Untimed-C based SoC Architecture Design Space Exploration for 3G and Beyond Wireless Systems

Yuanbin Guo, Dennis McCain
Nokia Research Center
Irving, TX
Email: {Yuanbin.Guo, Dennis.McCain}@nokia.com

***Abstract*— In this paper, we propose an un-timed C/C++ level verification methodology that integrates key technologies for truly high-level VLSI modelling to keep pace with the explosive complexity of SoC designs in the 3G and beyond wireless communications. A Catapult C/C++ based architecture scheduler transfers the major workload to the algorithmic C/C++ fixed-point design. Case study is given to explore the VLSI design space extensively for various types of computational intensive algorithms in MIMO-CDMA systems, such as a MIMO equalizer to avoid the Direct-Matrix-Inverse. Extensive time/area tradeoff study is enabled with different architecture and resource constraints in a short design cycle. Architecture efficiency and productivity are improved significantly, enabling truly rapid prototyping for the 3G and beyond wireless systems.**

I. Introduction

MIMO (Multiple Input Multiple Output) technology [1][2] using multiple antennas at both the transmitter and receiver sides is leading to MIMO-CDMA [7], MIMO-OFDM [8] etc. as enabling techniques in future 3G and beyond wireless systems. Wireless communication is entering a new era to provide broadband multimedia services and ubiquitous networking via mobile devices. This is pushing both advanced algorithms and hardware technologies for receiver architectures for much higher data rates than current systems.

To achieve better performance, much more complicated signal processing algorithms are required. This gives tremendous challenges for real-time hardware implementation [3]. Although System-On-Chip (SoC) architectures offer more parallelism and flexibility for utilizing low-level silicon resources than DSP processors, the conventional gap between the algorithm researchers and the hardware teams is resulting in many algorithms that are not realistic for real-time implementation and lack insights into the potential architectures. Moreover, there are many area/time/power tradeoffs in the VLSI architectures. Extensive study of the different architecture tradeoffs provides critical insights into implementation issues that may arise during the product development process and allows designers to identify the critical performance bottlenecks in meeting real-time requirement. However, this type of SoC design space exploration is extremely time consuming because of the current standard trial-and-optimize approaches using hand-coded VHDL/Verilog or Graphical schematic design tools [4]. The System-C[5] based high-level abstraction is also not intuitive to system engineers and requires very detailed hardware and timing specification in the language. Extensive tradeoff analysis is very difficult for the manual parallelism/pipelining design and the still low-level hardware specifications.

For truly high-level VLSI modelling to keep pace with the explosive complexity of SoC designs in the MIMO mobile devices, an un-timed C/C++ level verification methodology that integrates key technologies is proposed in this paper. A Catapult C based architecture scheduler is applied to explore the VLSI design space extensively for various types of computational intensive algorithms in HSDPA, MIMO-CDMA systems. The major workload is transferred to the algorithmic C/C++ fixed-point design and high-level architecture scheduling. Extensive time/area tradeoff study is enabled with different architecture and resource constraints in a short design cycle. Synthesizable RTL is generated directly from a C/C++ level design and imported to the graphical tools for module binding.

We focus the case study on the FPGA evaluation of an FFT-based MIMO equalizer that avoids the Direct-Matrix-Inverse [7]. The key factors for optimization of the area/speed in loop unrolling, pipelining and the resource multiplexing are identified. Multi-level pipelining/parallelism are explored extensively to search for the most efficient VLSI architecture. Architecture efficiency and much improved productivity are achieved by reducing the design cycle by $50\% - 70\%$, enabling truly rapid prototyping for the computational extensive signal processing algorithms. This significantly shortens the technology transition time from algorithm to reality and reduces the risk in product development for 3G and beyond wireless systems.

II. MIMO-CDMA Downlink Receiver

A. Background

Known as D-BLAST [1] and a more realistic algorithm as V-BLAST [2] with better performance/complexity tradeoff for real-time implementation, the original MIMO spatial multiplexing was proposed for narrow band and flat fading channels. In a multipath fading channel, the orthogonality of the spreading codes is destroyed, introducing both the Multiple-Access-Interference (MAI) and the Inter-Symbol-Interference (ISI). The conventional Rake receiver [9] could not provide acceptable performance because of the very short spreading gain in the Multi-Code CDMA downlink to support high rate data services. LMMSE (Linear-Minimum-Mean-Square-Error)-based chip equalizer is promising to restore

the orthogonality of the spreading code and suppress both the ISI and MAI [9] in traditional single antenna systems. However, this involves the inverse of a large correlation matrix with $\mathcal{O}((NF)^3)$ complexity for MIMO systems, where N is the number of Rx antenna and F is the channel length. Traditionally, the implementation of equalizer in hardware has been one of the most complex tasks for receiver designs. The MIMO extension gives even more challenges for real-time hardware implementation.

In this section, we first present an FFT-based fast algorithm for the tap solving by approximating the block Toeplitz structure of the correlation matrix with a block circulant matrix to avoid the matrix inverse. The direct matrix inverse problem for the large covariance matrix is reduced to many parallel FFT/IFFT operations and the inverse of some much smaller sub-matrices. Part of the algorithm will be presented in the proceeding of [7]. This algorithm reduces the complexity order to $\mathcal{O}(NF\log_2(F))$, which makes the real-time implementation much easier.

