Unified Hardware/Software Co-verification Framework for Large Scale Systems

著者	Nana Sutisna
その他のタイトル	大規模システムLSI設計のための統一的ハードウェア・ソフトウェア協調検証手法
学位授与年度	平成29年度
学位授与番号	17104甲情工第328号
URL	http://hdl.handle.net/10228/00006407

UNIFIED HARDWARE/SOFTWARE CO-VERIFICATION FRAMEWORK FOR LARGE SCALE SYSTEMS

NANA SUTISNA

Contents

List of Tables

List of Figures

List of Abbreviations

ADC	Analog-to-Digital Converter
API	Application Program Interface
ASIC	Application Specific Integrated Circuits
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BPSK	Binary Phase Shift Keying
CAD	Computer Aided Design
CAPIM	Client Application Interface Module
CFO	Carrier Frequency Offset
DAC	Digital-to-Analog Converter
DL	Downlink
DUT	Design Under Test
DVB	Digital Video Broadcasting
DVB-T	Digital Video Broadcasting - Terrestrial
EVM	Error Vector Magnitude
FEC	Forward Error Correction
FPGA	Field Programmable Gate Array
FFT	Fast Fourier Transform
Gbps	Giga byte per second
HAPS	High-performance ASIC Prototyping System
HDL	Hardware Description Language
HIL	Hardware In the Loop
HLS	High Level Synthesis
HW	Hardware
IFFT	Inverse Fast Fourier Transform
LLC	Logical Link Control
LSI	Large Scale Integrated System
LTE	Long Term Evolution
LUT	Look Up Table

MAC	Media Access Control
Mbps	Mega byte per second
MHz	Mega Hertz
MIMO	Multiple Inputs and Multiple Outputs
MLD	Maximum Likelihood Detection
MMSE	Minimum Mean Square Error
MPDU	MAC Protocol Data Unit
MSDU	MAC Sevice Data Unit
MU	Multi-User
OFDM	Orthogonal Frequency Division Multiplexing
OSI Model	Open Systems Interconnection Model
PA	Power Amplifier
PC	Personal Computer
PHY	Physical Layer
PE	Processing Element
PED	Partial Euclidean Distance
PER	Packet Error Rate
PLCP	Physical Layer Convergence Protocol
PPDU	PLCP Protocol Data Unit
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase Shift Keying
RAM	Random Access Memory
ROM	Read Only Memory
RF	Radio Frequency
RTL	Register Transfer Level
SAP	Service Access Point
SIFS	Short Interframe Space
SoC	System on Chip
SW	Software

TLM	Transaction Level Modeling
UMRBus	Universal Multi Resource Bus
VHT	Very High Throughput
VLSI	Very Large Scale Integrated Circuits
WiMAX	Worldwide Interoperability for Microwave Access
WLAN	Wireless Local Area Network

Summary

Currently, the complexity of embedded LSI system is growing faster than the productivity of system design. This trend results in a design productivity gap, particularly in tight development time. Since the verification task takes bigger part of development task, it becomes a major challenge in LSI system design. In order to guarantee system reliability and quality of results (QoR), verifying large coverage of system functionality requires huge amount of relevant test cases and various scenario of evaluations. To overcome these problems, verification methodology is evolving toward supporting higher level of design abstraction by employing HW-SW co-verification.

In this study, we present a novel approach for verification LSI circuit which is called as unified HW/SW co-verification framework. The study aims to improve design efficiency while maintains implementation consistency in the point of view of system-level performance. The proposed data-driven simulation and flexible interface of HW and SW design become the backbone of verification framework. In order to avoid time consuming, prone error, and iterative design spin-off in a large team, the proposed framework has to support multiple design abstractions. Hence, it can close the loop of design, exploration, optimization, and testing. Furthermore, the proposed methodology is also able to co-operate with system-level simulation in high-level abstraction, which is easy to extend for various applications and enables fast-turn around design modification. These contributions are discussed in chapter 3.

In order to show the effectiveness and the use-cases of the proposed verification framework, the evaluation and metrics assessments of Very High Throughput wireless LAN system design are carried out. Two application examples are provided. The first case in chapter 4 is intended for fast verification and design exploration of large circuit. The Maximum Likelihood Detection (MLD) MIMO decoder is considered as Design Under Test (DUT). The second case, as presented in chapter 5, is the evaluation for system-level simulation. The full transceiver system based on IEEE 802.11ac standard is employed as DUT. Experimental results show that the proposed verification approach gives significant improvements of verification time (e.g. up to 10,000 times) over the conventional scheme. The proposed

framework is also able to support various schemes of system level evaluations and cross-layer evaluation of wireless system.

