. To save power, the circuit is partitioned into small blocks that have their own derived clocks. Power savings are achieved by stopping the clock fed into the idle modules. The power savings achieved in this manner can be significant but are very algorithm dependent. A proper addressing of when to shutdown and how to scale the voltages can result in substantial improvement in energy efficiency with no or little loss in performance.

The power equation suggests many strategies for increasing the energy efficiency at various abstraction levels: from the algorithmic level down to the layout level. Thus, the working space for a joint algorithm and SoC architecture optimization is depicted in Fig. 1. On the algorithm side, the focus is the system performance in terms of bit error rate etc. On the architecture side, the focus is the VLSI performance in terms of the real-time cycle number, silicon area and dynamic power

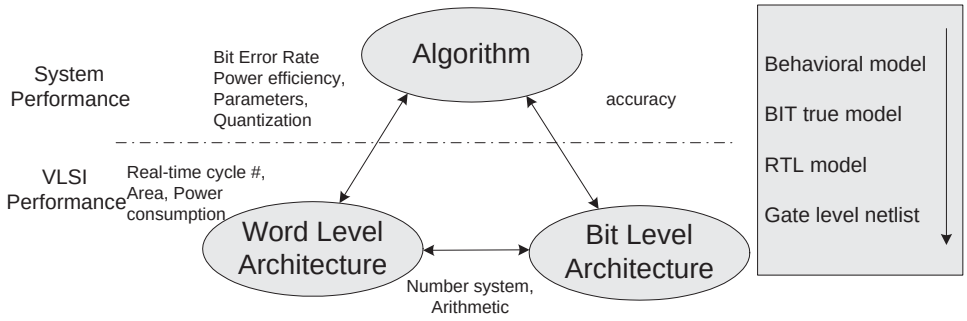

Figure 1: Working space for joint algorithm/SoC architecture optimization.

dissipation. To convert an algorithm to the real-time architecture, we work at different levels: from floating point algorithm to the behavioral model, the bit-true model, the RTL model and the gate level netlist. To achieve the power saving, we propose a CMV based on the algorithmic feature of the weights to provide a control logic signal to stop the multi-stage NLMS adaptation. We can employ either a central or a set of distributed control units. It is more favorable to use a set of distributed control units since long and fast control signals are eliminated. These distributed control units also become much simpler and faster and consume less power. The control unit is applied to shut down the system and enter into an idle state.

C. Stochastic Convergence Masking Vector

Because of the MMSE criteria in the adaptive PRC scheme, the mean squared error will converge in the NLMS update recursion. If the weight for the m^{th} stage is very close to the initial value of the NLMS algorithm, it means that the MSE has converged and the interference from the particular user has been cancelled out at the m^{th} stage, Thus, there is no need to continue the weight update and interference cancellation in the later stages.

To analyze the stochastic feature of the user-specific weights, the normalized optimal weights versus chip index for stages $1, 2$ and 3 are depicted Fig. 2. It is observed that the weights for some symbols converge to the initial values (a normalized weight of 1), which indicates that those symbols are detected almost correctly and at this stage the interference from this user is completely cancelled by re-generating the signal with the initial weights. Moreover, if a particular weight

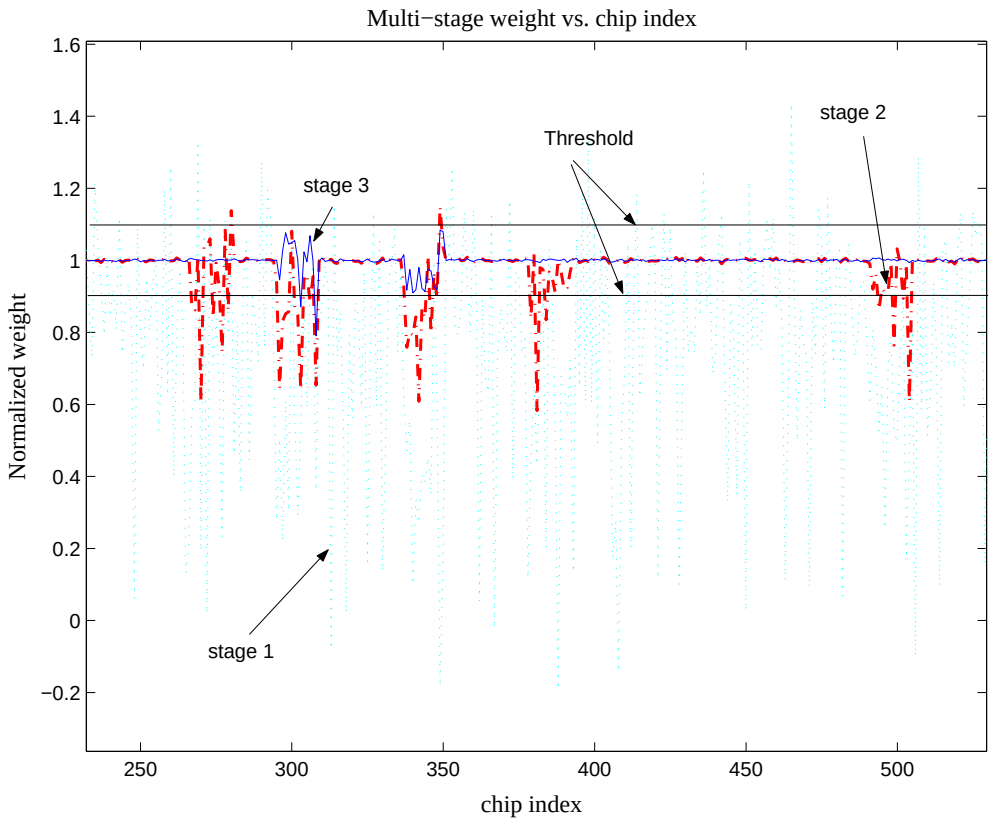

Figure 2: Inter stage feature of the normalized weights in adaptive PRC.

converges for a group of chips at the $(m-1)^{th}$ stage, the weight for this user and symbol tends to converge also at the m^{th} and later stages. This makes sense because the weight in the adaptive NLMS PRC is user, symbol and stage specific. The convergence of the weight depends on the confidence level of the correctness in symbol detection. If the symbol is already detected correctly, then a normalized weight "1" can cancel the interference from this user "*completely*". There is no need to continue the cancellation for the particular user symbol in later stages.

With more stages, more weights converge to the normalized "1" (in the case of BPSK, many of them converge in the second stage). Thus, after more interference is cancelled and the signal is getting cleaner in later stages, the majority of the weights will be close to the normalized weight "1". This is demonstrated by the probability-distribution-function (PDF) for stages $1, 2, 3$ and 4 in Fig. 3. For the later stages, only a small portion of both the weights and the interference need to be updated and cancelled, respectively. It also gives a metric to control the hardware utilization in

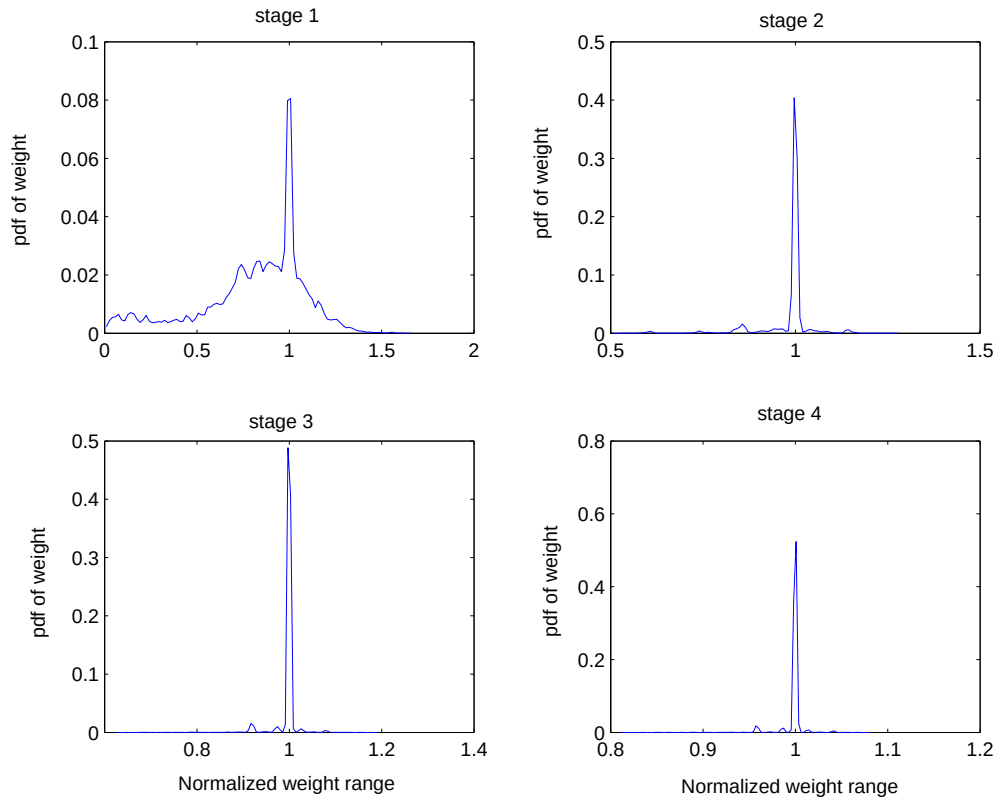

Figure 3: Probability Distribution Function of the normalized weights in adaptive PRC.

multi-stage architectures. Dynamic power management then can be applied for the design of the NLMS and PRC blocks.