B. System Model

In the MIMO-CDMA system using spatial multiplexing and M Tx antennas and N Rx antennas where usually $M \leq N$, multiple spreading codes are assigned to a single user to achieve high data rate. First, the high data rate symbols are demultiplexed into KM lower rate substreams, where K is the number of spreading codes for data transmission. The substreams are broken into M groups, where each substream in the group is spreaded with a spreading code of spreading gain G. The groups of substreams are then combined and scrambled with long scrambling codes and transmitted through the m^{th} Tx antenna. The chip level signal at the m^{th} transmit antenna is given by

$$d_m(i) = \sum_{k=1}^{K} s_m^k(j)c_m^k(i) + s_m^P(j)c_m^P(i) \tag{1}$$

where j is the symbol index, i is the chip index and k is the index of the composite spreading code. $s_m^k(j)$ is the j^{th} symbol of the k^{th} code at the m^{th} substream. In the following, we focus on the j^{th} symbol and omit the symbol index for notation simplicity. $c_m^k(i) = c_k(i)c_m^{(s)}(i)$ is the composite spreading code sequence for the k^{th} code at the m^{th} substream where $c_k(i)$ is the user specific Hadamard spreading code and $c_m^{(s)}(i)$ is the antenna specific scrambling long code. $s_m^P(j)$ denotes the pilot symbols at the m^{th} antenna. $c_m^P(i) = c^P(i)c_m^{(s)}(i)$ is the composite spreading code for pilot symbols at the m^{th} antenna. The received chip level signal at the n^{th} Rx antenna is given by

$$r_n(i) = \sum_{m=1}^{M}\sum_{l=0}^{L_{m,n}} h_{m,n}(l)d_m(i-\tau_l) + z(i) \tag{2}$$

where the channel is characterized by a channel matrix with elements taken from the channel coefficients between the m^{th} Tx antenna and the n^{th} Rx antenna.

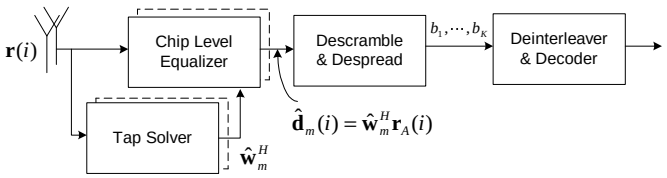

Fig. 1. The block diagram of the LMMSE chip equalizer.

By packing the received chips from all the receive antennas in a vector $\mathbf{r}(i) = [r_1(i), \cdots, r_n(i), \cdots, r_N(i)]^T$ and collecting the $L_F = 2F+1$ consecutive chips with center at the i^{th} chip from all the N Rx antennas, we form a signal vector as $\mathbf{r}_A = [\mathbf{r}(i+F)^T, \cdots, \mathbf{r}(i)^T, \cdots, \mathbf{r}(i-F)^T]^T$. In the vector form, the received signal can be given by

$$\mathbf{r}_A(i) = \sum_{m=1}^{M_t} \mathbf{H}_m\mathbf{d}_m(i) + \mathbf{z}(i) \tag{3}$$

where $\mathbf{H}_m$ is a block Toeplitz matrix constructed from the channel coefficients. The multiple receive antennas' channel vector is defined as $\mathbf{h}_m(l) = [h_{m,1}(l), \cdots, h_{m,n}(l), \cdots, h_{m,N}(l)]^T$. The transmitted chip vector for the m^{th} transmit antenna is given by $\mathbf{d}_m(i) = [d_m(i+F), \cdots, d_m(i), \cdots, d_m(i-F-L)]^T$.

C. Chip Level Equalizer

Linear MMSE based chip level equalization has been one of the most promising receivers in the single-user CDMA downlink. Chip equalizer estimates the transmitted chip samples by a set of linear FIR filter coefficients as

$$\hat{\mathbf{d}}_m(i) = \hat{\mathbf{w}}_m^H\mathbf{r}_A(i). \tag{4}$$

It is well known that the LMMSE chip equalizer is given by

$$\begin{aligned}\hat{\mathbf{w}}_m^{opt} &= \arg\min_{\mathbf{w}_m} \mathrm{E}[||\mathbf{d}_m - \hat{\mathbf{w}}_m^H(i)||^2] \\ &= \sigma_d^2(i)\mathbf{R}_{rr}(i)^{-1}\hat{\mathbf{h}}_m.\end{aligned} \tag{5}$$

where the correlation matrix and the channel estimation are given by the time-average with ergodicity assumption as

$$\mathbf{R}_{rr}(i) = \mathrm{E}[\mathbf{r}_A(i)\mathbf{r}_A^H(i)] = \frac{1}{N_B}\sum_{i=0}^{N_B-1} \mathbf{r}_A(i)\ \mathbf{r}_A^H(i) \tag{6}$$

where N_B is the length for the time average. The channel coefficients are estimated as

$$\hat{\mathbf{h}}_m(i) = \mathrm{E}[\mathbf{r}_A(i)\mathbf{d}_m^H(i)] \tag{7}$$

using the pilot symbols. In the HSDPA standard, about 10 % of the total transmit power is dedicated to the Common Pilot Channel (CPICH). This will provide accurate channel estimation. Fig. 1 gives the simple model of the LMMSE chip equalizer in the MIMO-CDMA downlink. The direct matrix inverse has high complexity at the order of $\mathcal{O}((NF)^3)$. This complexity is too high for real-time implementation. In [7], we have proposed a FFT-based fast algorithm to avoid the DMI with complexity of $\mathcal{O}(NF\log_2(NF))$. In this paper,

we implement this algorithm using the Catapult C based methodology. In the following, we present the design space exploration.