Chapter 1

Introduction

1.1 Background

Recently, with the tight demand of time-to-market for product deployment, fast verification time has become a main hurdle to guarantee a reliable product. The fast verification time is more demanding particularly in development of a large scale system. The conventional simulation techniques, such as RTL simulation is unacceptable for complex SoC simulation and early embedded software development[1]. In particular, such system also employs various system parameters and supports multiple operational modes. As confirmed by a study conducted by Wilson Research Group and Mentor Graphics, the verification process takes significant amount of overall development time and become a bottleneck for design completion[2]. Furthermore, the gap between design productivity and circuit complexity will constantly happen in the future. It is occurred since the effort on design and verification cannot catch up the increasing of system complexity[3]. Verification is also called as evaluation when the context of assessments are more comprehensive. Some texts refer to evaluation term when the tasks not only cover validation of the correctness, but also include examination the feasibility and the interaction with other metrics or layers of information processing. However, the terms of verification and evaluation will be used interchangeably throughout the thesis.

A suitable example to represent this phenomena is system development in the field of

wireless communication system. For the last two decades, wireless communication technology has evolved in fast and continuous progression. Wireless system standard always changes to meet user experience demands, such as high-throughput, high-reliability, and various uses-cases. Unfortunately, every introduction of new standard always adopts complex and advanced signal processing, as well as has to support various system features. Consequently, the system complexity will increase significantly. For example, in the latest wireless LAN standard IEEE802.11ac[4], Downlink Multi User MIMO (DL MU-MIMO) system and higher-order modulation scheme (up to 256-QAM) are adopted. This standard is well known as very high throughput wireless communication system.

Figure 1.1 shows an illustration of design and verification productivity in wireless system development. The design complexity follows the Moore's Law; the circuit complexity doubled in 2 years, while the design productivity approximately doubled every 39 months. This chart confirms that the productivity gap tends to increase over the time.

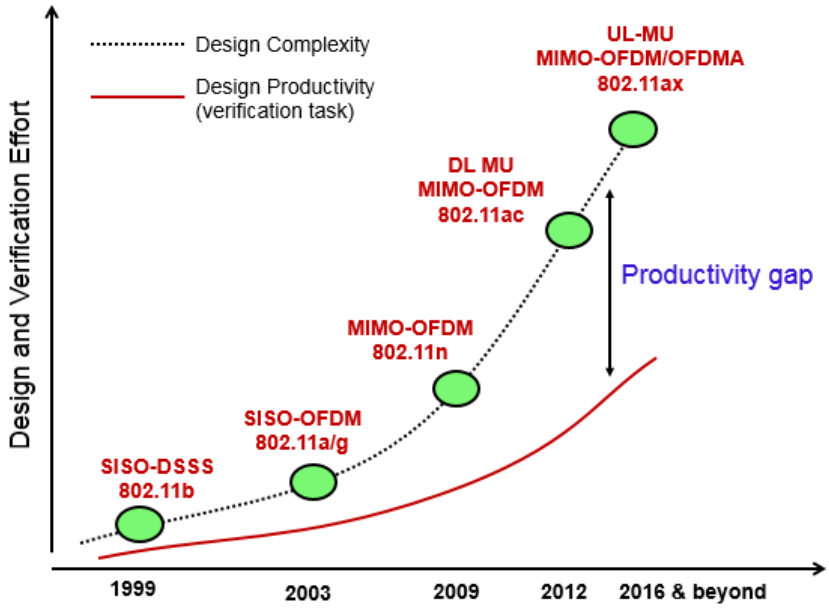

Figure 1.1: Design and Verification Productivity in Wireless System development

Verifying the system functionality in all possible conditions need huge relevant test cases in order to achieve a confidence level of acceptance test criteria. Furthermore, in development process, we also must validate each design layer, as depicted in Fig. 1.2. The

verification task in each layer involves :

1. **Algorithm evaluation**

   This task evaluates the correctness of algorithms transformation (e.g. from floating point to fixed-point)and also examines the impact of bit-width optimization in data-path design. Therefore, the employed algorithms do not degrade the overall system performance, exceeding the tolerable margin.

2. **Register Transfer Level (RTL) simulation**

   This task verifies the correctness of functionality of the designed hardware architecture according to the developed algorithm in system-level modeling.

3. **Hardware verification**

   This task performs the evaluation of the final implementation, whether satisfy the system requirements or not.

From those reasons, the verification task become a relevant issue in the development of a complex circuits.