Specifically, we can construct a CMV for the SoC architecture with pipelined multi-stage layout processing units to provide control logic to stop the multi-stage NLMS adaptation and the PRC sections. The CMV provides a stochastic clock gating scheme to save dynamic power for each stage. The procedure is summarized briefly in the following. First, we detect the $\hat{\mathbf{\Omega}}^{(0)}(i)$ from the 0^{th} stage MF. Then at the 1^{st} stage, we initialize $\mathbf{w}^{(1)}(0) = \hat{\alpha}$ and update the weights according to the NLMS recursion in (8). For each user, we set the CMV vector $V_k^{(1)}$ for $k \in [1, K]$ using a threshold T_{sh} as

$$\begin{cases} V_k^{(1)} = 1 & |w_{opt,k}^{(1)} - \hat{\alpha}_k|/|\hat{\alpha}_k| < 1 - T_{sh} \\ V_k^{(1)} = 0 & o.w. \end{cases} \tag{14}$$

where $V_k^{(1)} = 1$ means that the k^{th} user has converged to a "correct" symbol detection, there is no

need to continue the detection of this symbol for this user in later stages. Else $V_k^{(1)} = 0$ indicates that the receiver needs to continue the multistage weight update for the k^{th} user. We also separate the weighted-sum chip signal in (9) into two terms: the converged term $\hat{r}_V^{(1)}$ and the not-converged term $\hat{r}_w^{(1)}$ as

$$\hat{r}_{w,opt}^{(1)} = \underbrace{\sum_{\substack{k=1,\\ V_k^{(1)}=1}}^{K} \hat{\alpha}_k[c_k(i)s_k^{(0)}]}_{\hat{r}_V^{(1)}} + \underbrace{\sum_{\substack{k=1,\\ V_k^{(1)}=0}}^{K} w_{opt,k}^{(1)}[c_k(i)s_k^{(0)}]}_{\hat{r}_w^{(1)}(i)}. \tag{15}$$

Then the symbols $\hat{\mathbf{\Omega}}^{(0)}(i)$ are detected from (12) and (13) with the interference cancelled. For the $m^{th}(m > 1)$ stage, if $V_k^{(m-1)} = 1$, then $V_k^{(m)} = 1$. There is no need to detect these symbols at later stages. Otherwise, we initialize and update the weight from the NLMS update equation (8). We compute $w_{opt,k}^{(m)} = w_k^{(m)}(N-1)$ only for the users that have not converged, i.e., for those $V_k^{(m-1)} = 0$. For the newly updated weights, we compare them with the threshold again. If $|w_{opt,k}^{(1)} - \hat{\alpha}_k|/|\hat{\alpha}_k| < 1 - T_{sh}$, then the weight for the user k has already converged and we can set $V_k^{(m)} = 1$. There is also no need to detect symbol $\hat{s}_k^{(m)}$ at later stages and we set $\hat{\mathbf{\Omega}}^{(m)}(i) = \hat{\mathbf{\Omega}}^{(m-1)}(i)$. Moreover, because we separate the weighted-sum in (15) to a converged-term and a not-converged term, we can add the newly converged users from the m^{th} stage into the converged term in the weighted-sum chip signals during the accumulation.

$$\hat{r}_V^{(m)}(i) = \sum_{\substack{k=1,\\ V_k^{(m)}=1}}^{K} \hat{\alpha}_k[c_k(i)s_k^{(m-1)}] = \hat{r}_V^{(m-1)}(i) + \hat{r}_{V,new}^{(m-1)}(i) \tag{16}$$

$$\hat{r}_w^{(m)}(i) = \sum_{\substack{k=1,\\ V_k^{(m)}=0}}^{K} w_{opt,k}^{(m)}[c_k(i)s_k^{(m-1)}]. \tag{17}$$

where

$$\hat{r}_{V,new}^{(m-1)}(i) = \sum_{\substack{k=1\\ V_k^{(m-1)}=0\\ V_k^{(m)}=1}}^{K} \hat{\alpha}_k[c_k(i)s_k^{(m-1)}] \tag{18}$$

is the new term found converged at the m^{th} stage. Else if $V_k^{(m)} = 0$, we need to continue the

weight update and PRC computation for later stages as $\hat{\gamma}_{W,MF}(k) = \frac{1}{N}\sum_{i=0}^{N-1} \hat{r}_{w,opt}^{(m)}(i)c_k^*(i)$ and $\hat{s}_k^{(m)} = \mathrm{sgn}\big(\tilde{\mathbf{S}}_{MF0}(k) - \hat{\gamma}_{w,MF}(k) + w_s(k)\big)$.

Unlike the differential multi-stage implementation for the complete PIC in [3], the proposed weighted PRC guarantees convergence of the MSE. For the differential complete cancellation in [3], if the interference is estimated incorrectly, the mistake is locked in latter stages. However, the simulation results show that large power savings are achieved using the CMV in the proposed adaptive PRC with negligible loss in performance.

IV. Pipelined Multi-stage SoC Architecture

To meet the challenges and reap the rewards of SoC design, engineering teams need a scalable verification solution that addresses all aspects of the design cycle and reduces the verification gap. In this section, we focus on the hardware implementation of the NLMS based adaptive PRC architecture. The following issues will be addressed: design methodology, hardware resource/architecture constraints and system partitioning etc.

A. Catapult C HLS Architecture Scheduling

Functional verification is a critical bottleneck for SoC implementations. We apply an efficient Catapult C HLS methodology [9] from Mentor Graphics to investigate various pipelined architectures and different levels of parallelism. Catapult C provides architecture scheduling to generate efficient RTL on different resource/timing requirements. Configurable parallelism is enabled by assigning the number of FUs according to area/time constraints. The best solution would be the smallest design meeting the real-time requirements.

Fig. 4 shows the conceptual SoC architecture designed by Catapult C. The system level VLSI design is partitioned into several Subsystem Blocks (SB) according to the functionality and timing relationship. Each SB reflects some Catapult C design blocks. They are cascaded in pipeline by glue logics or Xilinx CoreGen dual-port RAM IP Cores in HDL Designer. Each SB consists of several Processing Elements (PE) either in pipelined mode or parallel mode. The pipelining and

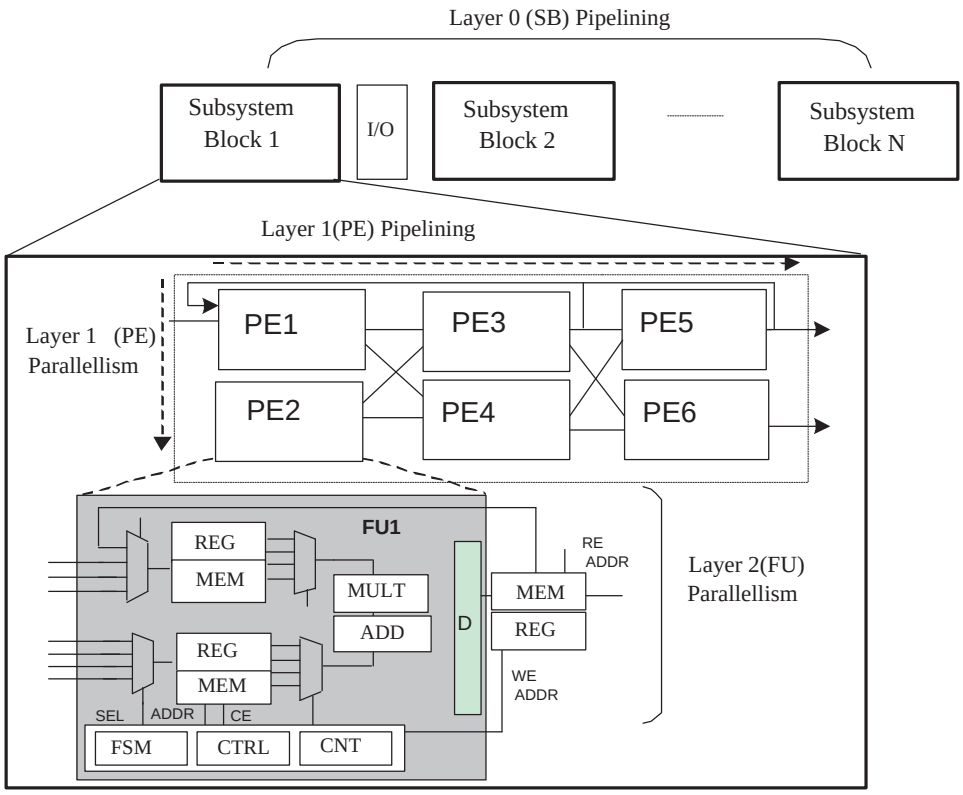

Figure 4: SoC architecture with layered parallelism/pipelining using Catapult C methodology.

parallelism in the PE level reflects the loop structures in the algorithm and has the most opportunity for optimization. The PE is mapped to the hardware resources in the Functional Units (FU) level, i.e., the multipliers, adders and register filters or memories. Finite-State-Machines (FSM), counters and controllers are generated to load the data at the correct timing to the input of the FUs. MUXs are utilized to share some of the big FUs. Data dependency and structure hazard may stall the pipeline while tbe logics and MUXs determine the clock rate and cycle number. The keys for optimization of the area/speed are loop unrolling, pipelining and the resource multiplexing. Multi-level parallelism/pipelining are studied extensively to find the most efficient VLSI architecture for the multi-stage adaptive PRC architecture.

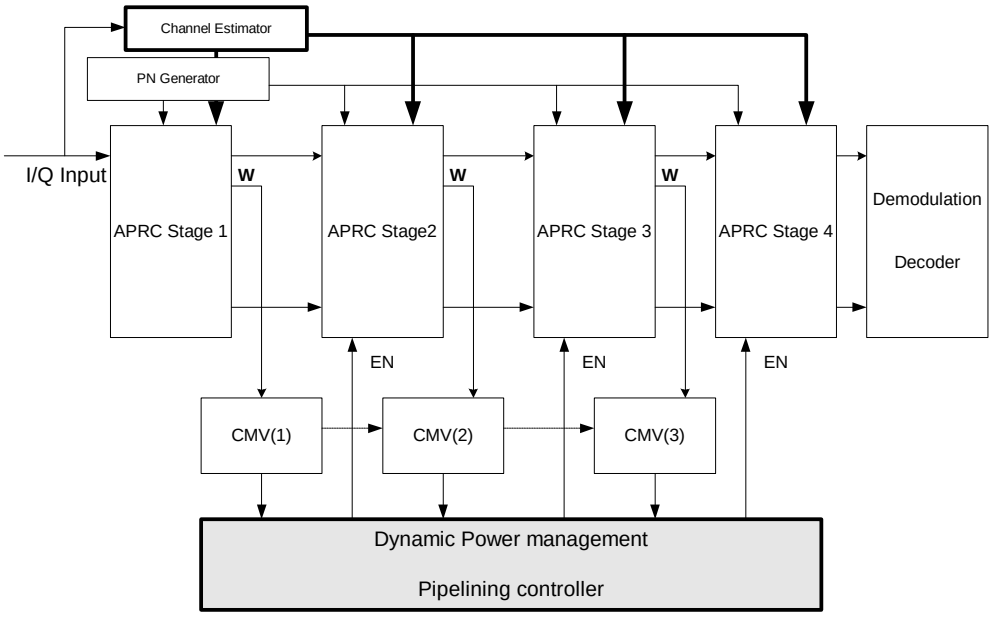

Figure 5: Block diagram of the pipelined APRC architecture with dynamic power management.

B. Pipelined Multistage VLSI Architecture

The top-level logic block diagram of the multi-stage adaptive PRC architecture using the CMV to control the dynamic power management is shown in Fig. 5. A PN generator generates the spreading codes either from a ROM block or from a simple combinational logic block. The channel estimator takes the input samples and the pilot symbols to estimate the channel coefficients and feeds them to the multi-stage APRC component. In each APRC stage, there is an NLMS block to update the weight and the PRC module to do the actual interference cancellation based on the optimal weights from the NLMS block. A CMV block will take the output of the weight update block and set the values for the convergence masking vector $V_k^{(m)}$. The CMV is sent to the dynamic power management module to generate the control logics for power saving of each stage. A pipeline controller also generates the control logic for the multi-stage pipelined processing to reduce processing latency.

