System-level Design Methodology Enabling Fast Development of Baseband MP-SoC for 4G Small Cell Base Station

Tang Shan, Zhu Ziyuan, Su Yongtao
Wireless Communication Technology Research Center, Institute of Computing Technology, Chinese Academy of Sciences
Beijing Key Laboratory of Mobile Computing and Pervasive Device
Beijing, China
tangshan@ict.ac.cn

***Abstract*—"Small Cell" is regarded as the solution to optimize 4G wireless networks with improved coverage and capacity and expected to be deploy in a large number. To meet performance requirements and special constraints on the cost and size, we design a heterogeneous multi-processor SoC for small cell base station, which is composed of ASP (Application Specific Processor) cores, hardware accelerators, general-purpose processor core, and infrastructure and interface blocks. The challenges of developing such a complex chip drive us to employ system-level design methodology in both single core and mutli-core architecture optimizations. The paper discusses in detail the LISA (Language for Instruction-Set Architectures)/SystemC based ASP-algorithm joint optimization, and task-graph driven multi-core architecture exploration. Finally, the results of silicon implementation on SMIC 55nm technology are presented.**

***Index Terms*— System-level design, MP-SoC, ASP-algorithm joint optimization, multi-core architecture exploration**

I. INTRODUCTION

By year 2015, mobile subscribers are expected to exceed 6 billion and mobile data will be 30 times more than that of today [1]. The actual numbers may be even much larger considering the foreseeable exponentially grow of M2M applications. One solution to help wireless operators to optimize their networks with improved coverage and capacity is "Small Cells", which are low-powered radio access nodes providing a small radio footprint from 10 meters to 1 or 2 kilometers. By reducing the "distance" between the users and the base station as well as the number of instantaneous users, small cells are regarded as the vital element to improve the signal quality and off-load the mobile data traffic, which are essential in 4G networks.

The required performance of a small cell base station is not at the same level as the requirements for a macrocell base station. However, as an equipment will be deployed in a large quantity, small cell base station should be cost and energy efficient, and compact for acceptance. In this context, we designed a baseband MP-SoC for small cell base station, which is composed of application specific processors (ASPs), hardware accelerator, general-purpose processor cores, infrastructures, and interfaces. Together with the firmware and software running on it, this SoC provide all baseband processing functionalities.

This work is funded by National Science and Technology major project ZX2013ZX03003014, Beijing natural science foundation major project 4110001, and Beijing municipal science & technology commission project Z121100001412006

For the key component, ASPs, to our best knowledge, previous research, and development focus on the mobile devices instead of base station. For example, H. Lee et al. [2] present 'SODA-II', a low-power DSP core for wireless communications, and Ziyuan, Zhu et al. [3] present a 100 GOPS ASP designed for LTE mobile devices. Designing an ASP suitable for small cell base station is the first challenge.

Moreover, almost every phases of developing such a complex heterogeneous MP-SoC are challenging, from algorithm and architecture design to the hardware and software implementation, which result in longer R&D cycles and higher risk of failure. On the other hand, for the system-level design methodology whose purpose is improving the productivity of SoC design, it is a perfect battlefield [4][5][6].

In our project, system-level design methodology is used mainly in two design phases. The first one is optimizing single ASP core, targeting at most efficient ISA and micro-architecture for particular baseband signal processing functions, such FFT, FIR filter, matrix operation, and map/de-map. Using LISA [7] "Language for Instruction-Set Architectures", the ASP is modeled in a higher abstraction level so that we can analyze and fine tune the ASP core together with key algorithms within very short iterations. Secondly, for tuning the task mapping among ASP cores and hardware accelerators, and achieving better design of on-chip connections and storages, task-graph driven multi-core analysis methodology are used. It provides reliable estimates for the expected system performance of the small cell baseband processing applications.

The rest of the paper is structured as follows. Section II introduces hardware and software system of SoC based small cell base station as a background. Next, Section III gives an overview of the enhanced design flow employing the system-level design methodology. Section IV and Section V discuss the system-level design methodology used in single ASP core and multi-core optimization in details. Section VI presents the design environment, the final architecture of the baseband chip, and silicon implementation results. Section VII summarizes the paper at last.

