Efficient VLSI Architectures for Recursive Vandermonde QR Decomposition in Broadband OFDM Pre-distortion

Yuanbin Guo
Nokia Research Center
Irving, TX, 75039

Abstract — The Vandermonde system is used in OFDM pre-distortion to enhance the power efficiency dramatically. In this paper, we study efficient FPGA architectures of a recursive algorithm for the Cholesky and QR factorization of the Vandermonde system. We identify the key bottlenecks of the algorithm for the real-time constraints and resource consumption. Several architecture/resource tradeoffs are studied to find the commonalities in the architectures for a best partitioning. Hardware resources are reused according to the algorithmic parallelism and data dependency to achieve the best timing/area performance in hardware. The architectures are implemented in Xilinx FPGA and tested in Aptix real-time hardware platform with 11348 cycles at 25ns clock rate.

I. INTRODUCTION

High power efficiency is an important requirement for mobile stations. Despite of many attractive features that make OFDM very successful in wideband wireless communication standards, e.g., 802.11a, the high peak-to-average-power-ratio (PAPR) [1] is a major drawback of OFDM systems. Because of the summation feature of IFFT computation, the baseband signal will exhibit very high peak-to-average power ratio. This will cause non-linearity distortion from power amplifier with the in-band distortion and out-of-band spectrum growth. On the other hand, to achieve higher power efficiency, it is desirable to begin with a *nonlinear PA* and use linearization circuits. Several linearization schemes have been proposed in different systems, such as Cartesian loop, feed-forward, LUT-based pre-distortion, adaptive pre-distortion etc [2,3,4]. The choice of schemes depends on the different system requirement in data rate, bandwidth and the PAPR. Because of the extremely high PAPR and broadband in OFDM systems, some of these schemes could not meet the requirement. Also because of aging and shift of temperature, frequency and current in PAs, the linearization should be adaptive. In [4], we analyzed the drawbacks of current schemes and proposed a novel polynomial-based adaptive pre-distorter working effectively in OFDM systems. The algorithm will basically solve a Vandermonde system as $\hat{\mathbf{A}}=\mathbf{X}^H\mathbf{X}\mathbf{I}^{-1}\mathbf{X}^H*\mathbf{Y}$. In this paper, we will focus on the efficient real-time implementation architectures.

The direct solution of the system with matrix inversion by Gaussian elimination has a high complexity at the order of $O(n^3)$. By using the structure of Vandermonde matrix, the propagating algorithm requires $(9n^2+n-10)/2$ multiply and divide operation (MDO) for the solution of the normal equations (NE) plus $n^2/2$ multiplications to compute the mean-square error during the recursion [6, 7]. By using the principle of Levinson, the recursive algorithm proposed in [6] reduced

the complexity to $3n^2+9n+3$ MDO. An extra feature is the minimized mean squared error (MMSE) is available as an output in each recursion of polynomial order. The avoidance of matrix inversion makes it attractive for real-time implementation. Despite of several patents in conventional linearization schemes, only few reports could be found for real-time architectures of pre-distortions based on the Vandermonde system. Although [3] discussed the feasibility of DSP and FPGA implementation, it is still a software simulation of matrix inversion in Matlab. For a wireless system, the computation of Vandermonde system is considerably expensive. The exploration for efficient real-time architectures is of great interest both theoretically and practically.

In this paper, we study several efficient architectures in FPGA implementation, e.g. single DSP processor type of architecture; fully parallel manual layout architecture; semi-parallel and pipelined architecture with configurable number of functional units (FU). We identify the key bottlenecks of the algorithm for the real-time constraints and resource consumption. By studying the commonalities in the architectures, we achieved the efficient system partitioning and resource sharing. A Precision-C based High-Level Synthesis (HLS) design methodology is applied to schedule the area/time efficient RTL by studying the architecture tradeoffs. The design is implemented by Xilinx Vitex-II FPGA in a real-time prototyping system.