The power management unit is also responsible for the generation of the clocks, which are supplied to the rest of the design. Clock gating is a commonly used technique to reduce dynamic power dissipation by gating off clock signals to registers, latches and clock regenerators. An

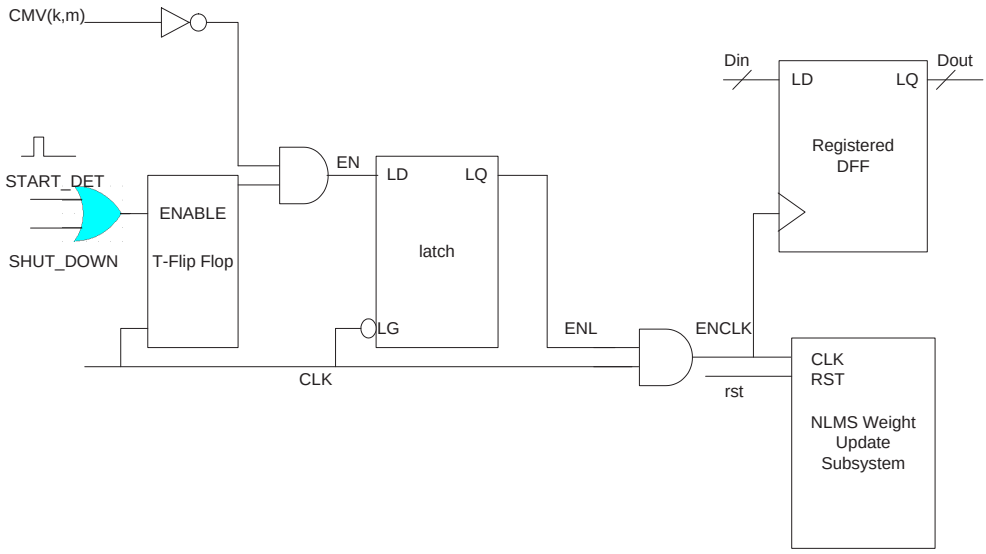

Figure 6: The clock gating dynamic power management using CMV.

example logic block is shown in Fig. 6. Gating may be done when there is no required activity to be performed by logic whose inputs are driven from a set of storage elements. Since output values from the logic will be ignored, the storage elements feeding the logic can be blocked from updating to prevent irrelevant switching activity in the logic. The "START_DET" and "SHUT_DOWN" signals are designed in a pattern to serve as the pulse into the T-flip flop to generate an enable signal output. This enable signal "AND"s with the inverse of the CMV for the k^{th} user at the m^{th} stage to generate an enable signal for the clock. It is worth noting that, in order to prevent glitches in the clock network, for each enable signal we must introduce a latch, which contributes an overhead in energy consumption. However, this overhead is negligible compared with the overall system complexity. Consequently, in order to reduce the energy dissipation, a simple circuit detects the occurrence of convergence events for each user at each stage. It also detects the convergence event of the earlier stages. When this occurs, the clocks to the NLMS and the PRC modules are blocked and no further weight calculations are performed.

C. System Level Partitioning

Because the transmitter design is relatively simple, we focus on the receiver design architecture. The loop structures and the intrinsic timing in the algorithm need to be arranged well to achieve

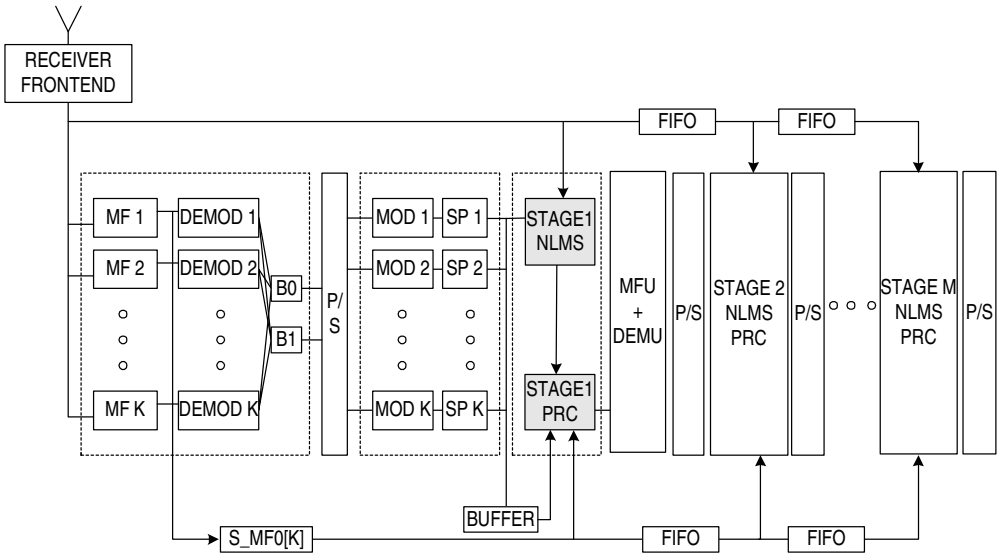

Figure 7: System partitioning of the multi-stage Adaptive PRC VLSI Architecture.

pipelining and parallelism. Pipelining, timing balance and modularity are important considerations for the system partitioning. The top-level logic block diagram is depicted in Fig. 7.

The first stage matched filter output for K codes are stored in memory block $\mathbf{S}_{MF0}[K]$ for the symbol level PRC. At the output of the demodulators (DEM), the detected bits for K users are packed into two words $\mathbf{B}_0$ and $\mathbf{B}_1$ for QPSK modulation. The output of the re-constructor using modulator (MOD) and spreading (SP) units is passed to the stage 1 NLMS block for weight computation and simultaneously stored in a buffer for PRC while the weight is being computed. The signal after the interference cancellation is detected by merged Matched-Filter-and-Demodulator-Units (MFU+DEMU) for K users. Multiple stage hardware units of the NLMS-PRC block are laid out in pipelined mode. The detected bits are passed to the later stages for multi-stage processing. FIFOs are applied to balance the processing latency in different chains. The input bit streams for K users are packed into one single word bit vector buffer as: $\mathbf{B}[n] = \sum_{k=1}^{K} b_k(n)2^{k-1}$ to save the storage. The spreading codes for K-users can also combine to form a code vector ROM as $\mathbf{C}[i] = \sum_{k=1}^{K} c_k(i)2^{k-1}$.

D. Pipelined Weight-Updating-Block

The NLMS is a major design bottleneck since it involves divisions and multiplications with feedback structures as (8). This design block takes the input vector for the chip-based complex NLMS algorithm and computes the optimal weights for all the users in each symbol. Although it is relatively straightforward to synthesize the high-speed architectures for feed-forward-only signal processing structures such as the conventional PIC, it is considerably more difficult to synthesize similar architectures when there is a feedback structure. In the NLMS adaptation, the error of the weighted hard-decision signal is used to adjust the weight coefficients in real time.

There are two top-level loop structures $L1, L2$ corresponding to the equations in (7) and (8). $L1$ loop is the recursive loop for the updates in chip-basis for each symbol. $L2$ updates the weight estimates from registers to memory blocks when one symbol is ready. The loops are mapped to hardware units as shown in the block diagram in Fig. 8. In $L1$ loop there are two second-level loops corresponding to the user indices: $L1.1$ computes the weighted estimation of the received signal based on the current weights. $L1.2$ computes the iterative weights for K users. According to the loop structures for the code index k and chip index i, the NLMS block can be partitioned into two major functions: the Weighted-Sum-Function (**WSF**) as in equation (7) and Weight-Adaptation-Function (**WAF**) as in equation (8). In the **WSF** sub-block, the estimated hard-decision bits are extracted from the bit vectors $\mathbf{B}_0$ and $\mathbf{B}_1$ by the De-Packing Unit (**DPU**) block. The $\hat{\mathbf{\Omega}}^{(m-1)}(i) = [c_1(i)\hat{s}_1^{(m-1)}\ c_2(i)\hat{s}_2^{(m-1)}\ \cdots c_K(i)\hat{s}_K^{(m-1)}]$ vector is generated using the same Modulator-Spreader-Unit (**MSU**) as in the transmitter from the estimated bits and the spreading code vector $\mathbf{C}[i]$. This vector is then stored either in memory blocks or register files. In the same loop structure, the Chip-Weighting-Unit (**CWU**) and Complex-Add-Unit (**CAU**) will generate the weighted sum of the replica as in (7). This replica of received signal is then subtracted from the received chip samples to form the residual error as $\epsilon^{(m)}(i) = r(i) - \hat{r}_w^{(m)}(i)$. The $\mathbf{\Omega}$ vector and the residue error are then sent to the **WAF** block for weight update. The omega vector is first multiplied by the residue and then multiplied by the factor of $\mu/norm$. This quantity is then added to the previous iteration of the weights and written back to the $W_{tmp}[K]$ space. This is repeated iteratively for the chips in one symbol. After the weights are ready for each symbol, the Weight Load Process (**WLP**) in loop $L2$

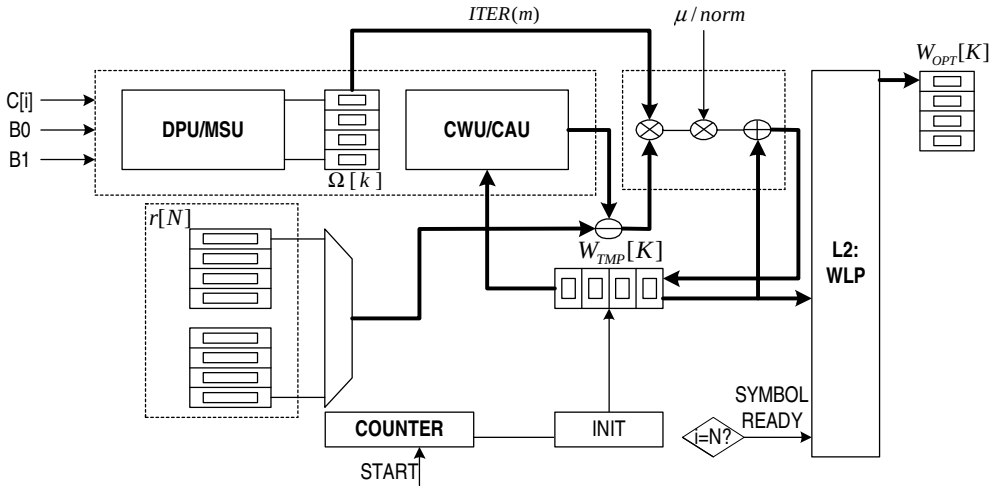

Figure 8: The logic block diagram of the NLMS weight updating module.

loads the optimal weights for interference cancellation.