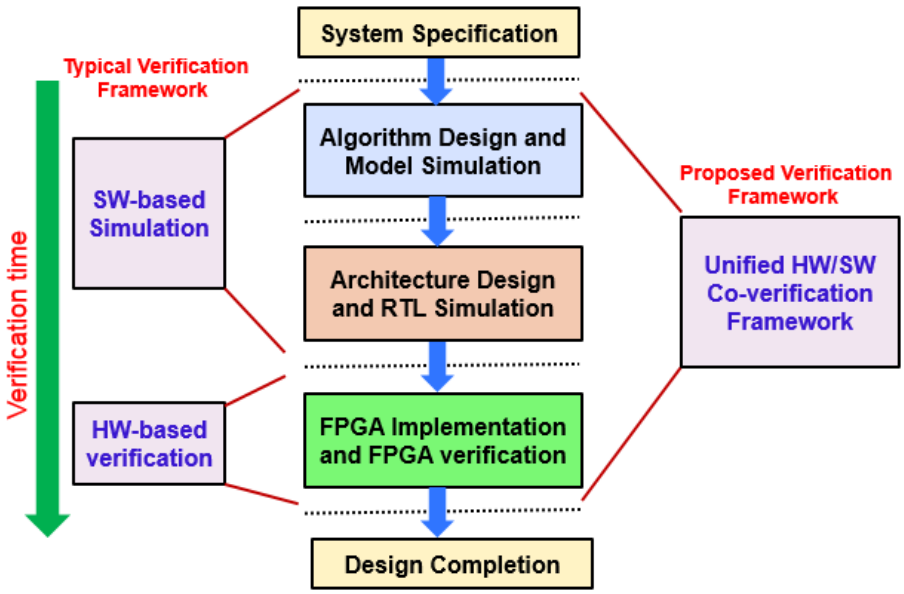

Figure 1.2: Design Verification Flow

The verification task in conventional method is typically carried out in each layer independently (as illustrated by dot line). The algorithm evaluation in system-level simulation and RTL simulation are carried out in software-based platform, while the hardware verification is performed later by using hardware-based platform, such as FPGA prototyping. For large-scale circuits, these conventional verification approaches (full-software verification and full-prototyping) have main drawbacks, which are : (1) excessive run-time verification and (2) low flexibility for covering large numbers of test scenarios and various use-cases of evaluations.

Furthermore, to be used as a comprehensive verification platform for wireless communication system, the existing verification approaches do not quantify clearly all required metrics, such as the efficiency of verification (verification time improvement), the flexibility to be employed with various function blocks, and the integration with system level simulation. As a result, those verification platforms cannot be used as integral part of system development process, particularly in the early stage of development.

To address the limitations of the existing verification methods, this thesis proposes a novel unified framework of HW/SW co-verification. This includes verification methodology, quantitative metrics of the effectiveness of evaluations, application examples, and performance characterizations. For the sake of clarity, the term of HW verification refers to set of system components that are executed in FPGA platform, while the SW verification refers to the code program that runs in host PC. This SW code verifies the rest of system components and maintains data flows with HW platform. Finally, it co-operates with HW design target in HW platform to build unified HW/SW co-verification framework.

1.2 Research Objectives

The objectives of this thesis is to present new methodology of verification framework for large scale circuits, specifically for wireless communication system. The study does not intent to promote particular HLS tools or present the advancement of specific FPGA platform. Instead, the study proposes a procedural and reproducible framework of verification for large scale circuits. Hence, the proposed framework is platform-independent and application-independent. Moreover, the framework should be easily employed for different

use cases in various FPGA platforms.

The proposed methodology includes :

1. **Unified co-evaluation framework**
   This allows for evaluation of various design layers with flexible partition of hardware design and software design. Therefore, the verification process can cover all verification stages, from block component up to complete system-level simulation, including algorithm verification, RTL verification, and real-time HW/SW co-verification. Thus, the proposed methodology closes the loop of design, exploration, optimization, and testing. This approach can avoid time consuming, prone error, and multiple iterations of design spin-off in a big research and development team.

2. **Data-driven simulation method**
   This method performs simulation based on the availability of data in HW/SW interface and further eliminates intensive HW/SW interaction. Hence, the proposed simulation method can improve the run-time

3. **Generic, flexible and scalable architecture in both HW and SW**
   These enable design extension and could be applied to different systems with only minor modifications.

4. **Tight integration with system level simulation**
   This feature allows for unified system level evaluation within high-level language (e.g. MATLAB, C/C++) and physical level verification.

In order to show the applicability of the proposed method, the unified HW/SW co-verification methodology is applied to our hardware platform with the case study of IEEE 802.11ac system. The co-verification of MLD MIMO Decoder of high throughput system is selected in order to evaluate the proposed method in verifying complex circuit, particularly for reducing computation time. Moreover, the whole 802.11ac transceiver is employed to show the capability of proposed method for system-level evaluation. Experimental evaluations of several performance metrics are carried out to confirm the effectiveness of our proposed co-evaluation framework.