III. Classical Implementation Technologies

The most fundamental method of creating hardware design for an FPGA or ASIC is by using industry-standard hardware description language (HDL), such as VHDL or Verilog [4], based on dataflow, structural or behavioral models. However, this design method is very low-level for system engineers to understand and highly based on the off-line logic modelling. For a very complex design like the CDMA system, the design and troubleshooting can be very difficult. Graphical schematic design tools such as Hardware Design System (HDS) from Cadence or HDL designer from Mentor Graphics are more intuitive. However, the design is still manual and the intrinsic architecture tradeoffs need to be studied offline. It is not easy to change a design dramatically once the hardware architecture is laid out.

The High-Level-Synthesis methodology [10] [12] [11] [13] provides a bridge by offering rapid system prototyping to the SoC design. Some C/C++ level RTL tools such as System-C [5] and Handel-C [6] attempt to combine a high-level of abstraction with the ability to generate synthesizable RTL. However, these design flows requires detailed knowledge of hardware implementation such as clocking, control logic, resource allocation etc. Moreover, detailed knowledge of hardware components is still required because all of the hardware components need to be synthesizable for a hardware implementation. They are still not intuitive to system engineers to understand and the detailed hardware specification in the language requires the designer to manually decide the architectural parallelism and pipelining. Manual optimization makes the tradeoff study in terms of time and area of the design difficult to evaluate, especially when the re-timing is critical for high-speed designs.

IV. Un-timed C/C++ SoC Design Space Exploration

Scheduling and allocation are two important tasks in hardware or software synthesis of DSP systems. They are both interrelated and dependent on each other and are among the most difficult problems of high-level synthesis. Scheduling involves assigning every node of the Data-Flow-Graph (DFG) to control time steps. Control time steps are the fundamental sequencing units in synchronous systems and correspond to clock cycles. Resource allocation is the process of assigning operations to hardware with the goal of minimizing the amount of hardware required to implement the desired behavior. The hardware resources consist primarily of functional units, memory elements, multiplexes, and communication data paths.

To achieve the goal of efficient verification, we propose the state-of-the-art hybrid design and verification methodologies in this paper for very rapid prototyping of the MIMO SoC architectures, which is out of the reach of simulation. A Catapult C [14] based High-Level-Synthesis (HLS) design flow for modelling, partitioning, verification and synthesis of the complete system is derived. Unlike the HLS methodologies that require the detailed timing specification of the design, Catapult C is an un-timed tool which requires much fewer modifications to the ANSI-C algorithm.

A. Catapult C based High-Level Synthesis Methodology

Catapult C synthesizer is a new RTL design tool optimized for hardware design from Mentor Graphics. It is a true C/C++ level architecture scheduler without requiring timing specification in the C source code. We were one of the first Beta users of the tool and one of the first in the industry to integrate Catapult C in a complete rapid prototyping methodology for advanced wireless communication systems. In the beta stage in 2002, Catapult C was called Tsunami HLS designer. It was then changed to Precision-C in the 2003 production release. The current name was officially released in the ACM Design-Automation-Conference (DAC) 2004 in San Diego California.

It contains the features to integrate domain specific specifications, support composition of models that describe both systems and components. The support for more abstract modelling provides predictive analysis and verification. Synergistic integration of all these technologies in a unified platform to create higher automation will treat the traditional Register-Transfer-Level (RTL) as an assembler language for system-level languages. The VLSI design space exploration is carried out according to the design flow involving major design tools from Catapult C synthesizer, Mentor Graphics Advantage HDL Designer, and other Xilinx implementation tools. The system level VLSI design is partitioned into several subsystem blocks (SB) according to the functionality and timing relationship. The intermediate tasks will include high-level optimizations, scheduling and resource allocation, module binding, and control circuit generation. The final architecture is the RTL that can be represented by a netlist consisting of a network of functional units, registers, multiplexes and buses. The procedure for implementing an algorithm in the SoC hardware includes the following stages described as follows:

1) Algorithm verification in Matlab and ANSI C/C++: In the algorithmic level design, we first use Matlab to verify the floating-point algorithm based on communication theory. The matrix level computations must be converted to plain C/C++ code. All internal Matlab functions such as FFT, SVD, eigenvalue calculation, complex operations etc, need to be translated with efficient arithmetic algorithms to C/C++.
2) Catapult C HLS: RTL output can be generated from C/C++ level algorithms by following some C/C++ design styles. Many FUs (Functional Units) can be reused in the computational cycles by studying the parallelism in the algorithm. We can add both timing and area constraints and Catapult C will schedule efficient architectural solutions according to the specified constraints in Catapult C. The number of FUs is assigned according to the timing/area constraints. Software resources such as registers and arrays are mapped to hardware components and required Finite State Machines (FSM) necessary for

accessing these resources are generated. In this way, we can study several architectural solutions efficiently and achieve the flexibility and productivity of a DSP with the performance of an FPGA.