Ping-pong buffers are designed to store the input chip samples of the next symbol while the NLMS block is computing the weights. In the NLMS $L1$ structure, a counter i is applied to control the iteration. For the first chip of each symbol, the initial values of the weight vector are set to be the channel estimation for each user scaled by the $N_{scale} = 2^{B_W}$ as $\mathbf{w}^{(m)}(i) = round\{[\hat{\alpha}_1\ \hat{\alpha}_2 \cdots \hat{\alpha}_K] * N_{scale}\}$, where B_W is the number of scaling bits to avoid overflow.

The top-level loop for the K-user $\mathbf{\Omega}$ vector generator is merged with the loop in the NLMS algorithm. As a summary, the scripts of the **WSF** and **WAF** in $L1.1$ and $L1.2$ are shown in Table 1. In the **WSF** block, a vector processing of the modulation is formed for all the K users. In the **WAF** block, we need to compute the norm of the $\hat{\Omega}^{re,im}(k)$ vector. A straightforward computation of the norm for the $\mathbf{\Omega}$ vector is as $||\hat{\Omega}||^2 = \sum_{k=1}^{K}[\hat{\Omega}_{re}(k)^2 + \hat{\Omega}_{im}(k)^2]$. This norm has the complexity of $2K$ multiplications and $(K-1)$ additions. If $\hat{\Omega}^{re,im}(k)$ are stored in memory arrays, this also requires $2K$ memory READs. However, since $\hat{s}_k^{(m-1)} \in \{\pm 1 \pm j\}$ and $c_k(i) \in \{\pm 1\}$ for QPSK, the norm does not need to be computed for each symbol individually. It can be shown that $||\hat{\mathbf{\Omega}}||^2 = 2K$ is a constant. The division can be implemented by a right-shift of $\log_2(2K)$. Since the step size μ does not need to be a very accurate particular value without loss in performance, we can combine μ and the norm into one coefficient and right-shift only by $\log_2(K)$, which can be computed as a

Table 1: Scripts For The **WSF, WAF** Loops

WSF (L1.1): for $k \in [1, K]$	**WAF** (L1.2) for $k \in [1, K]$
$w^{re}(k) = N_{scale}; w^{im}(k) = 0;$ if($i == 0$)	$\Im^{re} = \epsilon^{re}\hat{\Omega}^{re}(k) - \epsilon^{im}\hat{\Omega}^{im}(k)$
$\hat{\Omega}^{re}(k) = \{1 - 2[(\mathbf{C}_i >> k)\&1]\}\{1 - 2[(\mathbf{B}_0 >> k)\&1]\}$	$\Im^{im} = \epsilon^{im}\hat{\Omega}^{re}(k) + \epsilon^{re}\hat{\Omega}^{im}(k)$
$\hat{\Omega}^{im}(k) = \{1 - 2[(\mathbf{C}_i >> k)\&1]\}\{1 - 2[(\mathbf{B}_1 >> k)\&1]\}$	$w^{re}(k) += [(\mu\Im^{re}) >> B_w] >> \log_2 K$
$\hat{r}_w^{re} += w^{re}(k)\hat{\Omega}^{re}(k) - w^{im}(k)\hat{\Omega}^{im}(k)$	$w^{im}(k) += [(\mu\Im^{re}) >> B_w] >> \log_2 K$
$\hat{r}_w^{im} += w^{re}(k)\hat{\Omega}^{im}(k) + w^{im}(k)\hat{\Omega}^{re}(k)$	if (i==N) then { $w_{opt}^{re,im} = w^{re,im}(k)$;}

constant offline.

E. Bit-ware Sumsub-Mux-Unit

A conventional design of the Spreading-Unit(**SU**) and Chip-Weighting-Unit (**CWU**) in (7) and (8) utilizes explicit dedicated multipliers for all the involved multiplications. The circuit is shown in Fig. 9 for two users. Each **SU** has 2 multipliers and each **CWU** has 4 multipliers. Moreover, there will be 2 adders for each **CWU** and a pipelined **CAU** tree layout is required for a fully pipelined summation of K users. The complexity is still rather high with $6K$ multiplications for the loop. However, since the real and imaginary parts denoted by $\hat{s}^{re/im}(k), \hat{\Omega}^{re/im}(k)$ and $\mathbf{C}_i(k)$ only take values from $\{\pm 1\}$, we use $\{0, 1\}$ instead to represent these values and pack the K users into vector words $\mathbf{B}_0, \mathbf{B}_1, \mathbf{C}_i$. The bit-ware values are extracted from the words as $b_0 = (\mathbf{B}_0 >> k)\&1$; $b_1 = (\mathbf{B}_1 >> k)\&1$; $C_k(i) = (\mathbf{C}[i] >> k)\&1$. The actual value of $\hat{\Omega}(k)$ can be determined from a truth table based on different input bits of the spreading code and the hard decision bits. By using $\{0, 1\}$ instead of $\{\pm 1\}$ to represent $\hat{\Omega}(k)$ too, the logic design is shown to be

$$uint1 : \hat{\Omega}^{re}(k) = \{[(\mathbf{C}_i >> k)\&1]\}\text{XOR}\{[(\mathbf{B}_0 >> k)\&1]\} \tag{19}$$

$$uint1 : \hat{\Omega}^{im}(k) = \{[(\mathbf{C}_i >> k)\&1]\}\text{XOR}\{[(\mathbf{B}_1 >> k)\&1]\} \tag{20}$$

where *uint1* denotes the unsigned one-bit data type.

The multiplication by $\hat{\mathbf{\Omega}}^{(m-1)}$ with 2-bit values of $\{\pm 1\}$ then can be implemented with MUX circuits controlled by the decoder of $\hat{\Omega}(k)$ with 1-bit values $\{0, 1\}$. The multiplications in (7) are

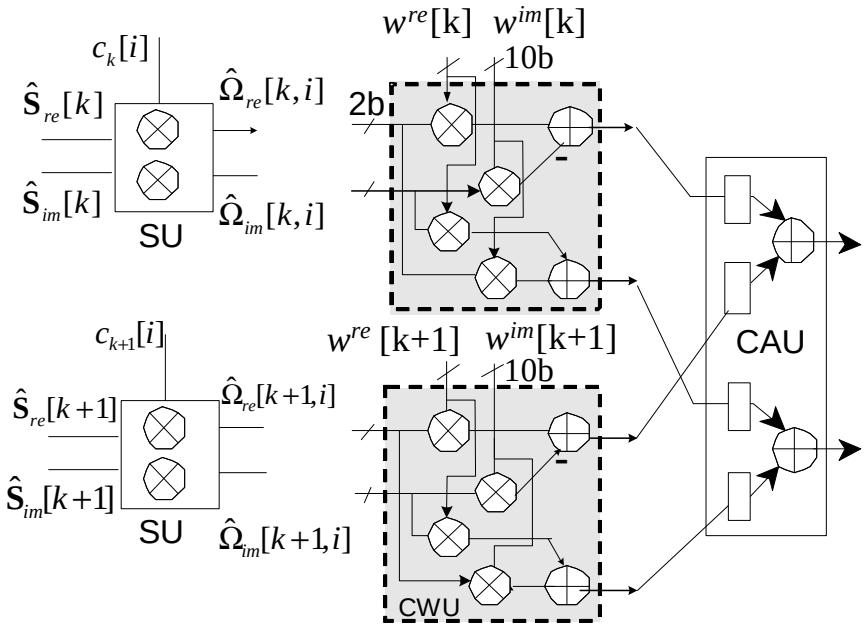

Figure 9: The multiplier-based Chip-Weighting Unit (CWU).

implemented as **Sumsub-Mux-Unit** for **W**eighted symbols (**SMUw**) as

$$\begin{cases} \hat{r}_{w,sum} = w^{re}(k) + w^{im} \\ \hat{r}_{w,sub} = w^{re}(k) - w^{im}(k) \\ \hat{r}_w^{re} += A^{re}, \quad A_{re} \in \{\pm\hat{r}_{w,sum}, \pm\hat{r}_{w,sub}\} \\ \hat{r}_w^{im} += A^{im}, \quad A_{im} \in \{\pm\hat{r}_{w,sum}, \pm\hat{r}_{w,sub}\} \end{cases} \tag{21}$$

The same structure can be used for $\hat{\mathbf{\Omega}}^{(m-1)}(i)^*\epsilon^{(m)}$ in (8) as an **SMU** block for **E**rror (**SMUe**). This is shown as $\Im^{re} \in \{\pm\hat{\epsilon}_{sum}, \pm\hat{\epsilon}_{sub}\}$ and $\Im^{im} \in \{\pm\hat{\epsilon}_{sum}, \pm\hat{\epsilon}_{sub}\}$ where $\hat{\epsilon}_{sum} = \epsilon^{re} + \epsilon^{im}$ and $\hat{\epsilon}_{sub} = \epsilon^{re} - \epsilon^{im}$. The circuit logic for one **SMUw/SMUe** is depicted in Fig. 10, where only the sign and input to the accumulator is controlled by the 4-way MUX. The "Sel" signal decoder generates the $Sel[k]$ signals to replace the original $\mathbf{\Omega}$ vector. The difference of the **SMUw** and **SMUe** is the input to the **Connection Network** (CN) to the MUX. The Connection Network can be designed from truth tables as in Table. 2 for different "Sel" signals.

The **WSF** and **WAF** blocks for the NLMS algorithm then can be integrated with these basic design blocks. An example with two **SMUw** and **SMUe** engines in parallel is shown in Fig. 11. In the **WSF** function, the "SELdecoder" takes the $\mathbf{C}[i]$ and $\mathbf{B}_0, \mathbf{B}_1$ to generate the select signals

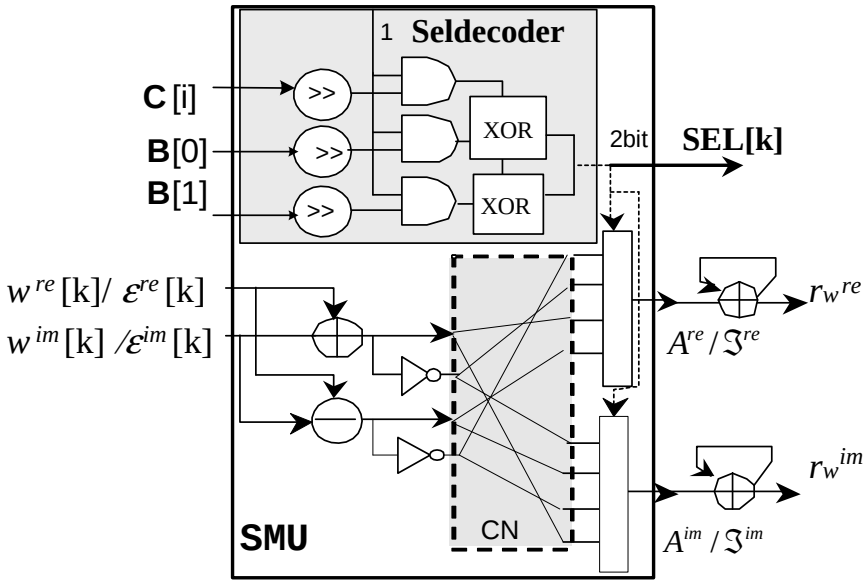

Figure 10: The SMU-based Chip-Weighting Unit.