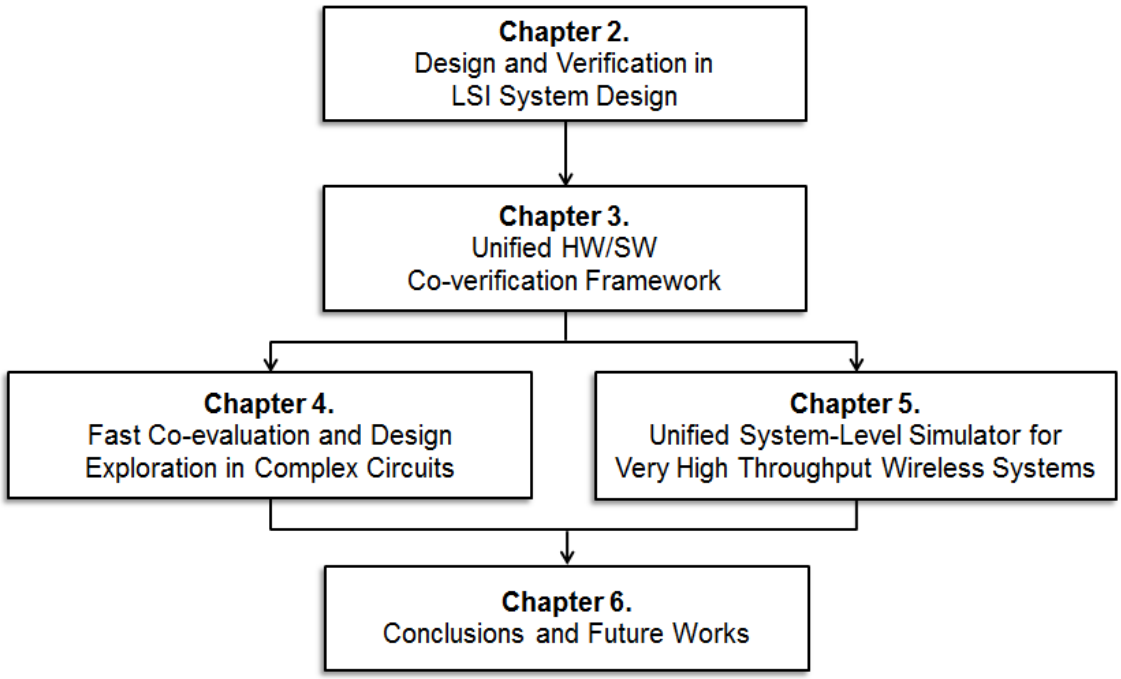

Figure 1.3: Thesis Organization

1.3 Thesis Organization

The structure of this thesis is shown by Fig. 1.3. The first chapter describes the motivations and the objectives of research task. The remaining chapters are organized as follows.

Chapter 2. Design and Verification in LSI System Design

This chapter presents an overview of existing design and verification methodology in LSI system design. In this chapter, we elaborate and investigate several important features of the existing methods. These include abstraction of simulation, types of verification framework and communication interface. Furthermore, we point out the necessity of research and development task to improve the efficiency of verification methodology in large scale circuits, particularly for Wireless Communication Communication Systems.

Chapter 3. Unified HW/SW Co-verification Framework

This chapter focuses on the description of the proposed co-verification methodology. This

includes task partition methodology, general architecture of HW/SW framework, and the development of hardware-in-the loop system. The first-two features, which are task partition methodology and the generic HW/SW architecture, are proposed in order to overcome the flexibility issue of co-verification framework. The task partition methodology in section 3.3 is proposed in order to accommodate various levels of verification task. The task partition is carried out by allocating some blocks to be implemented in HW and the remaining blocks are implemented in SW. Furthermore, all entire blocks could be verified concurrently in unified framework. Hence, we can select interested design more flexibly, while at the same time maintain the verification in the point of view of system-level simulation. The flexible and scalable architecture of HW/SW framework is explained in section 3.4.1.

This proposed framework is provided in order to facilitate the framework usage is applicable and extensible for various designs and applications. Additionally, in order to speed-up the simulation time as well as to cover various system parameters, the verification is carried out using hardware-in-the loop scheme that employs data-driven simulation, as described in section 3.4.3. This proposed scheme is expected to address the limitation of verification speed and also the verification coverage.

Chapter 4. Fast co-evaluation and Design Exploration in Complex Circuits

In order to show the applicability of the proposed co-verification method for evaluating complex circuit, chapter 4 presents application example of MLD MIMO Decoder. This block is selected as DUT since it is considered as the highest complexity circuit in transceiver system. The detail implementation of hardware and software for unified co-evaluation and the proposed data-driven co-simulation are described. In this chapter, the performance metric is also introduced in order to quantify the effectiveness of proposed design methodology. Finally, several examples of experimental evaluations of MLD MIMO decoder are presented, include the verification-time speed-up, the verification efficiency, and the evaluation of multi-dimensional design exploration.

Chapter 5. Unified System Level Simulator for Very High Throughput MIMO Wireless Communication System

This chapter shows the extension of the proposed co-evaluation framework for system-level

evaluation of wireless system. These include evaluation of transceiver algorithms, providing reproducibility data, supplying reference data for benchmark, and assessing cross-layer performance (PHY and MAC layer). The recent technology of wireless LAN system which is IEEE 802.11ac standard is selected as a case study. In this chapter, we show that the proposed methodology can perform flexible verification task within HW/SW platform and also maintain the verification in the point of view of system-level simulation. The issues of the flexibility such as design partition, tight-integrated with system-level simulation, and large-coverage of test scenarios have been addressed. Latter, the evaluation results are provided in order to show the capability of the proposed unified co-evaluation framework.