3) RTL Integration and module binding: In the next step of the design flow, we use HDL designer to import the RTL output generated by Catapult C. A testbench is built in HDL designer corresponding to the C++ testbench and simulated using ModelSim. At this point, several intellectual property (IP) cores might be integrated, such as the efficient cores from Xilinx CoreGen library (RAM/ROM blocks, CORDIC, FIFO, pipelined divider etc) and HDL Module ware components for the test-bench. ModelSim simulation results should match the C fixed-point/integer testbench using the same test vectors to verify the stand-alone algorithm design.
4) Gate level hardware validation: Leonardo Spectrum or Precision-RTL is invoked for gate-level synthesis. Place & Route tools such as Xilinx ISE are used to generate a gate-level bit-stream file. For hardware verification and validation, a configurable Nallatech hardware platform is used. The hardware is tested and verified by comparing the logic analyzer or ChipScope^{TM} probes with the ModelSim simulation. Other goals during synthesis include minimizing the number of memory elements, reducing the power consumption, minimizing the number of buses, incorporating reliability and testability into the design. Xilinx Xpower is utilized to estimate dynamic power consumption and study the power tradeoffs.

V. VLSI Design Space Exploration

In addition to minimizing the circuit area used, the design needs to work within a set time budget. There are many area/time tradeoffs in the VLSI architectures. Extensive study of the different architecture tradeoffs provides critical insights into implementation issues that may arise during the product development process. In this section, we apply the Catapult C HLS methodology to explore the VLSI architecture tradeoffs extensively, which is enabled by allocating different architecture/resource constraints in the Catapult C scheduler. Synthesizable RTL is generated directly from an algorithmic C/C++ fixed-point design, integrated in other key downstream flows and validated in a Xilinx FPGA prototyping platform.

A. Top-Level Partitioning

We emphasize the interaction between architecture, system partitioning and pipelining with these objectives: 1). Implement the equalizer with the minimum hardware resources needed to meet the real-time requirement; 2) obtain an efficient architecture with optimal parallelism and pipelining for the critical computation parts. To explore for an efficient architecture, we elaborate the tasks as the following procedure:

1) Compute the independent correlation elements and form the first block column of circulant by adding the corner elements. Each element is an $(N \times N)$ sub block matrix.

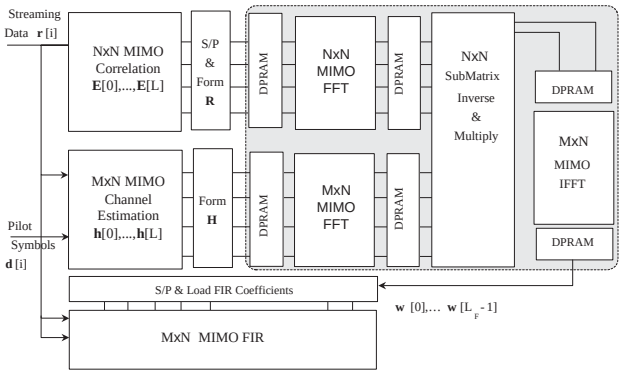

Fig. 2. The block diagram of the VLSI architecture of the FFT-based MIMO equalizer.

2) Take the element-wise FFT of the first block column with the corners added to the block-Toeplitz structure.
3) Compute the dimension-wise FFT of the channel estimation.
4) Compute the inverse of some $(N \times N)$ sub matrix.
5) Compute the matrix multiplication of the sub-matrices inverse with the FFT output of channel estimation coefficients.
6) Compute the dimension-wise IFFT of the multiplication results.

With a timing and data dependency analysis, the top level design blocks for the MIMO equalizer are shown in Fig. 2. The system-level pipeline is designed for better modularity. In the front-end, a correlation estimation block takes the multiple input samples for each chip to compute the correlation coefficients of the first column of $\mathbf{R}_{rr}$. It is made circulant by adding corner to form the block circulant matrix. The complete coefficients are written to DPRAMs and the $(N \times N)$ element-wise FFT module computes computes the frequency domain result of the covariance coefficients.

Another parallel data path is for the channel estimation and the $(M \times N)$ dimension-wise FFTs on the channel coefficient vectors. A sub-matrix inverse and multiplication block takes the FFT coefficients of both channels and correlations from DPRAMs and carries out the computation. Finally a $(M \times N)$ dimension-wise IFFT module generates the results for the equalizer taps $\hat{\mathbf{w}}_m^{opt}$ and sends them to the $(M \times N)$ MIMO FIR block for filtering. To reflect the correct timing, the correlation and channel estimation modules at the front-end will work in a throughput mode on the streaming input samples. The FFT-inverse-IFFT modules in the dotted line block construct the post-processing of the tap solver. They are suitable to work in a block mode using dual-point RAM blocks to communicate the data. The MIMO FIR filtering will also work in throughput mode on the buffered streaming input data.

B. Scalable Architecture for MIMO Covariance Estimation

FPGA provides a good platform to build, verify and prototype SoC designs quickly. There are many area/time/power tradeoffs in the VLSI architectures. Extensive study of the different architecture tradeoffs provides critical insights into

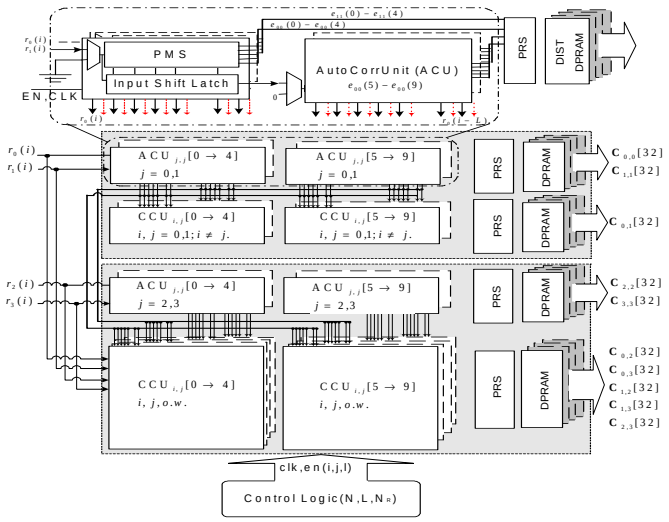

Fig. 3. The throughput mode scalable VLSI architecture for the MIMO covariance matrix estimation.