II. SMALL CELL BASE STATION OVERVIEW

A 4G small cell base station is a compact equipment with complex hardware and software functions and subsystems. In

978-3-9815370-2-4/DATE14/©2014 EDAA

general, the hardware includes the RF and antenna subsystem, the baseband processor, MPU and peripherals, backhaul interfaces and other basic hardware blocks

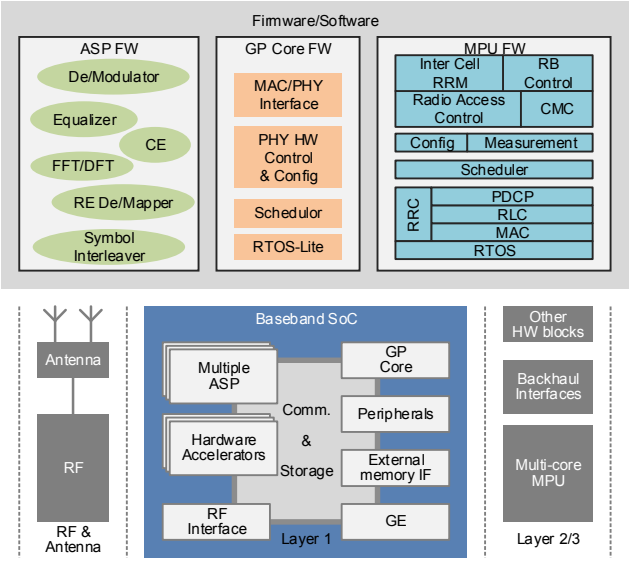

Fig. 1. Small cell base station System Architecture

The digital baseband processing in a 4G small cell base station is the focus of this paper. It is divided into several layers (shown in Fig.1.). Layer 1 is also called PHY, perform signal processing functions to transmit and receive data according to the air interface standard from 3GPP [8], which are implemented in the digital signal processing components, such as the ASP cores and hardware accelerators. The firmware running on the ASP cores provide basic signal processing functions, such as FFT/DFT, De/Modulation, channel estimation, equalization, RE map/de-map, interleaving etc. Besides, a L1 control firmware running on a general-purpose processor controls all hardware and firmware blocks and communicates to higher layers.

L2 and L3 layers can be decomposed in three sub layers: medium access control (MAC), radio link control (RLC) and packet data convergence protocol (PDCP). These layers are implemented as software running on a general-purpose processor, e.g. the multi-core MPU in Fig.1. Above that, there are configuration, management and application software.

III. Design Flow with System-level Design Methodology

In a traditional design flow, developing a SoC is as the dark gray parts of Fig.2. The flow starts from analyzing the requirements coming from standards and applications. Based on these requirements, algorithm and architecture are designed. Then, ASP cores and other hardware blocks are implemented in RTL. With all hardware blocks available, RTL to GDSII flow is conducted. On the other hands, the firmware/software are developed and debugged on system prototype, usually an FPGA system, and then on the real chip.

The old fashion works fine with “simple” designs, either in structure or in optimization targets, but is not good enough for the design which is not only complex but also ambitious in optimization of performance, cost or power consumption.