Table 2: The connection networks for the SMU of the weighted sum symbols and residual errors.

$\hat{\Omega}^{re}\hat{\Omega}^{im}$	A_{re}	A_{im}	$\Im^{re}$	$\Im^{im}$
00	$-\hat{r}_{w,sub}$	$-\hat{r}_{w,sum}$	$-\hat{\epsilon}_{sum}$	$\hat{\epsilon}_{sub}$
01	$-\hat{r}_{w,sum}$	$\hat{r}_{w,sub}$	$-\hat{\epsilon}_{sub}$	$-\hat{\epsilon}_{sum}$
10	$\hat{r}_{w,sum}$	$-\hat{r}_{w,sub}$	$\hat{\epsilon}_{sub}$	$\hat{\epsilon}_{sum}$
11	$\hat{r}_{w,sub}$	$\hat{r}_{w,sum}$	$\hat{\epsilon}_{sum}$	$-\hat{\epsilon}_{sub}$

for the **SMUw**. The **SMUw** takes the input from the temporary weight memory block. A **CAU** adds the two portions of paths to get the total weighted sum chip signal. It is then subtracted from the received original signal to generate the error, which is input to the **SMUe** module in the **WAF** block. After multiplying the "μ_{normed}", it is adjusted by the weights from the previous iteration and written back to the memory. In this way, each engine acts as a single processor for serial processing of $K/2$ users. Dramatic optimization in the VLSI area and timing closure can be achieved with this design compared to the conventional multiplier-based design as shown later.

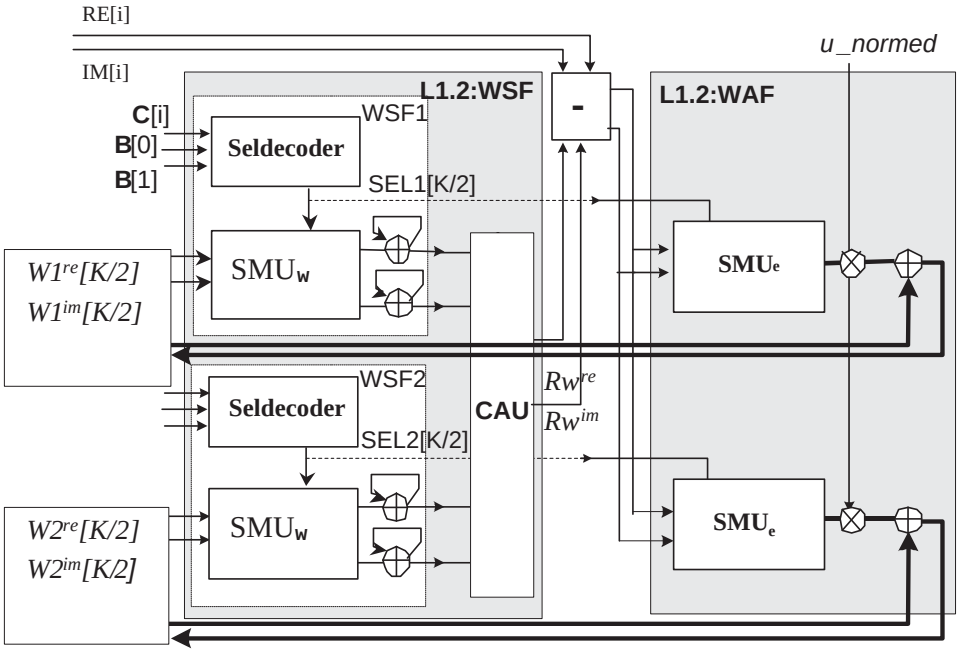

Figure 11: The data path of the SMU-based NLMS architecture using WSF and WAF.

F. Weighted Matched Filter and PRC Architecture

Another major block is the Weighted-Sum-Matched-Filter and the Residue-Compensation block denoted by equations (9). Similar to the NLMS block, symbol level Sumsub-MUX-Unit for Weighted-Symbol (**SMUws**)is designed with bit-ware combinational logic to generate $w_s[k]$. In this case, the Weighted-Symbol (**WS**) SMU is controlled only by the "SelDecoder" triggered by the $\mathbf{B}_0$ and $\mathbf{B}_1$ vectors. A MUX controlled by the spreading codes and the accumulator forms the equivalent Weight-Matched-Filter-Unit (**WMFU**). This generates the optimal weighted sum chip signal $\hat{r}_{w,opt}$ if the **WMFU** is accumulating partial results based on the user index k. This design module is shown in Fig. 12. Based on this basic design module, the complete data path logic block diagram for the Weighted-Sum-Matched-Filter and Residue-Compensation process denoted by equations (9)-(13) is shown in Fig. 13. This figure shows an example with two parallel Processing Elements built from the combinational logic. The K users are split into two groups of $K/2$ users. The users in each group utilize one PE in serial. In each PE, the optimal weights for one symbol are input into the **SMUws** module to form the weighted symbol $w_s[k]$ and the weighted sum

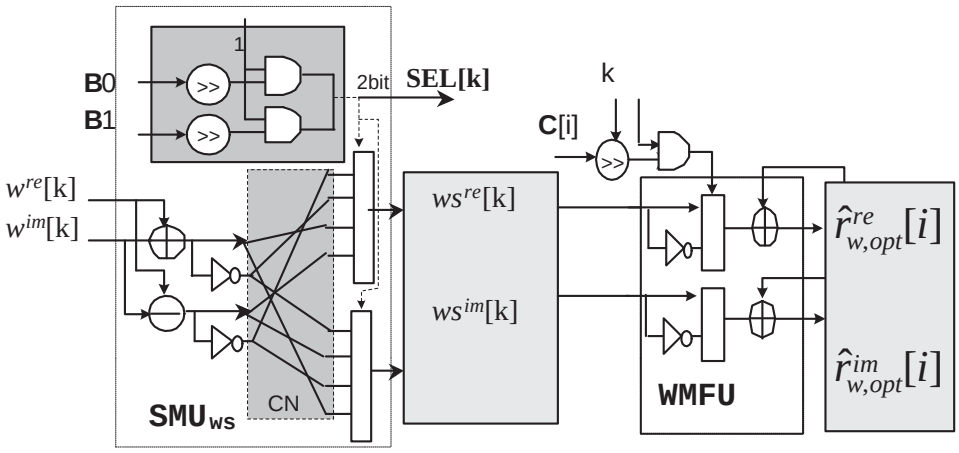

Figure 12: Single processing element for the Sumsub-MUX-Unit for Weighted Symbol (**SMUws**) and Weighted Matched Filter Unit (**WMFU**).

chip signal. The weighted sum chip signal is then detected with the chip matched-filter **WMFU** to form the $\hat{\xi}_{w,MF}(k)$, which is marked as $\hat{\Re}(k)$ in the figure. This quantity is subtracted from the $\tilde{\mathbf{S}}_{MF0}[k]$ and then added by the weighted symbol. This finally leads to the matched filter output of the interference-cancelled signal. When one symbol is accumulated, a "SymRDY" signal is asserted for the demodulator unit to read the symbol estimates.

Notice that in this design, we do not have to use general-purpose multipliers. The very simple combinational logic bit-level VLSI architecture can achieve much higher clock rate, as we will show later. This allows more time for the processing of each user and each chip and avoids the fully parallel layout of duplicate hardware design units at the order of user number K. This feature gives more flexibility in designing a configurable VLSI architecture that is important and challenging for the multi-code CDMA system.

V. Simulation and Emulation Results

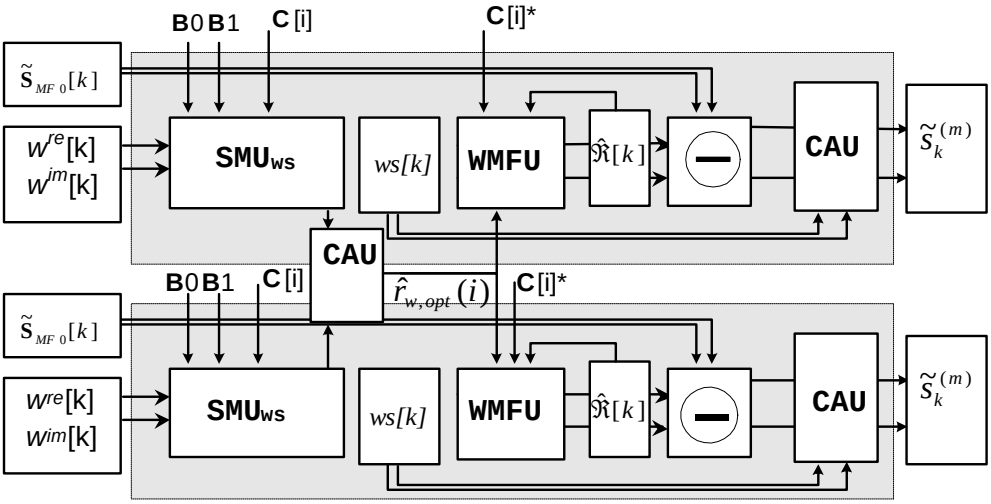

Figure 13: The data path of the SMU-based Weighted-Sum-Matched Filter and Residue-Compensation architecture.

A. Floating-point Performance

In Fig. 14, the BER performance versus the number of users for a fixed Signal-to-Noise Ratio (SNR) of $4dB$ is shown for stage 2 using random codes. The Spreading Factor SF is 64. For the PPIC case, a set of intuitive weights that satisfy $w_{m-1} < w_m$ are simulated. The PPIC starts to outperform the complete PIC after the number of users increases above 12. The performance of the proposed APRC outperforms both the PIC and PPIC significantly. It can be concluded that when the system load is low, the complete PIC works fairly well. However, when the system load is high, PPIC starts to outperform complete PIC. On the other hand, the adaptive PRC outperforms both the PIC and PPIC in a wide range of the system load. This demonstrates the superior performance of the adaptive PRC algorithm.