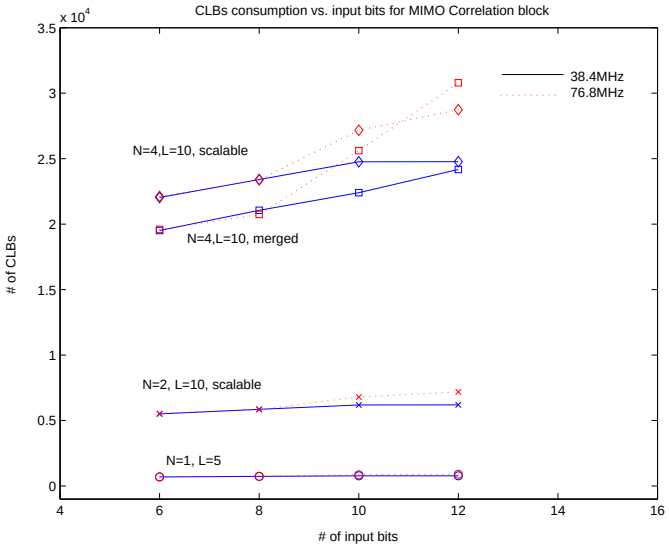

Fig. 4. CLB vs. # of input bits for the covariance estimation block with different architectures.

implementation issues that may arise during the product development process and allows designers to identify the critical performance bottlenecks in meeting real-time requirements.

The covariance coefficients are estimated as in (6). It can be shown that in a standard manual layout architecture, we could apply pipelined multipliers and adder trees in the delayed tap line structure. However, that requires tremendous multipliers which exceeds the resource limit even in a high-end FPGA device. For example, the Xilinx Virtex-II V6000 provides 144 dedicated ASIC multipliers. However, for a 4×4 MIMO receiver with 10 taps, there are at least 160 multiplications per chip period. If we want the design to work in $38.4MHz$ clock rate and $3.84MHz$ chip rate, we have to reuse the multipliers to achieve an efficient resource utilization. However, for this many multipliers, the multiplexing in a manual design is very tedious and time-consuming. Moreover, if the specification is changed, the RTL has to be redesigned completely with almost the same long design cycle.

We apply the Catapult C [14] based HLS methodology to explore the VLSI architecture space extensively in terms of time/area tradeoff, which is enabled with high-level architecture and resource constraints. In Catapult C, we can schedule architectures working in two basic modes according to the real-time behavior of the sub-system: the throughput mode or the block mode. Throughput mode assumes that there is a top-level main loop for the incoming data. The module processes for each input sample periodically, so there is strict limit for the processing time. Block mode processes once after a block of data is ready. The hardware interface will use RAM blocks to map the arrays/vectors in the C codes.

For the covariance estimation module, it is desirable to make it work on the streaming input data. So we design the un-timed fixed-point C source code in the throughput mode. Fig. 3 shows the scalable VLSI architecture for the MIMO covariance matrix estimation. For designs with a different numbers of input bits, we only need to change the word length in C level for only a few variables at the interface. Catapult C can figure out the optimal word length for the internal variables. For the same C source, we can generate dramatically different RTL with different latency and resource utilization by changing the architectural/resource constraints within the Catapult C environment. If we are not satisfied with some of the design specification, we can easily change the source code to reflect a different partitioning for the purpose of scalability. For the MIMO scenario, we can scale the covariance estimation module for different number of antennas. Thus, the same design is configurable to different number of antennas in the system. This scalability provides an approach for shutting down some idle modules so as to save the power consumption in the design, which is essential to mobile devices.

Fig. 4 shows the CLB consumption with different number of input bits for the covariance estimation module with different architectures. Fig. 5 summarizes the usage of the dedicated ASIC multipliers versus the number of input bits for different architectures. The details of the architectures are not explained in this paper. To achieve such an extensive study and verify in the real-time environment in a short time is virtually impossible with the conventional design methodology. However, this demonstrates that we can explore the design specifications with much less cost with the Catapult C based methodology.

The channel estimation has similar types of architecture based on the streaming input data and the pilot symbols. The pilot symbols are stored in the ROM blocks of the receiver. The channel estimation basically computes the cross correlation of the pilot symbols with the input chip samples. So this module is also working on the throughput mode. Because the pilot symbols are known at the receiver and take values only from $\{\pm 1\}$, we do not need to use dedicated multipliers. Instead, we can design the logic with a simple MUX for the correlation involving the pilot symbols. The architecture is shown in Fig.