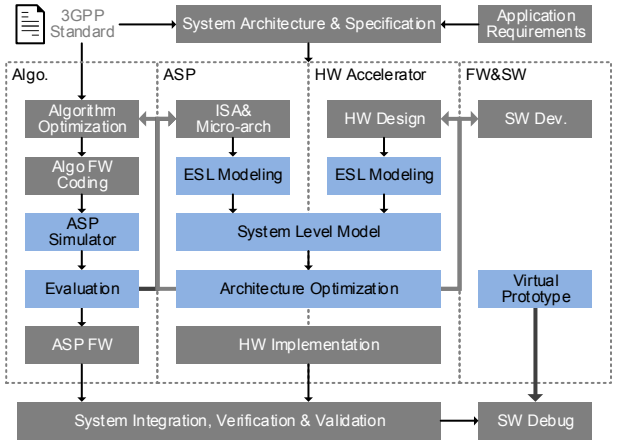

Fig. 2. System-level Design Methodology Enhance Design Flow

The light blue parts in the figure are design stages where system-level design methodology could be utilized to improve the productivity and avoid wrong-design, under-design or over-design in architecture exploration phase or implementation phase. The concept is described in [9], "the utilization of appropriate abstractions in order to increase comprehension about a system, and to enhance the probability of a successful implementation of functionality in a cost-effective manner." Following sections will demonstrates how this methodology helps us in real projects.

IV. ASP-algorithm Joint Optimization

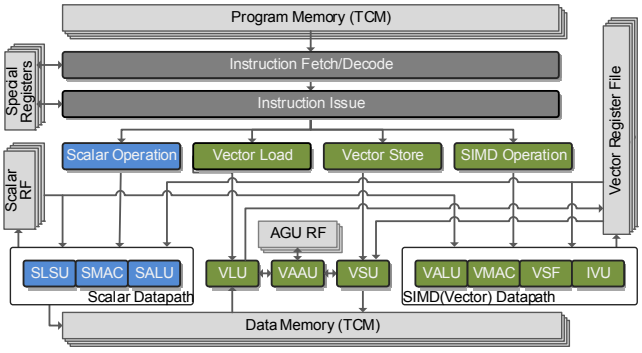

Fig. 3. Maotu ASP core basic architecture

The “Maotu” ASP is designed for baseband processing in 4G wireless communication. It is a variable length VLIW machine with a SIMD data path (shown in Fig. 3.) supporting a rich set of vector data processing instructions, which is suitable for high volume parallel data processing in MIMO/OFDM system. Meanwhile, Maotu has a relatively simple control path with regard to simple program flow of the target applications. As part of the baseband processor SoC, Maotu can be used to realize a wide range of signal processing functions including cell search, OFDM demodulation, channel estimation, MIMO detection, QAM de-mapping and downlink data composition. In the SoC, several Maotu cores are working

under the control of a GP (General Purpose) core to realize the physical layer procedures.

By profiling the major algorithms of small cell baseband processing, it reveals that most of them could be classified into following categories: FFT/DFT, matrix operations, map/de-map and filters. Because of the different characteristics of these operations, it is difficult to design a unified architecture suitable for all of them. Therefore, starting from a basic ASP template, we derived three types of Maotu ASPs, which are different in ISA, or microarchitecture, or memory access mechanism, or configuration of local memory to achieve higher efficiency for specific operations, i.e. Maotu-T1 for FFT, Maotu-T2 for matrix operation, and Maotu-T3 supporting vector division.

A. Architecture Exploration

Design space exploration of baseband processing ASPs is difficult due to the intrinsic complexities of wireless algorithms and processor design. Evaluating different processor instruction set and micro architecture together with a large variety of algorithms is a very tedious and error prone task. The ASP design is a joint optimization process of algorithm and hardware architecture. Any modification on the either algorithm or architecture will lead to a new modeling-simulation-evaluation cycle, which usually takes weeks or even months to be accomplished. This problem, in practice, limits the number of iterations and results in over/under-design in most cases.