B. Fixed-point Implementation and Performance

The VLSI implementation requires fixed-point arithmetic. The reduction of the bit width almost linearly reduces the design size, hardware complexity and power consumption. However, the stability of the algorithm and the performance may suffer from excessive finite word length effects due to the overflow and quantization noise, unless all signals are scaled properly and sufficient

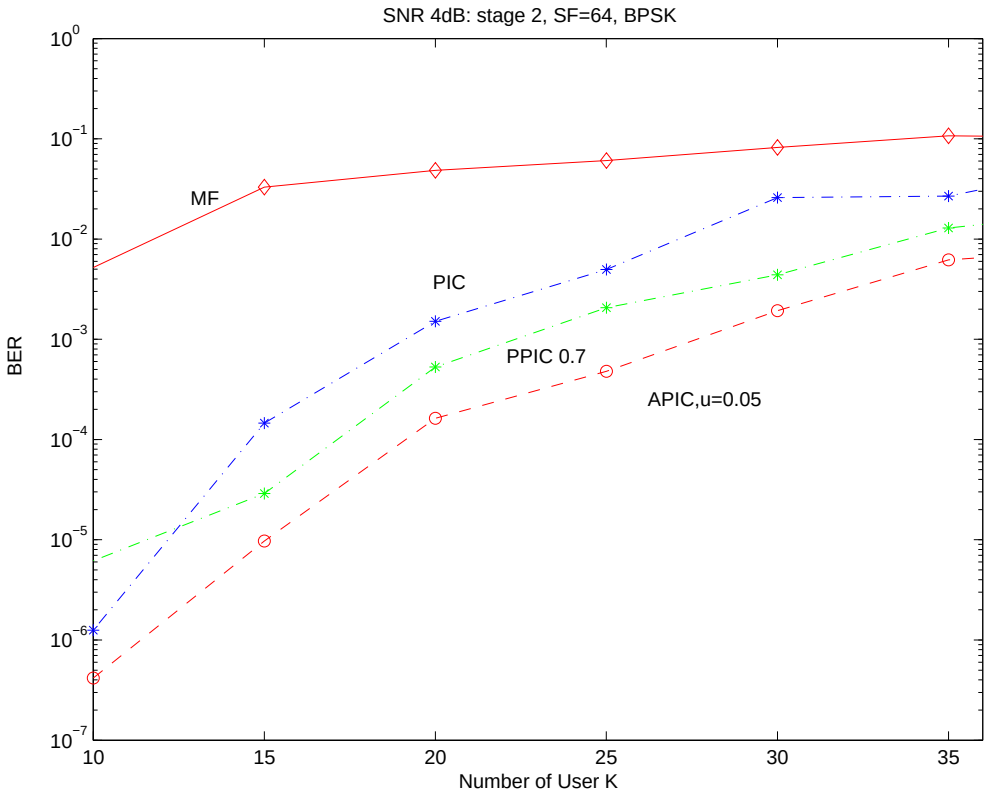

Figure 14: Floating point BER versus user number K: stage 2.

word length is assigned. So it is important to find a reduced word-length with negligible performance degradation. Because of the multiplications or divisions in the algorithm, overflow should be avoided by scaling the result back to the correct word length. Meanwhile, the precision should be kept enough to avoid underflow and divide by zero error. The fixed-point performance for QPSK modulation is shown in Fig. 15 and 16 for QPSK. It can be seen that for the APRC scheme, 10 bits will generate similar performance as the floating-point results. For the PPIC scheme, 8 bits have similar performance as $10-12$ bits and floating-point in the 3^{rd} stage. For the APRC, 9 bits and 8 bits will diverge from the floating-point performance considerably. Even though, it performs better than the PPIC algorithm.

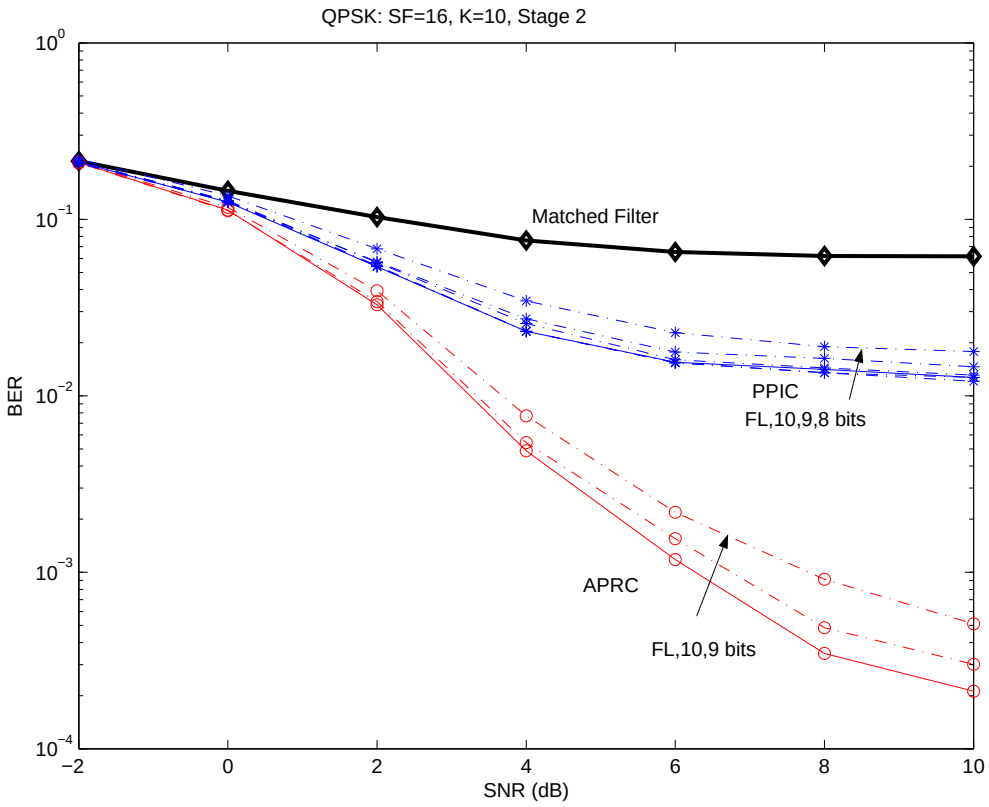

Figure 15: Fixed-point BER versus SNR: QPSK, stage 2.

C. Performance and Complexity Tradeoff Using CMV

The BER performance using convergence-masking vector for dynamic power management is compared with the original APRC algorithm in Fig. 17 and Fig .18 for different stages and relative thresholds. In both figures, the spreading factor is set to 16 and SNR is set to 12 dB. Fig. 17 shows the performance for a 10-user system while the number of users in Fig. 18 is 14. For a 10-user system, when the threshold is set to be 50% at stage 1, 70% at stage 2 and 90% at stage 3 and stage 4, the performance drop is negligible. However, if the threshold is below a certain level, e.g., 80% at stage 4 or 40% at stage 2, significant performance degradation is observed when compared with the original PRC scheme. For a 14-user system, the performance degradation is less sensitive to the threshold level because the BER floor of the 14-user system is higher than the 10-user system. If the threshold is set to 95%, the BER is almost the same as the "always-update" adaptive PRC scheme.

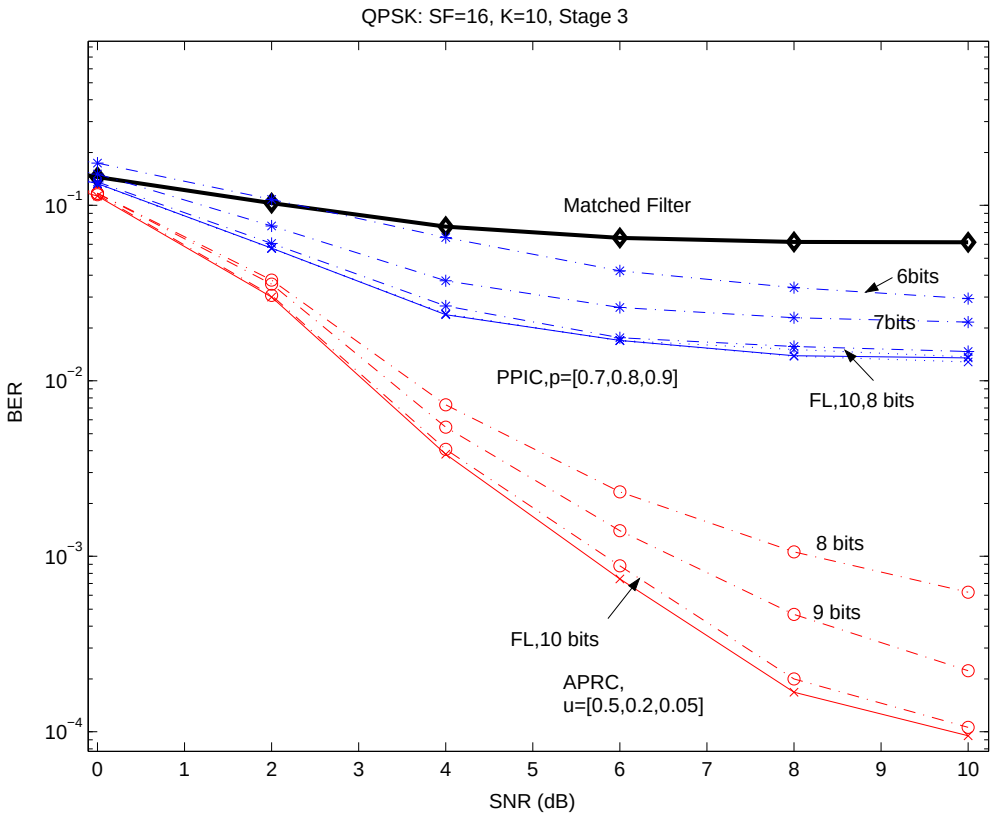

Figure 16: Fixed-point BER versus SNR: QPSK, stage 3.

D. Active Rate

The active rate of one component $\zeta_{act} = \tau_{act}/T_{all} * 100\%$ is the percentage of time when the component is not shut down through clock gating. Thus, the active rate is an indicator of the power savings for the pipelined VLSI architecture. In Fig. 19, we demonstrate the active rate of each stage under a different threshold level. The plain solid curve is the active rate of stage 1, the solid curve with square is stage 2, the diamond curve is stage 3 and the dotted curve is stage 4. From the simulation results in Fig. 17 and Fig. 18, it is demonstrated that different thresholds could be applied to different stages. If we choose a threshold of 75% for stage 1, 90% for stage 2 and 95% for stages 3 and 4, it leads to a 35% active rate for stage 1, a 10% active rate for stage 2 and roughly 5% active rate for stage 3. For stage 4, only if the threshold is set to above 96% will the active rate increase to 5%. The low active rate indicates that large power savings can be achieved over the original design with little or no loss in system performance.