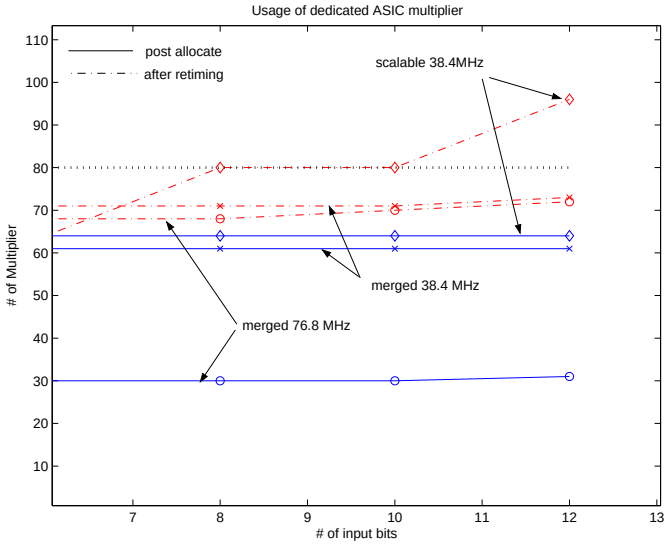

Fig. 5. # of ASIC Multipliers vs. # of input bits for the covariance estimation block with different architectures.

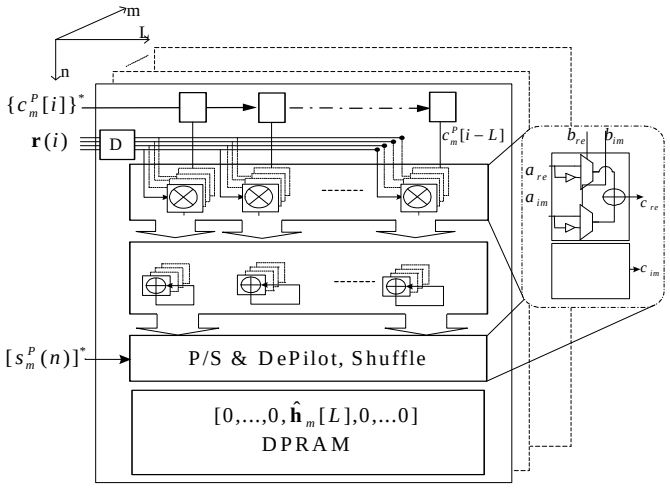

Fig. 6. The scalable VLSI architecture for the MIMO channel estimation using pilot symbols and combinational logic.

6. This simplifies the design and reduces the design area significantly.

C. Design Space Exploration of FFT-Modules

The C source code has a very similar style to the standard ANSI C syntax. The following shows a short example Catapult C code for the well-known radix-2 FFT/IFFT module. There is only minimal effort to modify this ANSI-C code for Catapult C synthesis. The modifications are shown as in *Italics*. First, we need to include the *mc_bitvector.h* to declare some Catapult C specific bit vector types such as the int16, uint2 etc. We first convert the cosine/sine phase coefficients to integer numbers and store them in two vectors that will be mapped to ROM hardware as *cosv* and *sinv*. If we consider the FFT module as the top level of the partitioned Catapult C module, we need to declare the *#pragma design top*. The input and output arrays ar0, ai0 could be mapped to dual-port RAM blocks in hardware. the flag is a signal to configure if it is an FFT or IFFT module. In the core algorithm, there are different levels of loop structure, the stage level, the butterfly-unit level and the actual implementation of the butterfly units. It can be seen that the Catapult C style is almost the same as the ANSI-C style. There is no need to specify the timing information in the source code. Based on the loop structure and the storage hardware mapping, we can specify the different architecture constraints within the Catapult C interface to generate the desired RTL architecture.

```
#include <mc_bitvector.h>
#define NFFT 32
#define LogN 5
const int16 cosv[LogN]={-1024,0,724,946,1004};
const int16 sinv[LogN]={0,-1024,-724,-392,-200};
#pragma design top
void fft32int16(int16 ar0[NFFT],int16 ai0[NFFT],const int16
cosv[LogN],const int16 sinv[LogN], uint1 flag)
{
  short i,j,k,l,le,le1;
  int16 rtmp0,itmp0,ru,iu,r,rw,iw; le=1;
  for(l=1;l ≤ LogN;l++)
  { //stage level
    le1=le; le = le*2;
    ru=1024; iu=0;
    rw=cosv[l-1]; //rw=cos(PI/le1);
    if(flag==0)
    { //forward fft
      iw=sinv[l-1];
    } //iw=-sin(PI/le1);
    else
    { //backward ifft
      iw=-sinv[l-1];
    }

  for(j=0;j <le1;j++)
  {
    for(i=j;i<NFFT;i+=le)
    {// BFU level
      k=i+le1;
      rtmp0=(ar0[k]*ru-ai0[k]*iu)>>10;
      itmp0=(ai0[k]*ru+ar0[k]*iu)>>10;
      ar0[k]=ar0[i]-rtmp0; ai0[k]=ai0[i]-itmp0;
      ar0[i]+=rtmp0; ai0[i]+=itmp0;
      }
      r=(ru*rw-iu*iw)>>10;
      iu=(ru*iw+iu*rw)>>10;
      ru=r;
    }
  }
}
```

For the multiple FFTs in the tap solver, the keys for optimization of the area/speed are loop unrolling, pipelining and resource multiplexing. Although Xilinx provides IP cores for FFTs, they are considerably larger and much faster than required. For example, a single v32FFT core in Xilinx Core-Gen library utilizes 12 multipliers and 2066 slices. Since we