II. PRE-DISTORTION IN OFDM SYSTEMS

In the transmitter of an OFDM system, a set of information bits $[b_1 \ b_2 \ b_3 \ \dots \ b_M]$ are first mapped into the I/Q channel baseband symbols $\{S_n^{(i,r)}\}$ using a modulation scheme such as phase-shift-keying (PSK) or quadrature-amplitude-modulation (QAM). Then each N symbols are packed into a parallel block $[S_0^{r,i} \ S_1^{r,i} \ \dots \ S_{N-1}^{r,i}]^T$ and OFDM symbols in the time domain over time interval $t \in [0, T_s]$ are generated by IFFT for $k=[1,2,\dots,N]$. The proposed pre-distorter includes two stages [4]. In the estimation stage, we will send a training sequence with sufficient dynamic range to probe the non-linearity. A feedback of the RF output to the base-band with this sequence is sampled and modeled as a P -order polynomial. The system in matrix form is $\mathbf{Y}_t=\mathbf{X}_t\mathbf{A}+\mathbf{W}$, with $\mathbf{Y}_t=[y_1 \ y_2 \ \dots \ y_N]^T$, $\mathbf{W}=[w_1 \ w_2 \ \dots \ w_N]^T \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$, $\mathbf{A}=[\alpha_1 \ \alpha_2 \ \dots \ \alpha_N]^T$ non-linearity coefficients vector and $\mathbf{X}_t=[\mathbf{X}_t^0 \ \mathbf{X}_t^1 \ \mathbf{X}_t^2 \ \dots \ \mathbf{X}_t^P]$ a $N \times (P+1)$ Vandermonde matrix of the $\mathbf{X}_t=[x_1 \ x_2 \ \dots \ x_N]^T$. The LS estimation of the coefficients is,

$$\hat{\mathbf{A}} = [\mathbf{X}_t^H \mathbf{X}_t]^{-1} \mathbf{X}_t^H * \mathbf{Y}_t \quad (1)$$

The principle of pre-distortion [4] is to find a function $g(x)$ before the actual non-linearity to make the overall effect of RF output linear. The inverse non-linearity is shown to be $\hat{\mathbf{A}} = (\mathbf{Y}_t^H \mathbf{Y}_t)^{-1} \mathbf{Y}_t^H \mathbf{X}_t$, where $\mathbf{Y}_t = [\mathbf{Y}_t^0 \ \mathbf{Y}_t^1 \ \mathbf{Y}_t^2 \dots \mathbf{Y}_t^Q]$ is also Vandermonde matrix of the output vector.

III. RECURSIVE QR FACTORIZATION

A fast QR decomposition algorithm is proposed in [6] for Vandermonde systems. The matrix $\mathbf{X}$ is written as a product of an orthogonal matrix $\mathbf{Q}$, with an upper triangular matrix $\mathbf{R}$ ($\mathbf{X} = \mathbf{Q}\mathbf{R}$). Because of the recursive feature in order, the Levinson type algorithm does not require explicit inversion of the coefficient matrix. The recursive feature also makes it suitable for parallel architectures in VLSI implementation. We summarize the key ideas of the algorithm here and make some modifications to facilitate real-time implementation.

The Levinson principle has been used extensively since its invention for structured matrices, especially Toeplitz systems. It states that the j^{th} order solution can be obtained by a linear combination of the forward & backward solutions of the $j-1^{\text{th}}$ order solution. The normal equation states the relation between the polynomial coefficients and the minimum mean square error (MMSE) $E_{A,j+1}$ as in

$$\begin{bmatrix} \mathbf{Y}^T \mathbf{Y} & \mathbf{Y}^T \mathbf{X}_{j+1} \\ \mathbf{X}_{j+1}^T \mathbf{Y} & \mathbf{X}_{j+1}^T \mathbf{X}_{j+1} \end{bmatrix} \begin{bmatrix} 1 \\ \mathbf{A}_{j+1} \end{bmatrix} = \begin{bmatrix} E_{A,j+1} \\ \mathbf{0}_{j+1} \end{bmatrix} \quad (L.1)$$

Using a guess procedure, the $(j+1)^{\text{th}}$ order solution is approximated by the forward estimation of the j^{th} order solution.

$$\begin{bmatrix} 1 \\ \mathbf{A}_{j+1} \end{bmatrix} = \begin{bmatrix} 1 \\ \mathbf{A}_j \\ 0 \end{bmatrix} + \mu_{j+1} \begin{bmatrix} 0 \\ \mathbf{F}_j \\ F_{j,0} \end{bmatrix} \quad (L.2)$$