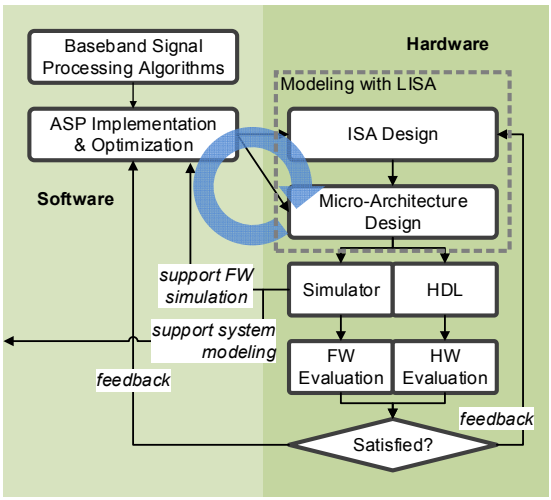

Fig. 4. ASP-algorithm Joint optimization Flow

To gain "perfect" ASP/algorithm design for 4G baseband processing, as shown in Fig. 4. , we using LISA to design the ISA and micro-architecture and deployed Processor Designer [10] tools from Synopsys, which accelerates the processor design and optimization process significantly by using the system-level design methodology. The flow start from the algorithm design, whose outputs are link level baseband algorithm simulation model and performance results that should meet the requirements of standards. Then, for the key algorithm functions mapped to ASP, initial fixed-point implementation codes are derived from simulation models. At the same time, the initial architecture design of the ASP starts based on the understanding of the key algorithm and design experience, which results in the first version of ISA and micro-architecture. After that, the ASP is modeled in PD tools using LISA language. Then, the simulator (an ISA simulator and a cycle-accurate simulator) and RTL code can be generated automatically. The ASP simulator is used to run and analyze the algorithm firmware codes while the RTL design can be used for cost (area) and power consumption evaluation. Both hardware and firmware evaluation results, such as the cycle count of critical functions, the usage of specific instructions, or the area of hardware modules are valuable feedbacks to the architecture refinements or the algorithm optimization.

The iteration will stop when the optimization target of the ASPs are hit or carry on until the end of the architecture design of whole chip to solve the system level problems. During system-level architecture design, we can derive requirements or constraints for the ASPs, which are more accurate to guide further fine-tune works.

Although, above optimization procedure is a common practice in ASP design, high-level modeling and analysis methodology and tool chain reduces the time of one iteration from months to days, even hours. It enables architecture exploration in bigger or deeper question/solution spaces.

B. Case Study on FFT

The performance (cycle count) of the 2048-point FFT operation is a typical benchmark for the baseband signal processing, which is affected by both ASP architecture and algorithm implementation.

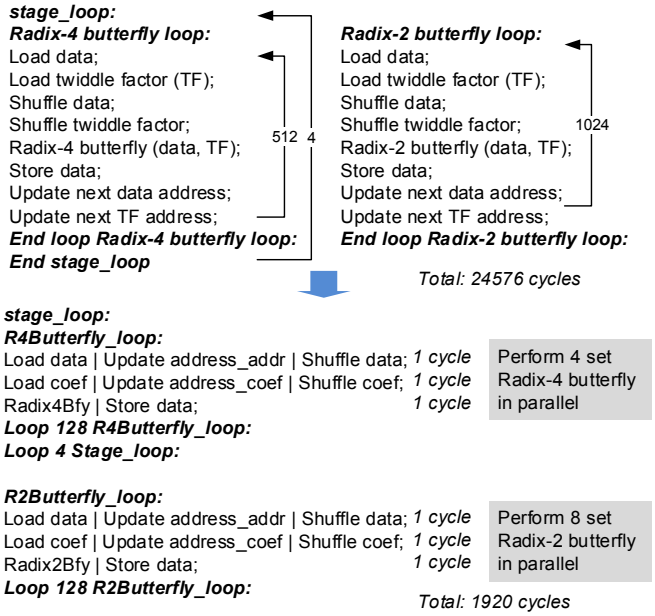

Fig. 5. Optimizing 2048-point FFT

In general, radix-4 and radix-2 mixed 2048 point FFT is most efficient structure. It is done in two loops, one for radix-4 butterfly, and the other for radix-2 butterfly, shown in left side of Fig. 5. For a general-purpose processor, the "Radix4Bfy" and "Radix2Bfy" are functions realized by several "multiply",

"add" and the "shuffle" operation. Even if these operations can be done in one cycle, it needs at least *8×512×4+8×1024 =24576* cycles to complete one 2048-point FFT.

Obviously, to reduce the total cycle count, we can exploit the data and instruction parallelism, i.e. making use of wider data-path and more MAC units to accelerate the butterfly operation and issue more instructions at the same time.