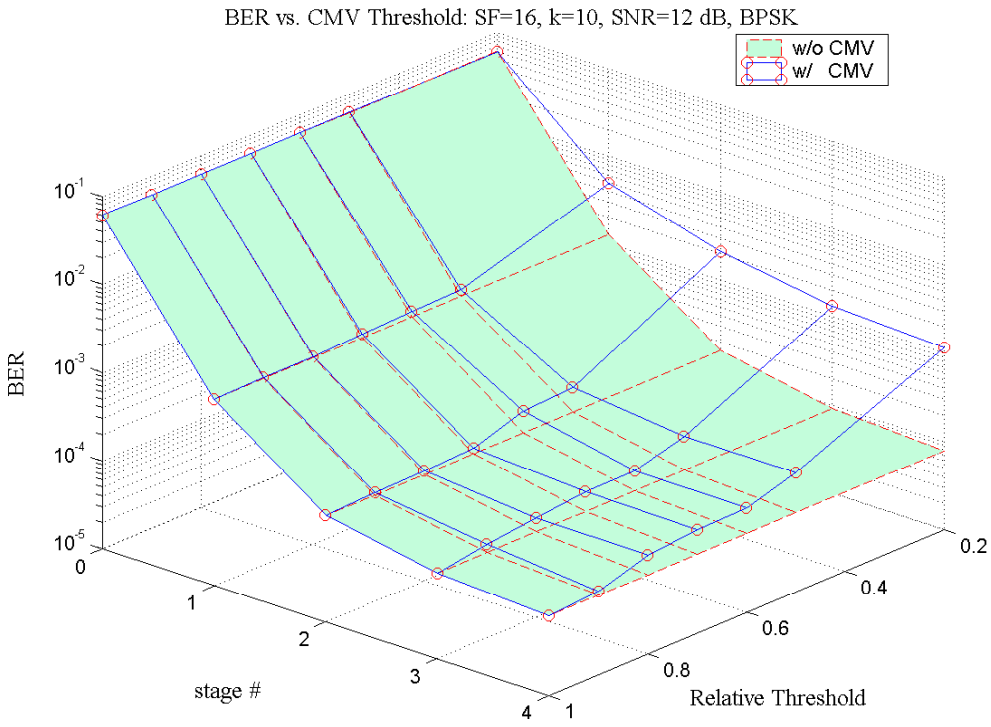

Figure 17: The BER performance vs. convergence masking vector threshold.

VI. SoC Architecture Design Space Exploration & Synthesis Results

A. Resource Mapping and Architectural Constraints

There are tradeoffs among the speed and size by using different storage hardware. If register files are applied to map the arrays, they can be accessed in parallel in one cycle. However, if the data arrays are mapped to memory block, only one entry can be accessed for a single memory block. Sometimes, the memory access race problem could stall the pipeline and force the design to process in serial. This increases the latency and reduces the processing speed. So register files tend to provide more parallelism. On the other hand, if multiple register files need to share the functional units, MUXs need to be applied in front of each input of the functional units. For a multi-user PIC system, they could be very large MUX with up to N inputs, where N is the spreading factor. In FPGAs, the large MUX could be even larger than the functional units such as the adders themselves. This can be a major contribution to the design size. Size reduction

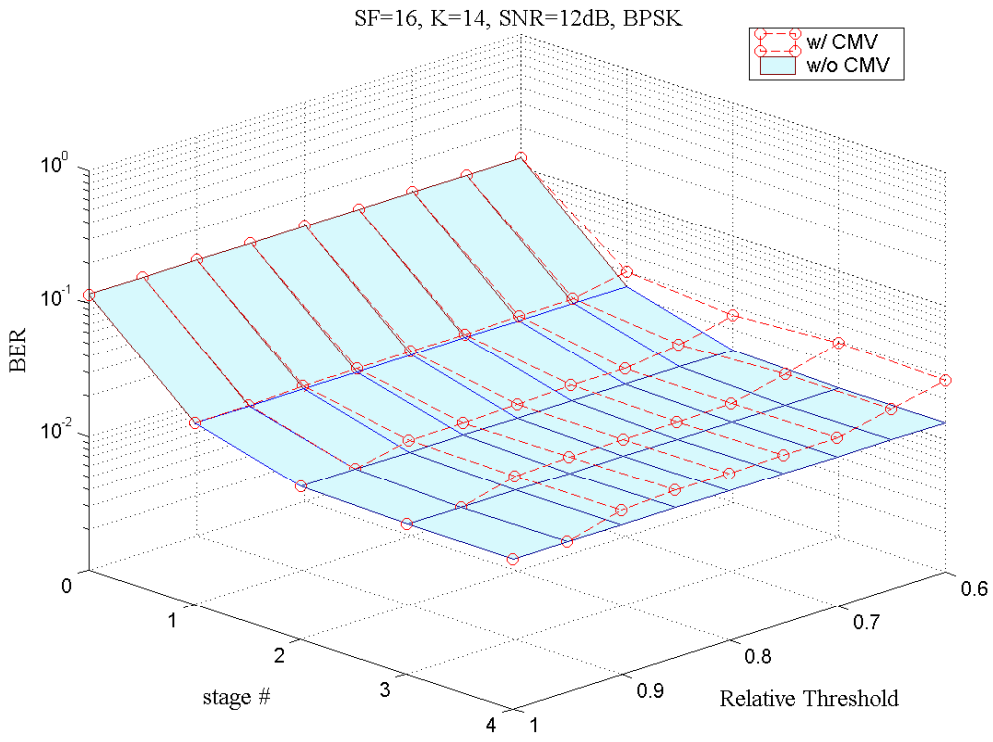

Figure 18: The BER performance vs. CMV threshold in multiple stages.

through multiplexing fails in such situations. Moreover, The big MUX makes the timing balancing difficult, so as to generate designs with slower clock rate.

The proposed VLSI architectures are implemented with the Catapult C methodology and prototyped on the Xilinx FPGA Virtex-II V6000 platform. The target real-time design specification follows the WCDMA and HSDPA system, where the chip rate is 3.84 MHz [9] for downlink wireless multimedia services. The spreading gain is 16. If the working clock rate is set to 38.4 MHz, there are 10 cycles resource for each chip and 160 cycles for one symbol. In Vitex-II V6000, there are 144 dedicated 18×18 ASIC multipliers. These multipliers do not occupy the Configurable-Logic-Block (CLB) sizes. Hardware net list is generated with Xilinx ISE P&R tools and verified in a Nallatech FPGA development platform [10].

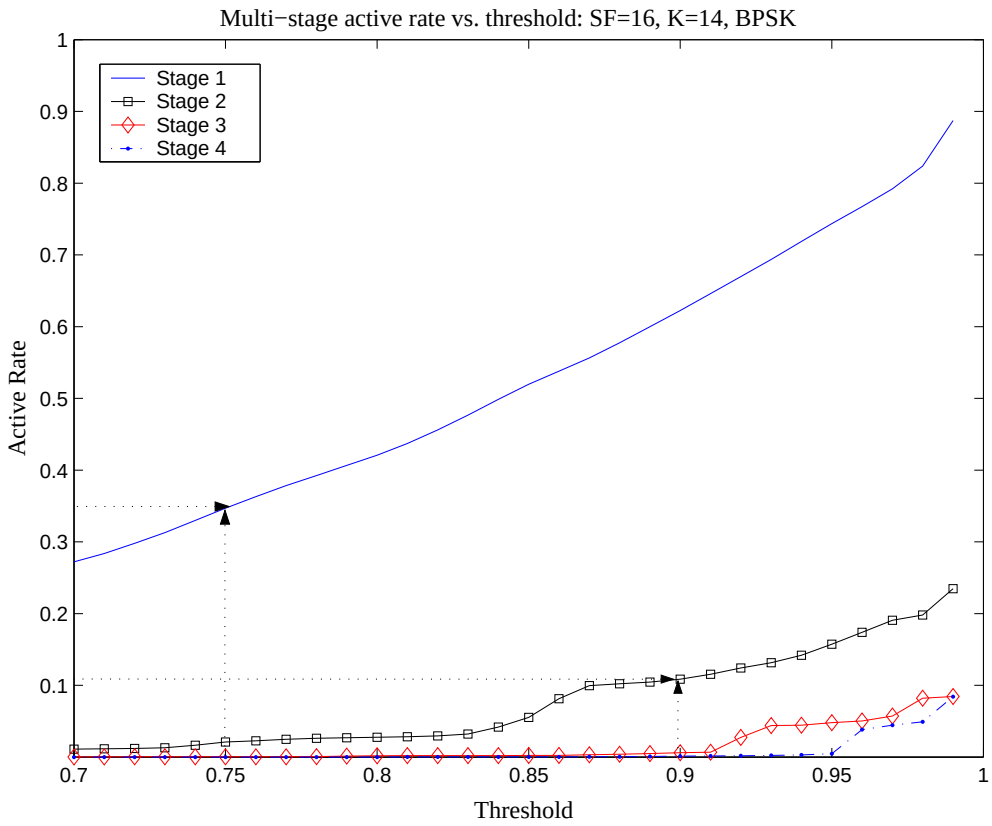

Figure 19: The active rate of the NLMS update and residue compensation operation vs. CMV threshold.

B. Scalable Architecture for the NLMS Module

We first explore the different mapping and pipelining options for the NLMS block. Table 3 shows the design space exploration for the resource mapping and architecture constraints of the multiplier-based NLMS weight update block. The abbreviation codes used in the table have the following meaning: “MEM” means the storage option for the array variables is memory; “REG” indicates that the arrays are stored in register files. For the loop architecture constraint, “-” denotes no special processing such as pipelining and the loop remains rolled for the serial processing; “P” means that the loop is pipelined and “UR” means that the loop is unrolled. Dedicated ASIC multipliers are applied for all the involved multiplication.

Table 4 shows the corresponding netlist synthesis specifications. Solution 1 applies memory blocks for both the I/O interface and local variables. All the loop structures remain rolled with no

Table 3: Hardware architecture constraints for the Mult-based NLMS architectures

Sol	I/O	Local	L1	L1.1	L1.2	L2
1	MEM	MEM	-	-	-	-
2	REG	REG	-	-	-	-
3	MEM	REG	-	P	P	P
4	REG	REG	-	UR	UR	P

Table 4: Synthesis results for the Mult-based NLMS architectures

Sol	CLB	RAM	DFF	Cycle	$F_{clk}(MHz)$	ASICMult
1	1303	7	703	856	55.0	18
2	1496	0	850	439	66.0	4
3	4014	0	2791	307	45.4	32
4	2697	0	2198	147	48.4	91

pipelining. For solution 2, both the interface and internal variables use register files. But all the loop structures remain rolled with no pipelining. In solution 3, the internal variables apply all register files while the interface apply the memory blocks for inter-module communication. Solution 1 represents the area-constrained design because all the multiplications can reuse multipliers in serial without using MUXs at the input of the multipliers. But this is also the slowest design with 856 cycles latency because the RAM bandwidth is limited. When the RAMs are replaced by register files, it is a little bit faster with 611 cycles latency as shown in solution 2. But the size is also bigger. In solution 3, since the pipelining of the loops is designed, the design is much faster with 307 cycles. But this requires more complex control logic and more multipliers for pipelined processing. To meet the 160 cycles latency requirement, register files have to be used everywhere and the two major loops need to be unrolled in parallel as shown in solution 4. Although it only has 147 cycles latency, the design is rather big with 91 dedicated ASIC multipliers. Without using special VLSI design circuit, the algorithm is not very suitable for practical implementation.