The forward estimation $\mathbf{F}_j$ is also formed similarly with a relation with forward MMSE $E_{F,j}$ as

$$\mathbf{R}_{j+1} \begin{bmatrix} \mathbf{F}_j \\ F_{j,0} \end{bmatrix} = \begin{bmatrix} \mathbf{0}_j \\ E_{F,j} \end{bmatrix} \quad (L.3)$$

where $\mathbf{R}_{j+1} = \mathbf{X}_{j+1}^T \mathbf{X}_{j+1}$ is a generalized self covariance matrix for the Vandermonde matrix of the $(j+1)^{\text{th}}$ order. Using L.2 and minimizing w.r.t. parameter μ_{j+1} , we will obtain a simplified form for L.1 in (2×2) form as

$$\begin{bmatrix} E_{A,j} & \mathbf{Y}^T \mathbf{X}_{j+1} \begin{bmatrix} \mathbf{F}_j \\ F_{j,0} \end{bmatrix} \\ \begin{bmatrix} \mathbf{x}_j^T \mathbf{Y} & \mathbf{x}_j^T \mathbf{X}_j \end{bmatrix} \begin{bmatrix} 1 \\ \mathbf{A}_j \end{bmatrix} & E_{F,j} \end{bmatrix} \begin{bmatrix} 1 \\ \mu_{j+1} \end{bmatrix} = \begin{bmatrix} E_{A,j+1} \\ 0 \end{bmatrix}. \quad (L.4)$$

Thus we can solve L.2 with μ_{j+1} and forward MMSE $E_{A,j}$. By combining the recursive estimation of both the forward and

backward estimation, we finally get a recursive algorithm with the j^{th} iteration as summarized in:

$$\Delta_{g,j-1} = (\mathbf{r}_{j,2j-1})^T \mathbf{G}_{j-1} \quad (R.1)$$

$$\begin{bmatrix} \mathbf{F}_j \\ F_{j,0} \end{bmatrix} = \begin{bmatrix} \mathbf{F}_{j-1} \\ F_{j-1,0} \\ 0 \end{bmatrix} - \frac{E_{f,j-1}}{\Delta_{g,j-1}} \begin{bmatrix} 0 \\ \mathbf{G}_{j-1,0} \\ \mathbf{G}_{j-1} \end{bmatrix} \quad (R.2)$$

$$E_{f,j} = (\mathbf{r}_{j,2j})^T \mathbf{F}_j \quad (R.3)$$

$$\begin{bmatrix} \mathbf{G}_{j,0} \\ \mathbf{G}_j \end{bmatrix} = \begin{bmatrix} \mathbf{G}_{j-1,0} \\ \mathbf{G}_{j-1} \\ 0 \end{bmatrix} - \frac{\Delta_{g,j-1}}{E_{f,j}} \begin{bmatrix} \mathbf{F}_j \\ F_{j,0} \end{bmatrix} \quad (R.4)$$

$$\Delta_{A,j} = \begin{bmatrix} \mathbf{x}_j^T \mathbf{Y} & \mathbf{x}_j^T \mathbf{X}_j \end{bmatrix} \begin{bmatrix} 1 \\ \mathbf{A}_j \end{bmatrix} \quad (R.5)$$

$$\mu_{j+1} = -\Delta_{A,j} / E_{f,j} \quad (R.6)$$

$$\begin{bmatrix} 1 \\ \mathbf{A}_{j+1} \end{bmatrix} = \begin{bmatrix} 1 \\ \mathbf{A}_j \\ 0 \end{bmatrix} + \mu_{j+1} \begin{bmatrix} 0 \\ \mathbf{F}_j \\ F_{j,0} \end{bmatrix} \quad (R.7)$$

$$E_{A,j+1} = E_{A,j} + \mu_{j+1} \Delta_{A,j} F_{j,0} \quad (R.8)$$

where $\mathbf{G}_{j-1}$ is the matrix of for the computation of the forward estimation of $\mathbf{F}_j$ and $\mathbf{r}_{j,2j}$ is a vector which contains the j^{th} to $2j^{\text{th}}$ independent coefficients of covariance matrix $\mathbf{R}$. This algorithm does not require the explicit computation of the inverse of the coefficient matrix.