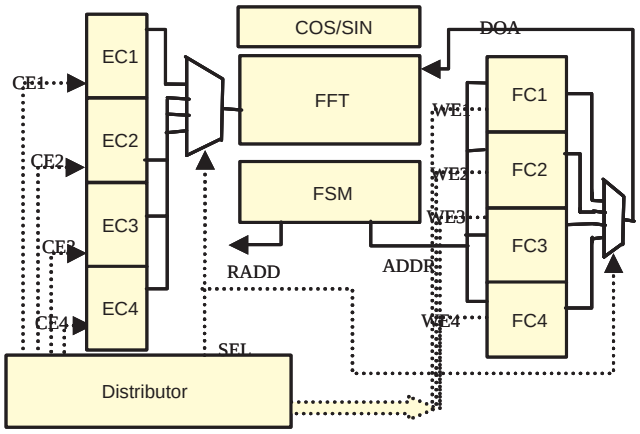

Fig. 7. A block diagram of the Catapult C scheduled RTL for the merged 4 MIMO-FFT/IFFT module.

need MIMO FFTs and IFFTs in one computation period, it is not easy to apply the commonality by using the IP core.

For the MIMO-FFT/IFFT modules, we can design a fully parallel and pipelined architecture with parallel butterfly-units and complex multipliers laid out in the pattern of a butterfly-tree at one extreme. To achieve the best area/time tradeoff in different situations, we apply Catapult C to schedule customized FFT/IFFT modules. We design the merged multiple input multiple output FFT modules to utilize the commonality in control logic and phase coefficient loading. The initial cos/sin phase coefficients for all stages are stored in ROM blocks. Parallelism/pipelining in the parallel FFTs are studied extensively in multi-levels. e.g., the Butterfly-Unit (BFU) level, the stage level and the FFT-processor level. By using Merged-Butterfly-Unit for MIMO-FFT, we utilize the commonality and achieve much more efficient resource utilization while still meeting the speed requirement.

The merged FFT/IFFT module architecture from Catapult C scheduling is shown in Fig. 7. In this block diagram, we have four parallel input ports to the FFT core. They share the same cos/sin phase coefficients from the ROM blocks. A Finite-State-Machine (FSM) will be generated to load and save data to the working space. In this figure, the ECs contain the input time-domain coefficients while the FCs contain the frequency-domain FFT/IFFT coefficients. A distributor will allocate the correct timing to output the coefficients to the next stage module in the equalizer.

As a simple example with four FFTs, the Catapult C scheduled RTL for 32-point FFTs with 16 bits are compared with a Xilinx v32FFT Core in Table. I. In a parallel layout with duplicate FFT modules, all the computations are localized and the latency is the same as one single FFT. For a reused module in serial processing, extra control logic needs to be designed for the multiplexing. The latency is equal to $4\times$ of the single FFT computation. However, with merged FFTs to apply the commonality, the Catapult C design demonstrates much smaller size for different solutions, e.g. from solution 1 with 8 multipliers and 535 slices to solution 3 with only one multiplier and 551 slices. Overall, solution 3 represents the

TABLE I

ARCHITECTURE EFFICIENCY COMPARISON FOR CATAPULT C VS. XILINX IP CORE

Architecture	mult	cycles	Slices
Xilinx Core	12	128	2066
Catapult C Sol1	8	570	535
Catapult C Sol2	2	625	543
Catapult C Sol3	1	810	551

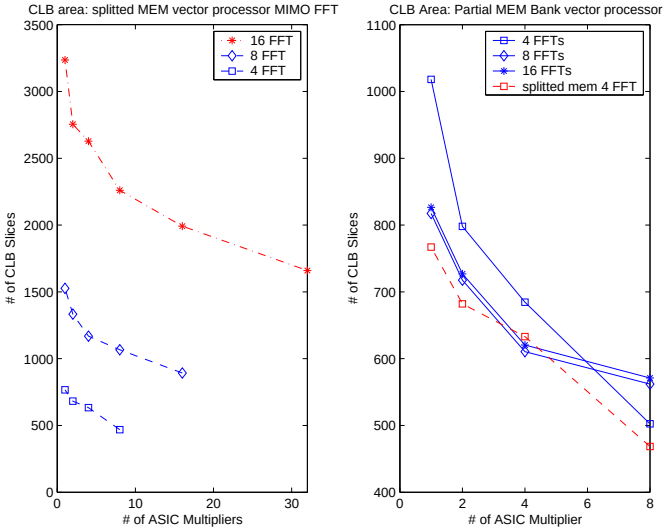

Fig. 8. CLB vs. # of multipliers for the different architectures of merged MIMO-FFT module.

smallest design with slower but acceptable speed for a single FFT. Compared to 4 parallel FFT blocks (each with 1 MULT) at 2204 slices and 810 cycles or 4 serial-FFT at 3240 cycles, the resource utilization is much more efficient.

The design space for different numbers of merged FFT modules is shown in Fig. 8 and Fig. 9. Fig. 8 shows the CLB consumption for different architectures versus the different number of multipliers. Fig. 9 shows the latency versus the number of multipliers for the merged MIMO-FFT modules. For the input and output arrays, two different type of memory mapping schemes are explored. One scheme applies split sub-block memories for each input array labelled as SM. This option requires more memory I/O ports but increases the data bandwidth. Another option is a merged memory bank to reduce the data bus. However, the data access bandwidth is limited because of the merged memory block. The details of the implementation are omitted here. However, this demonstrates the design space exploration capability enabled by the Catapult C methodology. We also designed the merged submatrix inverse and multiplication module also in block mode hardware mapping following the described VLSI architectures.