Following the ASP design flow, at first, we extend the general-purpose instructions with following specialized ones to facilitate algorithm implementation.

vector load/store instructions *for load/store data and twiddle factors (aka. Coefficients for FFT),*

shuffle instructions *for data reordering in the FFT algorithms,*

butterfly instructions *for computing a radix-2 or radix-4 butterfly,*

address generation instructions *for updating the address for data store and load,*

loop control instructions *for the program flow control, supporting the zero leading hardware loops.*

Secondly, we design the SIMD/VLIW micro-architecture to support these instructions and provide the capability of issuing and executing multiple instructions at the same time. Then we re-code the FFT algorithm based on new ISA and micro-architecture and do simulation and evaluation. Based on evaluation results, we start another iteration.

For each version of implementation, we do the profiling of the algorithm firmware to find the most significant contributor to the cycle count and give the possible solutions either by modifying the ASP or re-coding the firmware. TABLE I. shows four major optimizations we have done to achieve the final 2048-point FFT design. One thing should be mentioned is that the optimization is done under the constraints of area and power consumption and stopped with a balanced design.

TABLE I. IMPROVEMENT OF CYCLE COUNT FOR THE 2048-POINT FFT

Initial Design	VLIW	Optimized VLIW	Optimized Load/Store
6472	3844	2352	1920

Finally, we realized 2048-FFT in 1920 cycles on a four slots (three for vector instructions) VLIW, 512 bit width SIMD ASP with special vector load/store and shuffle instructions, as shown on the right side of Fig.5.

TABLE II. shows the cycle count of other major operations for uplink (from user equipment to small station) receiving procedure running on optimized ASP cores. The algorithms used in experiments are relatively simple but can guarantee the overall requirements defined by 3GPP standards.

TABLE II. OPTIMIZATION RESULTS OF OTHER TYPICAL OPERATIONS

Function	Cycle (for one symbol)
De-REmap	1157
Channel Estimation	473
Equalization	2006
IDFT-600	9625
DE-Modulation (64QAM)	7546

V. MULTI-CORE ARCHITECTURE OPTIMIZATION

By optimizing the ASP and particular algorithm functions running on it, a satisfied single core design is accomplished. However, it is not necessarily lead to a good design of entire baseband processing system chip, which requires further optimizations.

The goal of the task-graph driven multi-core optimization is to provide reliable estimates for the expected system performance of the baseband processing. It can be done in the early stage and does not require RTL models, nor does it require a set of fully refined application images for the ASP cores. Instead, two different kinds of models, a task graph of baseband processing, and a hardware platform model with generic virtual processing blocks are required (as shown in Fig. 6.) .

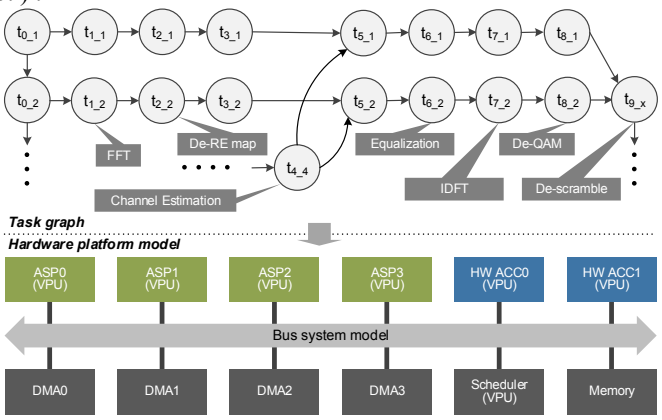

Fig. 6. Task-graph driven analysis of the system

The upper part of figure shows a small part of the task graph describing the uplink receiving procedure of the small cell baseband processing.