Chapter 6. Conclusion and Future Work

This chapter shows the summary of our whole works and the achievable results. We also discuss about the directions and recommendations on possible research tasks, to further improve the development of LSI system.

Chapter 2

Design and Verification in LSI System Design

This chapter aims to provide the reader a fundamental understanding on LSI design methodology and its verification system. Some concepts in HW/SW co-design and co-verification are explained. Furthermore, the key features and the major drawbacks of each method are evaluated to give insight and to provide a basis for conducting the research work.

2.1 HW/SW co-design Methodology

In recent years, we face the steady growth of advanced IC technology and complexity in system integration. The increased number of components in a system will imply higher degree of integration and result more complex designs. Traditionally, designers separate the hardware (HW) and software (SW) of an embedded system in early stage. The two groups of designers will develop their respective components independently. Moreover, the design process is performed sequentially. For example, the software design will be carried out after the hardware components are completely done and the verification task follows the final design task. Figure 2.1 shows illustration of a classical LSI system, involving hardware and software design[5].

From figure 2.1, the system specifications are first defined. Based on the initial specification, system level simulation and hardware design are performed. In many cases, the

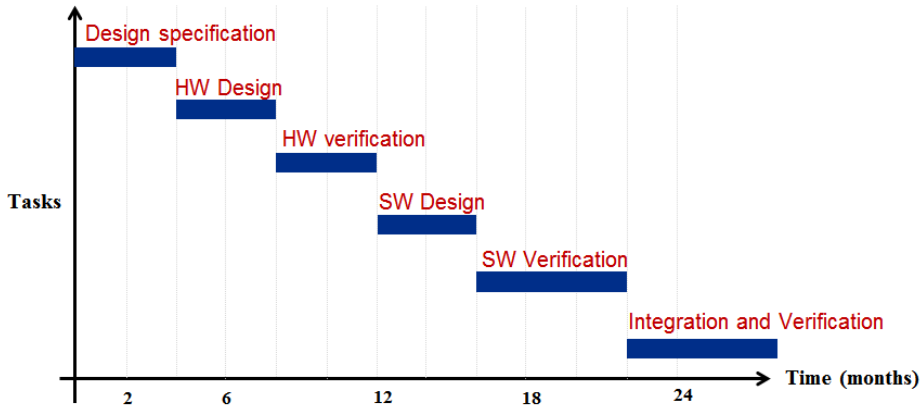

Figure 2.1: Conventional LSI design Flow

software team cannot start to develop and test their software until the hardware design is available. This has great risk of delaying the final product delivery when some design errors are detected very late. Additionally, we do not have any chance to explore potential option with respect to several implementation objectives, such as cost, performance and extensibility. Hence, the success of development is typically determined by the experience of designer team.

In summary, there are several main drawbacks of the conventional design flow [**?**], which are :

- *long development path*, resulting long and unpredictable time-to-market;
- *risk of potential errors* in each part design cannot be covered;
- *risk of over-design (excessive design) or under-design (insufficient design) of system* due to lack of early evaluation of design options.

To address the conventional LSI design methodology, HW/SW co-design has been considered as an established method to design complex IC circuits[5]. As supplementary to this design methodology, the verification methodology is also shifted into co-operative perspective. Co-design is a design methodology that allows the concurrent development of HW and SW in order to achieve system specifications. This method able to improve the predictability of embedded system design, by providing methods that tell to designers whether a system satisfies its requirements or not. Co-design methodology includes several tasks,

such as definition of system specification, design partitioning, modeling, validation, and implementation. An important note, co-simulation and co-verification have always been considered as an important topic in the co-design area, which is the main discussion in this thesis.