IV. VLSI IMPLEMENTATION ISSUES

4.1. Design Methodology

A high-level software technology such as General Purpose Processor or DSP is more flexible to program for many applications. However, they are not efficient enough in speed for many real-time applications. Although ASIC is compact and cheap when the product volume is large, it is not easy to study the architectures tradeoffs. FPGA provides programmability and the flexibility to study several area/timing tradeoffs in hardware architecture by applying the intrinsic algorithmic parallelism. The FPGA net list can be mapped to an ASIC VLSI design for mass production. We use Precision-C and HDL designer based design flow shown in Fig. 1 for our study of efficient architectures. Precision-C is an RTL scheduler from Mentor Graphics. It can assign the number of FUs according the time/area constraints. We start from a floating-point algorithm in matlab. Then we will build a C/C++ test bench to model the exact behavior of the algorithm in a real system. Using some special design styles, we convert the algorithm to Tsunami compatible version. Precision-C can help study the data dependency of the algorithm. We add both time and area constraints for the design and Tsunami will schedule the solutions for efficient architectures according to the

constraints. By studying the parallelism both within Precision-C and offline, many of the functional units are reused in the computational cycles. RTL output is generated and imported into HDL designer. In HDL designer, the arrays are mapped into memory blocks. Coregen will generate Xilinx IP cores for the RAM/ROM blocks as well as the pipelined dividers. After simulation in ModelSim, Spectrum Exemplar is used to do the synthesis and Xilinx Place & Route tools are used to generate the gate-level netlist. It is finally verified in a real-time configurable FPGA prototyping system from Aptix Inc.

Fig. 1. Design flow for the Tsunami architecture scheduling.

4.2. Real time requirement

Since the power amplifier property varies with time due to aging, temperature changes, supply voltage variations and the power control, the pre-distortion requires adaptive updating. The real time requirement determines the architecture tradeoffs. The real-time requirement is two-fold: 1). The update of the polynomial coefficients based on the changing of non-linearity. For a broadband PA, update rate within several *ms* is desirable based on measurement [3]. But considering the shift of frequency, fast update is in good demand. The interpolation requires much less training data than LUT-based schemes and can achieve adaptive updating. Moreover FPGA can work in much lower clock rate and consumes less power than DSP. Once the coefficients are updated, the computation units can be shut down to saves computation power. 2). Real-time generation of the actual pre-distorted signal with the captured non-linearity is critical for the date rate. Higher speed is favorable to higher data rate and wider bandwidth. With this knowledge in mind, we will derive different architectures with either different pipelining or resource sharing tradeoffs.

Fig. 2. Data dependency graph within an iteration.

4.3. Data dependency

The data dependency determines the algorithmic parallelism and possible resource sharing opportunities. In principle, two independent computations can be processed in parallel while two dependent computations must proceed in serial after the first result is ready. On the other hand, two serial computations can share the common resource or such as memory, or expensive computations such as multiply, divide. The data dependency during iteration is identified in Fig. 2. The data with an arrow ending will depend on the result of the data in the previous paths in the graph. Also highlighted are the three divisions in the iteration because the divide is much more expensive in both area and timing. A typical divider will take 1000 LUTs and 16 cycles. By studying the data dependency and the time/area tradeoff, we can employ the parallelism and resource sharing to the most extent.

V. SYSTEM PARTITIONING

Based on the data dependency graph, we partition the system into two major blocks: the initialization stage and the recursion stage. We also identify the critical path in terms of the system latency from this graph. Fig. 3 shows the major partitioning of the system. The data vectors $\mathbf{X}$ and $\mathbf{Y}$ are input into the INIT block using RAM block generated by Xilinx CoreGen tool. Although dual-port rams will usually be used for inter-process communication, since the $\mathbf{X}$, $\mathbf{Y}$ are only read, we can use a MUX to multiplex the ADDR_RD and ADDR_WR for a single RAM block because it is cheaper than dual port ram.

1). *Initialization*: The operations in the INIT block are shown in the picture. Since $\mathbf{R}_j$ and $\mathbf{x}_j^T \mathbf{Y}$ in (R.1), (R.3), (R.5) are used in all iterations with no dependency from other variables, we init the values of them before the iteration. The architecture will be discussed in more detail.

Fig. 3. VLSI implementation block diagram of the recursive QR factorization.