The first key element is the tasks (task graph nodes) specifying which portions of the application can run in parallel. In our example, each task is noted in the form "t_{x_y}", where the first digit is the index for one particular function, such as FFT, de-REmap, channel estimation, equalization, IDFT, de-QAM, or de-scramble; and the second one is the index for the symbol in a sub-frame being processed (defined in [8]). For example, "t_{2_2}" means the function "de-Remap" done on the 2nd symbol in a sub-frame. One important parameter for task is the processing delay, which specifies the amount of time a task occupies a processing resource when being activated. In our practice, the processing delay of a task running on ASP core is calculated by multiplying the cycle count number (obtained from ASP-algorithm Joint optimization process discussed in section IV) and expected clock frequency of ASPs. For a task running on hardware accelerator, the delay was estimated by static analysis of the reference code or by simulation of existing IP. Another important parameter for task is memory accessed per activation, modeling the amount of bytes written and read.

The second factor of the task graph is the dependencies (task graph edges), which specify in which order the tasks have to be activated for processing a specific set of data. One

essential parameter for task graph edge is the activation rate, specifying the frequency of task activations.

After creation of the task graph, it can be mapped onto the virtual hardware platform model shown in the lower part of Fig. 6. This model is created in Synopsys Platform Architecture tools [10] containing the VPU (Virtual Processing Unit), bus system model and memory in SystemC/TLM. A VPU is a processing resource for running a number of tasks.

How to map the tasks into the hardware platform is specified in a file as the input to the simulation. For one tasks mapping scenario, the performance analysis data of computation and communication could be collected from the simulation. The computation analysis focuses on the load of processing elements, such as ASPs, hardware accelerators, and schedulers. The result indicates how many and which kinds processing elements are required to fulfil the overall requirements. Meanwhile, it shows the best way to partition the target application. The communication analysis provides valuable information to optimize the interconnection, and the memory subsystem. When simulating a mapped baseband processing system, the complete model, including VPU and firmware, approximates the traffic that would be generated by a real processor executing real firmware. The utilization data of interconnection and memory can be collected and studied.

To compare different mapping scenario efficiently, we employed IMPO flow that sweeps over different simulations, and compares the results automatically. TABLE III. shows the computation analysis results (in percentage) for each of key processing elements in different mapping scenario. The insights we get from them are: 1) one or two ASP cores plus one hardware accelerator cannot finish all tasks in time. 2) adding another hardware accelerator will reduce ASP2's utilization number by about 20% but will increase the work load of scheduler by about 12.5%. 3) In the first mapping scenario of "3ASP+2ACC" configurations, the workloads of the three ASP cores are not balanced. The reason is the ASP2 has to wait for completion of processing on certain data in ASP1. By refining the task mapping on these ASP cores (as shown in the second mapping scenario of "3ASP+2ACC" configurations), the utilization of ASP1 and ASP2 are balanced.

TABLE III. UTILIZATION RESULTS FOR DIFFERENT MAPPING

Scenario	scheduler	ASP0	ASP1	ASP2	ACC0	ACC1
1ASP + 1ACC	25.4	>100	0	0	64.6	0
2ASP + 1ACC	36.1	>100	>100	0	64.6	0
3ASP + 1ACC	41.6	86.2	66.7	98.2	63.4	0
3ASP+ 2ACC(1)	46.8	86.2	66.7	78.1	63.4	34.6
3ASP+ 2ACC(2)	48.2	77.2	74.9	78.1	63.4	34.6

With similar methodology, we can quickly obtain other important information for making architecture decisions, such as the state statistics of the VPUs, the contention of on-chip communication, and utilization of shared storage. Moreover, this methodology is scalable and extensible. By integrating more functions into the processing model or adding more tasks' parameters, more architecture analyses are possible, e.g. high-level power analysis.

VI. IMPLEMENTATION RESULTS

A. Design Environment Setup

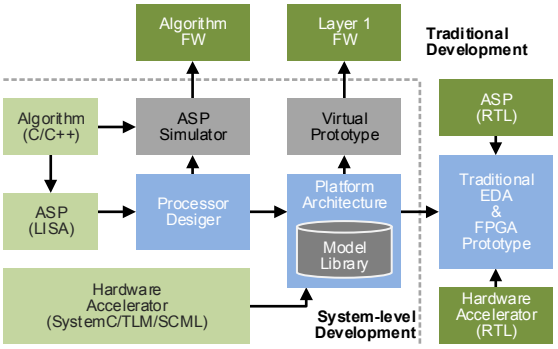

Fig. 7. Design Environment with System-level Design Methodology

Fig. 7. is our design environment with system-level design facilities. In addition to the traditional flow and tools, we employ Processor Designer (PD) and Platform Architecture (PA) to build an integrated system-level design platform.