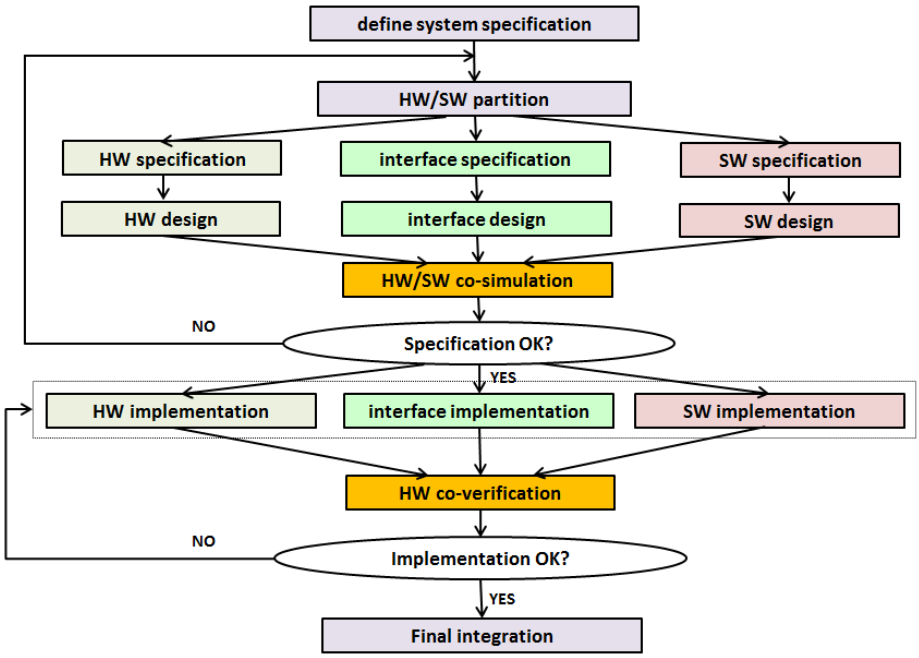

Figure 2.2: HW/SW Co-design Flow

The typical HW/SW co-design flow for LSI system could be figured out in Fig. 2.2. The development starts with defining initial system specification. Some mandatory parameters are selected in this stage, while other decisions such as architecture, algorithm, etc., could be refined in later stage, according to performance-cost trade-off.

The HW/SW partition stage is carried out to allocate where the set of system processing is executed. Several tasks are performed in hardware design, while the others functions are realized as SW in host processor or emulated in host PC. In this stage, the interface of HW and SW is also specified. The task mapping can be performed by considering cost metrics on each module or function. For HW component, the typical cost metrics are execution time, resource area, power consumption, and testability. On the other hand, SW cost metrics may includes execution time and required memory size. In practical, functions

with regular computation and parallel operation are realized in HW. Other functions that have irregularity and perform complex calculation are mapped into SW.

After partitioning is done, a simulation environment is developed in certain abstraction layer (design layer) to model its behavior, including computational process and data signaling. Some iterations of simulation are performed to estimate system performance. In typical DSP simulation, co-simulation is performed in high-level language. When system-level model satisfy the requirements, the hardware component could be implemented into hardware design in RTL level. At the same time the SW process is also implemented by adopting the data flow as described in system-level. This SW implementation could be re-used for final implementation.

In final stage, the RTL design is synthesized and verified on HW platform, which is FPGA. Hardware verification can be also performed with co-operation with SW design. If any error that affects system performance is found, the design has to move back to the previous stage. The designers need to modify the initial design or even to change specification. This design turn-around results a high cost as well as long cycle development.

2.2 HW/SW Co-verification

Verification task has become more challenging due to rapid increasing of system complexity and the requirement to maintain the gap between the productivity and circuit complexity. In the following subsections, we elaborate and investigate several characteristics of verification approaches, for example: achievable simulation time, the cost for system setup, simulation accuracy, debugging flexibility, and the coverage for large test scenarios.

2.2.1 Abstraction Level of Simulation

From the perspective of abstraction level, offered accuracy and degree of observability, as shown in Fig. 2.3, simulations can be classified into the following categories:

- **Data-flow simulation**, This simulation represents functional behavior of signals or data stream without notion of time. Each component/block is connected by signal and is executed when the inputs are available. Simulation is performed in high-level