VI. PRODUCTIVITY EFFICIENCY

Table II compares the productivity of the conventional HDL based manual design method and the Catapult C based design space exploration methodology. For the manual design method we assume that the algorithmic specification is ready and there is some reference design either in Matlab or C to use as a baseline source code. For the Catapult C design we

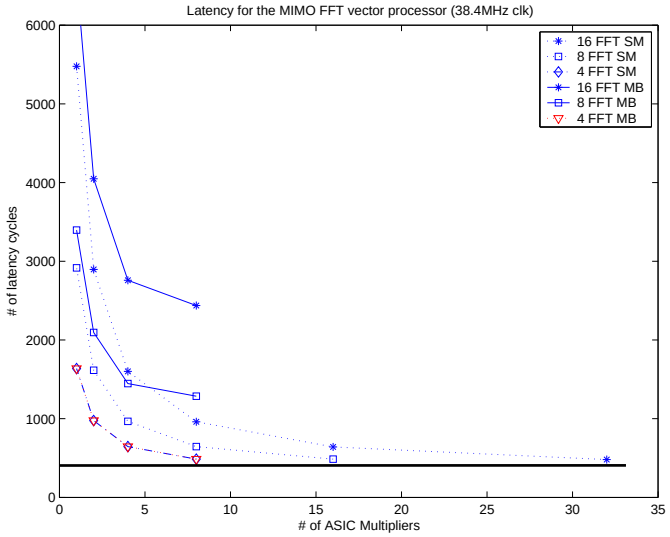

Fig. 9. Latency vs. # of multipliers for the merged MIMO-FFT module.

TABLE II

PRODUCTIVITY IMPROVEMENT FROM THE UN-TIMED C BASED DESIGN SPACE EXPLORATION

Task	VHDL	Catapult C
Covariance estimation	3 weeks/sol	1 week
Channel estimation	3 weeks/sol	1 week tradeoff study
MIMO-FFT	5 weeks/sol	2 weeks tradeoff study
FIR Filtering	3 weeks/sol	1 weeks tradeoff study

assume that the fixed-point C/C++ code has been tested in a C testbench using test vectors. The work load does not include the integration stage either within HDL designer or writing some high-level wrapper in VHDL. For the Catapult C design flow, there are possibly many rounds of edit in the C source code to reflect different architecture specifications. It is shown that with the manual VHDL design, it may require a much longer design cycle to generate one working architecture than the extensive tradeoff exploration using Catapult C. The improvement in productivity for the given case study in the 3G and beyond MIMO-CDMA equalizer is significant compared with the conventional design methodology.

VII. CONCLUSION

In this paper, we described a rapid verification methodology integrating Catapult C and other key technologies. We used Catapult C to build the backbone RTLs of new advanced algorithms in the MIMO-CDMA system and integrated them with Xilinx IP cores as well as HDL designer blocks with the schematic capture of HDL designer. We studied FPGA architecture tradeoffs efficiently and found the most efficient solution for a specific architecture/resource constraint. The productivity was improved significantly by moving the work-load from low-level logic layout to high-level abstraction of the architectural research.

REFERENCES

[1] G. J. Foschini, *Layered space-time architecture for wireless communication in a fading environment when using multi-element antennas*, Bell Labs Tech. J., pp. 41-59, 1996.

[2] G. D. Golden, J. G. Foschini, R. A. Valenzuela and P. W. Wolniansky, *Detection algorithm and initial laboratory results using V-BLAST space-time communication architecture*, Electron. Lett., Vol. 35, pp.14-15, Jan. 1999.

[3] Y. Guo, J. Zhang, D. McCain and J. R. Cavallaro, *Scalable FPGA architectures for LMMSE-based SIMO chip equalizer in HSDPA downlink*, 37th IEEE Asilomar Conference, Monterey, CA, 2003.

[4] J. Bhasker, *VHDL Primer: third edition*, Prentice-Hall, 1999.

[5] http://www.systemc.org/.

[6] http://www.celoxica.com/methodology/handelc.asp.

[7] Y. Guo, J. Zhang, D. McCain and J. R. Cavallaro, *Efficient MIMO equalization for downlink multi-code CDMA: complexity optimization and comparative study*, to appear in IEEE GlobeCom 2004.

[8] J. Yue, K. J. Kim, J. D. Gibson and R. A. Iltis, *Channel estimation and data detection for MIMO-OFDM systems*, IEEE Globecom, vol. 22, no. 1, pp. 581 - 585, Dec 2003.

[9] K. Hooli, M. Juntti, M. J. Heikkila, P. Komulainen, M. Latvaaho and J. Lilleberg, *Chip-level channel equalization in WCDMA downlink*, EURASIP Journal on Applied Signal Processing, pp. 757-770, Aug.2002.

[10] G. De Micheli and D. C. Ku, *HERCULES - A System for High-Level Synthesis*, the 25th ACM Design Automation Conference, Anaheim, CA, June 1988.

[11] G. De Micheli, *Hardware Synthesis from C/C++ models*, Proceedings of the conference on Design, automation and test in Europe, Munich, Germany, pp. 80, 1999.

[12] R. Domer, *System-level Modeling and Design with the SpecC Language*, Ph.D. Thesis, University of Dortmund, 2000.

[13] C. Y. Wang and K. K. Parhi, *High-level synthesis using concurrent transformations, scheduling, and allocation*, IEEE Trans. On Computer-Aided Design, vol. 14, no. 3, March, 1995.

[14] *Catapult C Manual and C/C++ style guide*, Mentor Graphics, 2004.