2). *Recursive Loop*: Equations (R.1, R.5) can be processed in parallel in each iteration given the initial values of $\mathbf{R}_{j+1}$ and $\mathbf{Y}^H \mathbf{X}^j$. By defining an intermediate variable $\text{TE} = 1/E_{f,j}$, and $\gamma_{j+1} = \Delta_{g,j-1} * \text{TE}$, we can reduce the explicit divisions to 2 in each iteration. With pipelined divider, the two divides in R.2 and R.4 can be pipelined. R.4, R.7, R.8 can be in parallel while R.4 can also be in a pipeline with R.2 since they have similar computation pattern for the j^{th} order vector. The computation latency in the j^{th} iteration can be reduced by about half with $4j+4$ from $6j+10$ multiplications. For a P order polynomial, the complexity is reduced to $O(2P^2+2P)$ from $O(3P^2+7P)$ for multiplications and to 17P for division cycles. Moreover, by comparing the MSE with a threshold, we can terminate the iteration when the MSE converges to a threshold.

3). *Pipelined Pre-distortion*: In the pre-distortion stage, $y[k] = \sum \alpha_i x^i[k]$. Each transmitted sample needs to be processed according to the sampling rate. So the accumulation architecture is not fast enough. Fig. 4 shows a pipelined architecture that applies the delay-line for a nested filter type of architecture. In this architecture, we only need $2P$ multipliers for the $\text{pow}(P)$ computation and P adders for the accumulation.

Although this will consume more areas, it is still acceptable because typically the algorithm will converge at the order $P < 10$. All functional units are fully utilized in a pipelined style, achieving the highest throughput.

4). *Architecture Area/time tradeoffs*: With the partitioning, the major computation load is the INIT of $\mathbf{R}$ matrix and the $\mathbf{Y}^H \mathbf{X}$ cross correlation for a polynomial if $P \ll N$, which is always true in reality. A naïve implementation of the $\mathbf{R}$ in the row order has prohibitive complexity of $N * P * (P-1)/2$ MULTS for $\mathbf{X}$ matrix and another $N * P^2$ for the $\mathbf{X}^T \mathbf{X}$ matrix computation. The storage is also considerably large. However, it is shown that $\mathbf{R}_j$ is a Hankel matrix

$$\mathbf{R}_{j+1} = \mathbf{X}_{j+1}^T \mathbf{X}_{j+1} = \left[\sum_{k=0}^{N-1} x_k^{i+j} \right]_{i,j}$$

which has only $2j+1$ independent entries $r_l = \sum_{k=0}^{N-1} x_k^{l+j}$ ($l=0 \rightarrow 2j$). This dramatically reduced the computation complexity.

Even though, there still exist several architecture tradeoffs for further optimization. If we compute the $\text{pow}(x[k], 0 \rightarrow l)$ for $\mathcal{I}_l^j$ and don't keep the intermediate value, we need to re-compute the all of them when we compute $\mathcal{I}_l^{j+1}$. This obviously wastes the computation. So by applying a **tempXJ** RAMblock with N entries and compute the $\mathcal{I}_l^j$ in a pattern as

$$\begin{aligned} \text{tmpXJ}[k]^l &= \text{tmpXJ}[k]^{l-1} * x[k] \quad (k=0 \rightarrow N-1; l=1 \rightarrow 2j) \\ \mathbf{R}[l]_{l=0 \rightarrow 2j} &= \sum_{k=0}^{N-1} \text{tmpXJ}[k]^l \end{aligned}$$

Because $\Psi^l = \mathbf{Y}^H \mathbf{X}^l$ also depends on the $x^l[k]$, by merging the two matrix computations, we not only save many computation, but also save many memory accesses. The comparison of the two schemes is summarized in Table. 1 and Table. 2.

Section	Mult #	Add #	RDMEM	WRMEM
RR alone	$N(2P^2+P)$	$2NP$	$N(2P^2+2P)$	$N2P$
YX alone	$N(P^2+1/2)$	NP	$N(P^2+P)$	NP
Total	$3NP^2$	$3NP$	$3NP^2$	$3NP$

Table. 1. Operations for separated initialization with no tempXJ ram.

Section	Mult #	Add #	RDMEM	WRMEM
Init ($j=0,1$)	N	$3N$	$2N$	$N+6$
Sharing ($j=2 \sim P$)	$2N(P-1)$	$2N(P-1)$	$3N(P-1)$	$NP-N+2P$
($j=P+1 \sim 2P$)	NP	NP	$2NP$	$P+NP$
Total	$N(3P-1)$	$N(3P-1)$	$4NP$	$2NP+3P$

Table. 2. Operations for merged initialization with tempXJ ram.