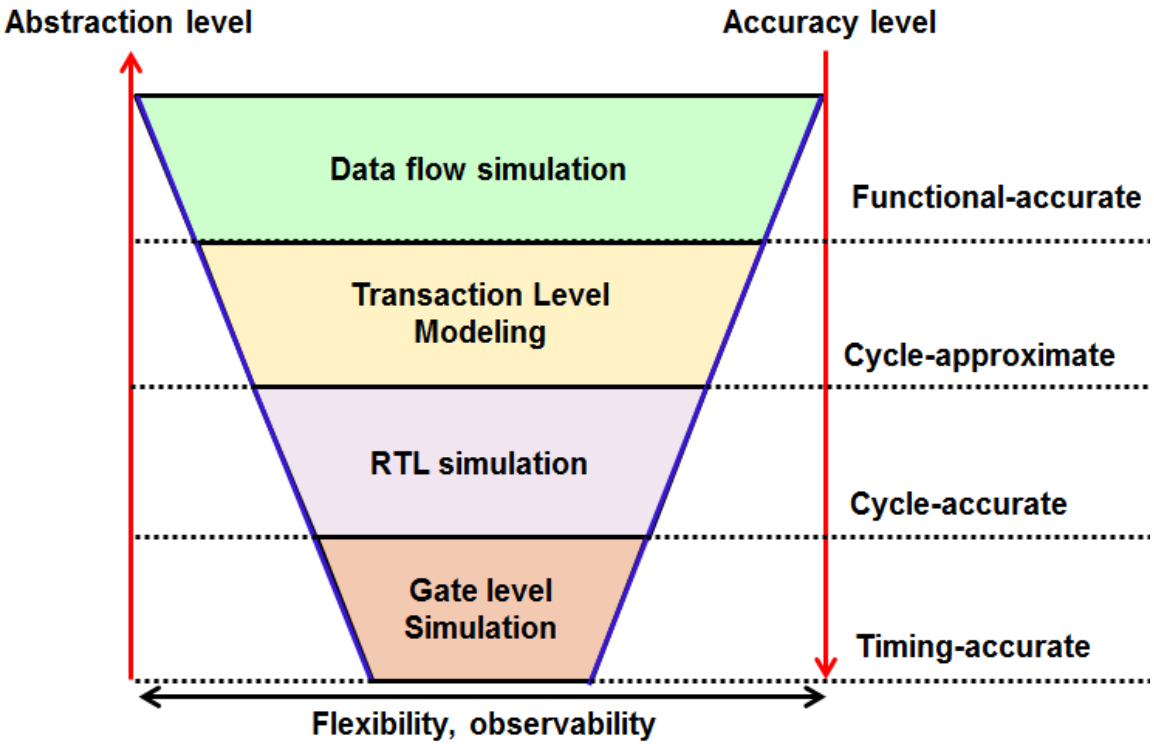

Figure 2.3: Abstraction of Simulation

abstraction during early stage of development. The Main objective of this simulation is to verify the correctness algorithm or data-flow of system. Since this simulation is performed without taking into account the timing behavior, the simulation results cannot predict timing accuracy performance of system.

- **Transaction-level simulation**. In this simulation. the details of communication (e.g. HW/SW interface) among computation components are separated from the details of the implementation of computation components. Communication is modeled as channel (interconnection) and transaction request takes place by calling interface functions of these channel models. Unnecessary details of communication and computation are hidden. In this simulation, SW function calls are used to model the communication between HW and SW components. For example, transaction level model (TLM) performs burst transfer task by using only single function call, with an object representing burst request and another object representing burst response.

This simulation is consider as cycle approximate simulation.

- **RTL simulation**. This simulation only calculates the state of the signals at clock edges and it is usually implemented for simulation of RTL hardware design. This simulation can predict the actual processing in cycle-count. However, the simulation does not reflect the design can work at actual speed. Furthermore, in this simulation any signals can be captured and debugged. Hence, it can be used for complex design and verification. However, this simulation suffers from excessive run-time simulation.

- **Gate-level simulation**. This simulation is the most accurate simulation since every active signals is evaluated during the clock cycle as it propagates. Each signal is simulated for its values and its time occurrence. To perform gate-level simulation, the synthesized version of hardware designs are used. Therefore, this simulation is very useful for timing analysis of HW circuit. However, the achievable simulation time is very slow, particularly when the circuit is quite complex. As a result, this simulation is intended only for HW verification in component (unit) level, not for system-level evaluation.

2.2.2 HW/SW Co-verification Requirements

In order to achieve efficient and effective simulation task, the following requirements need to be considered as trade-off metrics by HW/SW co-verification framework:

1. **speed** is one of the most critical requirements to enable fast design exploration and also cover large test scenarios. Main issues that affect maximum achievable simulation speed are the interaction between HW /SW component and and also data synchronization of HW/SW. Hence the design of HW/SW interface will be critical.

2. **simulation accuracy** is an important metric to decide the best design alternative that satisfies requirements. To avoid many iterations of feedback, particularly in very late stage, accurate relevant results should be obtained to assess system performance in early stage development. These include cycle timing processing, fixed-point error rate, and others metrics. Cycle-accurate simulation or stand-alone hardware platform

is typically used to evaluate final performance. However, this approach needs high effort and practically will be available in later design stage. Hence, in order to allow designers make modification in both the design and the specification in early stage, the HW/SW co-verification should provide higher accuracy of simulation results

3. **cover multiple abstraction levels**. In designing complex embedded system, different team of designers often do not have enough expertise knowledge in other's domain. Moreover, the simulation and verification usually are carried out using different framework. To address this practical issue, it is desirable to provide HW/SW co-verification platform that can be used many different group. Hence, the consistency of results can be guaranteed until final implementation. Furthermore, HW/SW co-verification platform can eliminate effort when any changes made in one domain.

2.3 Platforms for LSI verification

According to the implementation platform, co-verification methods can be categorized into three main groups, which are the software-based approach (Full-software), the hardware-based approach (FPGA prototyping), and the combination of hardware-software (HW-assisted/HW-accelerated).

2.3.1 Software-based Design Verification

The co-verification approach in full-software platform is mainly characterized with the implementation of testbench program and abstraction of circuit design in SW platform. Both two designs are running within host PC, as depicted in Fig. 2.4. This method includes the conventional one which is Register Transfer Level (RTL) simulation [7] or its extended version with addition of Bus Functional Model (BFM) [8] or Transaction Level Modeling (TLM) features. This approach is able to simulate hardware model at cycle-accurate and also offers the designers to observe any signals in any levels of design hierarchy. Additionally, it is easy to build the verification environment in this platform since it only needs the simulation models. However, the run-time verification is very slow where the simulator can only be run at a speed of about 10 - 100 Hz. For verifying such complex SoC

circuits, it could take several days to simulate within the RTL simulator. Therefore, the RTL simulation can barely be used for complex hardware verification and embedded system verification. This approach is suitable particularly for block-level simulations, instead of whole system of complex-circuit simulations.