In this environment, we model the ASPs in LISA language with PD, and simulate algorithm firmware on the ASP simulator generated by PD, and realize the task-graph driven multi-core optimization in PA.

Moreover, we integrate the SystemC/TLM models for hardware accelerator, ASP cores, and other hardware with models of on-chip connections. From it, a virtual prototype is derived to support the debugging of the baseband layer 1 firmware. In the traditional flow, we were expecting that the FPGA prototype could ease the design and debug of the firmware and software. However, for such a complex system, although the latest FPGA chip (Xilinx V7 2000T) and EDA tools are utilized, the efforts on bringing up the FPGA prototype is huge and even unpredictable to some extends. Virtual prototype saved us in firmware development by providing excellent capability to control and observe the system.

Then, we use the output from system-level design, such as the ASP and hardware accelerator models and architecture specifications, as the references for the implementation, which is conducted in the traditional chip design flow.

B. Final Architecture Design

The final baseband MP-SoC design are depicted in Fig. 8. Seven Maotu ASPs are implemented as inner transceiver, working together with hardware accelerators based outer transceiver (Outer Rx and Outer Tx) to perform major baseband signal processing. The GP core is the layer 1 control processor. The ASP cores, hardware accelerators and other hardware blocks, such as RF interface, external memory interface and peripherals are connected by AXI matrix and low speed peripheral buses.

Three types of Maotu ASPs are used in different stages of uplink receiving path and downlink transmitting path following

the architecture design (described in section IV and V), and some implementation constraints, e.g. avoiding difficulties in backend design. In the RX direction, data received from RF interface is processed by Maotu-T1 0, Maotu-T2 0 and Maotu T3 0 in a row for synchronization, de-modulation, channel estimation and equalization and fed to Outer Rx module for channel decoding. In the downlink TX (from small cell base station to UE) direction, the raw data from Layer 2 is coded by Outer Tx module, then fed to Maotu-T1 2 for modulation, then sent to RF subsystem through RF interface.

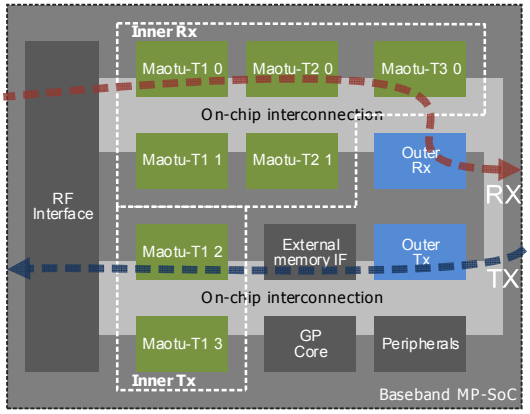

Fig. 8. MP-SoC Architecture

C. Silicon Implementation

Finally, the baseband MP-SoC is implemented on SMIC 55nm LL (Low Leakage) technology. Fig. 9. shows the layout of the chip.