Fig. 4. Pipelined architecture for pre-distortion.

There are also many tradeoffs for the **pow and accumulation** architecture. The area-constrained VLSI architecture is shown in Fig. 5 (a). It is similar to a processor type of architecture by repeating the computation within the loop structure with the same FU (multiplier and adders). It achieves the best resource sharing because of the sequential operation. We layout the FUs and use the Finite State-Machine (FSM) and logic design to generate the address signals for memory access and control signals for MUX with only one multiplier and adder for the whole system. However, this area-constrained architecture may take too much time. To achieve the highest throughput, a time-constrained architecture is shown in Fig. 5 (b). Because the computations of the x^j for N samples are independent, we can achieve full parallelism and pipelining by laying out identical Process Entity (PE) for each sample. This requires at least N multipliers and all storages mapped to register file because memory access can stall the pipeline. The area is too big and the achieved speed is more than enough. In an area/time efficient architecture as shown in Fig. 5 (c), we duplicate PEs for the smallest size by still meeting the time requirement. We divide the memories (X , Y , $\mathbf{tmpXJ}$) according to the unrolled PE number. This architecture acts like multiple DSP processors in parallel according to the speed requirement.

Because of the limited space, many important implementation issues such as the complex logic design and pipelining details as well as Finite State Machine (FSM) are not included in this paper.

The performance gain of the pre-distorter is depicted in Fig. 6. (Output Backoff) OBO is defined as the maximum output power (saturation power) over the mean power of the

transmitted signal as “ $\text{OBO(dB)} = 10 \log_{10}(P_{\text{maxo}}/P_{\text{avo}})$ ”. It determines the operating point of the amplifier. With pre-distortion, the BER curve is very close to the linear case with $\text{OBO}=7\text{dB}$, which is much better than pure non-linearity at the same OBO level.

VI. CONCLUSION

In this paper, we study efficient FPGA architectures of a recursive QR factorization for Vandermonde system in OFDM digital adaptive pre-distorter. The architectures utilize the arithmetic parallelisms and complex logic design for pipelining and resource sharing. It demonstrates the efficient area/time tradeoff by meeting real-time requirement. The architecture with partial optimizations is implemented in Xilinx FPGA real-time platform with 11348 cycles at 25 ns clock rate for $N=100$, $P=10$, which means a latency of 283 μs . Compared with a preliminary solution with the same size at 41205 cycles (1ms), this demonstrates the potential of the efficient architectures for adaptive pre-distortion.

REFERENCES

- [1] V. Tarokh, H. Jafarkhani, “On the computation and reduction of the peak-to-average power ratio in multicarrier communications”, *IEEE Trans. on Comm.*, Vol. 48, No.1, Jan. 2000.
- [2] D. Lottici, V. Reggiannini, “Nonlinear predistortion of OFDM signals over frequency-selective fading channels”, *IEEE Trans. Comm.*, Vol.49, May 2001.
- [3] Deniel Eric Johnson, “Adaptive digital pre-distortion with applications for LMDS systems”, MS thesis, Virginia Polytech, 2000.
- [4] Y. Guo, J. Cavallaro, “A Novel adaptive pre-distorter using LS estimation of SSPA non-linearity in mobile OFDM systems”, *ISCAS 2002*, pp. 453-456, Vol. 3, 2002.
- [5] Y. Guo, J. Cavallaro, “Enhanced power efficiency of mobile OFDM radio using pre-distortion and post-compensation”, *Proceeding of VTC 2002*, Vancouver, Sept. 21-24, 2002.
- [6] M. J. Porsani, T. J. Ulrych, “Levinson-type algorithms for polynomials fitting and for Cholesky and Q factors of Hankel and Vandermonde matrices”, *IEEE Trans. Signal Proc.* Vol. 43, no. 1, pp. 63-70, Jan. 1995.
- [7] H. Lev-Ari, T. Kailath and J. M Cioffi, “Adaptive recursive least squares lattice and transversal filters for continuous-time signal processing”, *IEEE Trans. On Circuits and Systems.* Vol. 39, no. 2, pp. 81-88, Feb. 1992.

Fig. 5 (a). Area-constrained VLSI architecture.

Fig. 5 (b). Time constrained VLSI architecture.

Fig. 5 (c). Area/time efficient VLSI architecture

Fig. 6. Bit error rate performance of the digital pre-distortion.