In Table 5, the design architectures based on the SMU combinational logic circuit with different resource mapping and architecture constraints are compared. Solution 1 corresponds to

Table 5: Synthesis results for the SMU-based NLMS architectures

Sol	CLB	RAM	DFF	Cycle	$F_{clk}(MHz)$	ASICMult
1	963	8	513	529	68.5	4
2	1496	0	850	439	66.0	4
3	3477	0	1076	151	59.0	9

the most area-constrained design. However, it could not meet the real time requirement with 529 cycles latency. Solution 2 only replaces the memory blocks with register files and has the same computation structure. It is slightly faster with 439 cycles latency. But the usage of MUX for the remaining multipliers leads to higher CLB number. Solution 3 uses register files for local arrays and memory for the interface. To meet the latency requirement, we unroll the L1.1 and L1.2 loops and design the logic circuits using the SMU units jointly for all the 10 users. Pipelining is achieved for the modules built from simple combinational gates. It could be seen that in solution 3, we meet the time requirement with 151 cycles at 59 MHz clock rate. 9 dedicated multipliers are still used for the remaining multiplications. Compared with the fastest design using dedicated multipliers, it achieves $10\times$ saving in the number of multipliers.

C. Scalable Architecture for the PRC-MFB Module

Table 6 and 7 present the scalable specifications for the multiplier-based and **SMUws**-based architectures of the PRC-MFB module, respectively. Solution 1 represents the most compact design in CLB consumption because it uses memory blocks for all arrays and no advanced architecture is designed for all level loops. Solution 2 uses register files for all arrays with no pipelining too. It is a little bit faster but also bigger. Solution 3 utilizes elegant partial loop pipelines/unrolling for different levels based on the algorithm structure. Solution 4 first unrolls the L1.1 and L1.2 and designs L1 pipelining with initial interval of 2. It is seen that for the multiplier-based design, although solution 3 is much faster than the solutions 1 and 2, it still does not meet the 160-cycle constraint. Only a fully pipelined architecture in solution 4 meets the 160-cycle requirement with 16 ASIC MULTs. This gives a design with 35 cycles latency, which is much faster than necessary.

Table 6: Synthesis results for the Mult-based PRC-MFB architectures

Sol	CLB	RAM	DFF	Cycle	$F_{clk}(MHz)$	ASICMult
1	389	2	329	594	67	4
2	966	0	729	434	82.6	2
3	1731	0	1186	290	91.6	4
4	869	0	389	35	77.5	16

Table 7: Synthesis Results For The SMU-Based PRC-MFB Architectures

Sol	CLB	RAM	DFF	Cycle	$F_{clk}(MHz)$	ASICMult
1	888	8	584	644	102.3	1
2	2555	0	1592	294	78.5	0
3	971	4	618	159	122.5	0
4	2520	0	1998	42	117.6	0

Table 7 explores the architectures for the SMU-based design architectures. In solution 1, memory blocks are used for every array. Although a very small design is achieved, the number of cycles is 644. In solution 2 where all arrays are mapped to register files, the number of cycles is 294, which is already much faster than the 434 cycles in a multiplier-based counter-part. The design solutions 3 to 4 meet the cycle constraint. In solution 4, a top level pipelining is designed after the L2.1 and L2.2 are unrolled. The total latency is only 42 cycles. But this fully pipelined architecture demands around $3\times$ hardware resource for CLBs and DFFs. Overall, solution 3 of the SMU-based design represents the most area/time efficient architecture, giving $4\times$ speedup over the most area constrained architecture in solution 1 with similar CLBs. It also gives $3\times$ saving in CLBs over the fully pipelined architecture in solution 4. Moreover, all the designs except solution 1 in the SMU-based architecture has 0 ASIC multipliers. Compared with the multiplier-based architecture, the superiority of the SMU-based VLSI architecture is obvious.

VII. Conclusion

In this paper, we propose a novel low power and low complexity SoC architecture for multi-stage adaptive interference cancellation for CDMA systems. The Parallel-Residue-Compensation architecture which avoids the direct interference cancellation is optimized to reduce the redundant computations for efficient VLSI design. The CMV is proposed to combine with clock gating for dynamic power management of the SoC architecture. Efficient VLSI architectures are designed based on combinational logic circuits to avoid the usage of dedicated ASIC multipliers. The SoC design space is explored by using a Catapult C HLS design methodology, which leads to area/time efficient architecture. The VLSI architectures demonstrate efficient hardware resource usage and significant power saving by meeting the real-time requirements.

Acknowledgments

We are very grateful to Dr. Behnaam Aazhang for his valuable comments. Dr. Cavallaro was supported in part by Nokia Corporation, Texas Instruments Inc., the Texas Advanced Technology Program under grant 1999-003604-080 and by NSF under grant ANI-9979465.

References

[1] M. K. Varanasi, B. Aazhang, "Multistage detection in asynchronous code-division multiple-access communications", *IEEE Trans. Communications*, vol. 38, pp. 509-519, Apr. 1990.

[2] M. K. Varanasi, B. Aazhang, "Near-optimum detection in synchronous code-division multiple-access systems", in *IEEE. Trans. Communications*, vol. 39, pp. 725-736, 1991.

[3] G. Xu, J. R. Cavallaro, "Real-time implementation of multistage algorithm for next generation wideband CDMA systems", *Proc. ASPA, IX, SPIE*, vol. 3807, Denver, CO, pp. 62-73, July 1999.

[4] Q. Sun, D. C. Cox, "A pipelined multi-stage parallel interference canceller for CDMA with realistic channel estimation", *IEEE Wireless Communications and Networking Conference*, No. 1, pp. 294-298, March 2002.

[5] M. J. Juntti, B. Aazhang, J. O. Lilleberg, "Iterative implementation of linear multiuser detection for dynamic asynchronous CDMA systems", *IEEE Trans. Communications*, vol. 46, pp. 503-508, Apr. 1998.

[6] D. Divsalar, M. K. Simon, D. Raphaeli, "Improved parallel interference cancellation for CDMA", *IEEE Trans. Communications* vol. 46, pp. 258-268, Feb. 1998.

[7] N. Correal, R. M. Buehrer, B. D. Woerner, "A DSP-based DS-CDMA multiuser receiver employing partial parallel interference cancellation" , *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 17, pp. 613-630, Apr. 1999.

[8] G. Xue, J. Weng, T. L. Ngoc and S. Tahar, "Adaptive multistage parallel interference cancellation for CDMA", *IEEE Journal on Selected Areas in Communications*, vol. 17, pp. 1815-1827, Oct. 1999.

[9] Y. Guo, G. Xu, D. McCain, J. Cavallaro, "Rapid scheduling of efficient VLSI architectures for next-generation HSDPA wireless system using Precision-C synthesizer", *Proc. IEEE Intl. Workshop on Rapid System Prototyping'03*, San Diego, CA, pp. 179-185, June 2003.

[10] Y. Guo, J. Zhang, D. McCain, J. R. Cavallaro, "Scalable FPGA architectures for LMMSE-based SIMO chip equalizer in HSDPA downlink", *IEEE Asilomar Conference on Signals, Systems and Computers*, vol. 2, pp. 2171-2175, Monterey, CA, Nov. 2003.

[11] Y. Lee, V. K. Jain, "VLSI architecture for an advanced DS/CDMA wireless communication receiver", *Proc. of IEEE International Conference on Innovative Systems in Silicon*, pp. 237-247, Oct. 1997.

[12] S. Das, C. Sengupta and J. R. Cavallaro, "Hardware design issues for a mobile unit for next generation CDMA systems", *Proc. SPIE Conf. Advanced Signal Processsing: algorithms, architectures and implementations*, vol.3461, San Diego, CA, July, 1998.

[13] S. Rajagopal, B. A. Jones, J. R. Cavallaro, "Task partitioning wireless base-station receiver algorithms on multiple DSPs and FPGAs", *International Conference on Signal Processing Applications and Technology (ICSPAT)*, Dallas, TX, October, 2000.

[14] S.Kim, K.Kum, W, Sung, "Fixed-point optimization utility for C and C++ based digital signal processing programs", *IEEE Trans. on Circuits and Systems-II: Analog and digital signal processing*, pp. 1455-1464, vol. 45, No.11, Nov. 1998.

[15] A. Evens, A. Siburt, G. Vrchoknik, T. Brown, M. Dufresne, G. Hall, T. Ho and Y. Liu, "Functional verification of large ASICS", *ACM/IEEE Design Automation Conference*, San Francisco, CA, pp. 650 - 655, June 1998.

[16] Y. Guo, D. McCain, J. R. Cavalaro, "Low complexity System-On-Chip VLSI architectures of optimal Parallel-Residue-Compensation for MAI suppression in CDMA systems, *IEEE International Symposium on Circuit and Systems*, vol.4, pp. 77-80, Vancouver, Canada, May 2004.

[17] M. S. Srivastava, A. P. Chandrakasan, R. W. Brodersen, "Predictive system shutdown and other architectural techniques for energy efficient programmable computation" *IEEE Trans. Very Large Scale Integration (VLSI) Systems*, vol. 4, No. 1, pp. 42-55, Mar. 1996.

[18] L. Nenini, G. De Micheli, "Dynamic power management: design techniques and CAD tools", Norwell, MA: Kluwer, 1998.

[19] N. Zervas, D. Soudris, S. Theoharis, C. E. Goutis and A. Thanailakis, "A methodology for the behavioral-level event-driven power management of digital receivers", *IEEE International Symposium on Circuit and Systems*, vol. II, pp. 589-592, Geneva, Switzerland, May, 2000.

[20] E. P. Zwyssig, A. T. Erdogan, T. Arslan, "Low power system on chip implementation scheme of digital filtering cores", *Low power IC design seminar*, Jan, 2001, London, UK.

[21] H. Ohlsson, W. Li, O. Gustafsson, L. Wanhammar, "A low power architecture for implementation of digital signal processing algorithms", *Proceeding of Swedish System-on-Chip Conf.*, Falkenberg, Sweden, Mar. 18-19, 2002.

[22] G. C. Cardarilli, A. Nannarelli, M. Re, "Reducing power dissipation in FIR filters using the residue number system", 43^{rd} *IEEE Midwest Symposium on Circuits and Systems*, vol. 1, pp. 320-323, Lansing (MI), USA, Aug. 2000.

[23] J. Park, W. Jeong, H. Choo, H. M. Meimand, Y. Wang, K. Roy, "High performance and low power FIR filter design based on sharing multiplication", *ISLPED'02*, Monterey, CA, USA, pp. 295-300, Aug. 12-14, 2002.