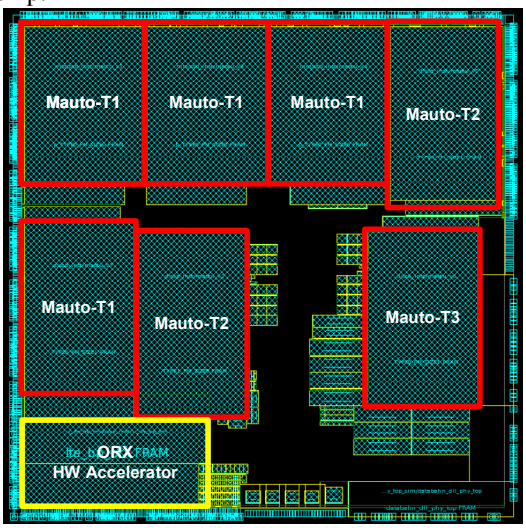

Fig. 9. Baseband SoC implemented on SMIC 55nm technology

The total die size is about 75 square millimeter. Three types of Maotu ASPs are implemented in similar die size. The maximum power consumption of whole chip is about 2000 mW. This figure is acceptable for the small cell base station. The low power design techniques, such as DVFS, multiple working modes, clock-gating and block level shut down, etc. are implemented. By optimizing the DVFS schemes, the power consumption could be further reduces.

ASP cores can run at configurable clock frequencies up to about 500MHz. At a clock frequency of 400MHz, four ASP cores' total computation power is more than 800 GOPS, which can supports more than 32 users (without using Maotu-T1 1, Maotu-T2 1, and Maotu-T1 3). TABLE IV. shows the area and maximum power consumption data of major blocks.

TABLE IV. SILICON IMPLEMENTATION RESULTS

HW Blocks	Area (mm^2)	Power (mW)
Maotu-T1	5.37	242@400MHz
Maotu-T2	5.46	257@400MHz
Maotu-T3	6.1	292@400MHz
ORX (Hardware Accelerator)	4.6	158@200MHz

VII. CONCLUSION

Designing and optimizing an ASP-based MP-SoC is not a trivial job because of its complex architecture and the huge efforts on hardware and software implementation. The system-level design methodology and tools significantly increase our productivity by 1) reducing the time of optimization cycle from months to days, even hours; 2) making quantitative analysis possible in early development stage avoiding over/under-design; 3) supporting the firmware and software debugging with virtual prototype. This paper demonstrates how it works in developing the baseband chip for 4G small cell base station.

[1] Jean-Christophe Nanan and Barry Stern, "Small Cells Call for Scalable Architecture", White Paper on www.freescale.com

[2] H. Lee, C. Chakrabarti, T. Mudge, "A low-power DSP for wireless communications", IEEE Trans. on Very Large Scale Integration (VLSI) Systems, vol. 18 (9), pp. 1310-1322. 2010.

[3] Ziyuan, Zhu; Shan, Tang; Yongtao, Su; Juan, Han; Gang, Sun; Jinglin, Shi, "A 100 GOPS ASP based baseband processor for wireless communication", Design, Automation & Test in Europe Conference & Exhibition (DATE), 2013, Page(s): 121 - 124

[4] Gerstlauer, A.; Haubelt, C.; Pimentel, A.D.; Stefanov, T.P.; Gajski, D.D.; Teich, J., "Electronic System-Level Synthesis Methodologies", Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on Volume: 28 , Issue: 10, , 2009 , Page(s): 1517 – 1530

[5] Jen-Chieh Yeh; Kung-Ming Ji; Shing-Wu Tung; Shau-Yin Tseng, "Heterogeneous Multi-core SoC Implementation with System-Level Design Methodology", High Performance Computing and Communications (HPCC), 2011 IEEE 13th International Conference on, Page(s): 851 – 856

[6] Willems, M.; Schirrmeister, F., "Virtual prototypes for software-dominated communication system designs", Communications Magazine, IEEE Volume: 48 , Issue: 6, 2010, Page(s): 37- 43

[7] Pees, S.; Hoffmann, A.; Zivojnovic, V.; Meyr, H., "LISA-machine description language for cycle-accurate models of programmable DSP architectures," Design Automation Conference, 1999. Proceedings. 36th, 1999 , Page(s): 933 - 938

[8] LTE/LTE-Advanced standards from 3GPP, www.3gpp.org

[9] Brian Bailey, Grant Martin and Andrew Piziali, ESL Design and Verification: A Prescription for Electronic System Level Methodology. Morgan Kaufmann/Elsevier, 2007

[10] Processor Designer (PD) and Platform Architect MCO (PA) User Manuals, on www.synopsys.com