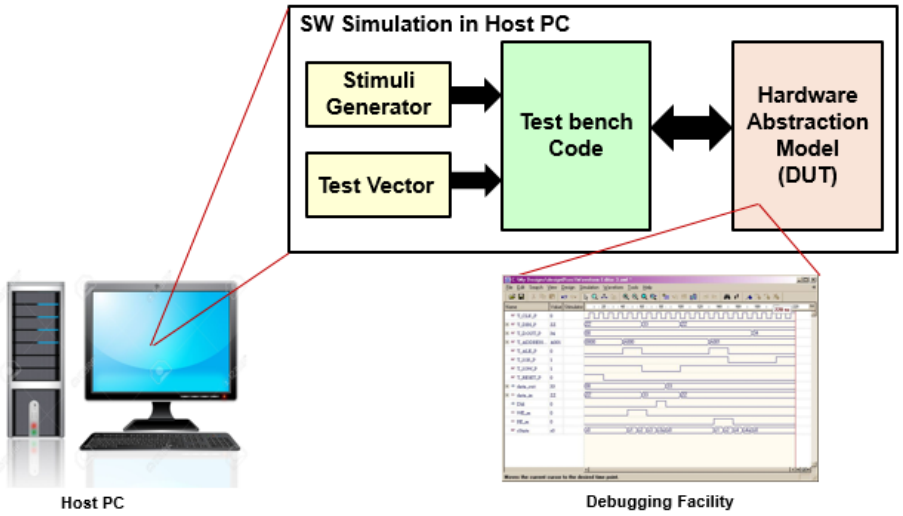

Figure 2.4: Full-software co-verification Architecture

In order to address the limitation of RTL-based simulation, the higher-abstraction model such C/C++ or systemC are employed[6], [7]. These simulation frameworks run much faster than RTL simulation. Additionally, in order to rapidly identify the errors in system functionality, various verification techniques such as assertion-based or formal method could be included in the software testbench. Moreover, these approaches also can be easily integrated with system-level simulator. Thus, evaluation of overall system can be performed more comprehensively. However, we cannot evaluate the timing processing performance since the simulation is not cycle-accurate. As a result, the accuracy of system is far from real-world condition.

2.3.2 FPGA Prototyping for Design Verification

Another approach is HW-based platform using a hardware emulator in re-configurable devices, such as FPGAs and GPU. In this method, both the testbench and DUT are realized in

hardware platform, as shown in 2.5. The testbench sequence could be implemented in dedicated circuit or implemented on executable code within on-circuit CPU, while the DUT is fully implemented in hardware resource block. The connectivity of host PC is required only for initial system configuration setup, such as writing bit configuration into hardware target.

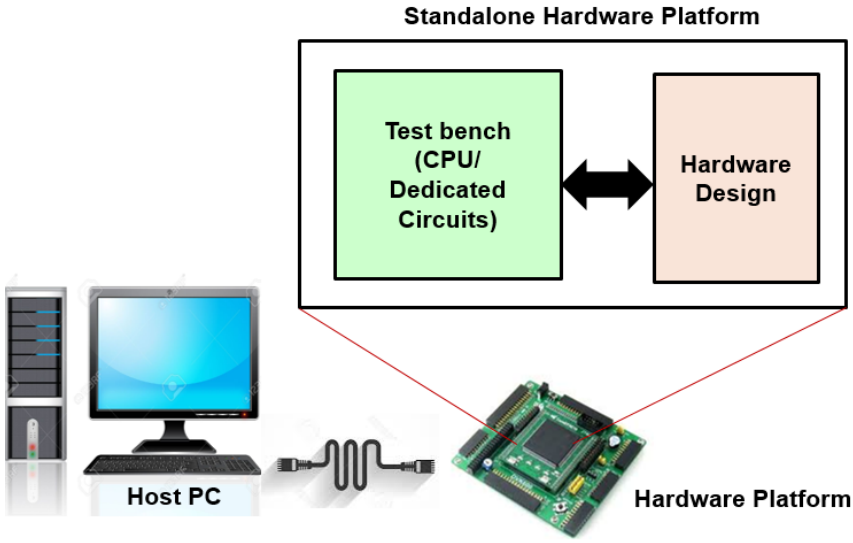

Figure 2.5: Co-verification Architecture in FPGA Prototyping-based

Recently, The FPGA prototyping has been intensively studied and applied in LSI system development, particularly in wireless communication system. With the availability of affordable FPGA platform, the FPGA prototyping has attracted as a tool for emulating complex systems since it has main advantage on improving verification time dramatically. The state-of the-art FPGA technology can be operated in several hundreds MHz of clock frequency. This feature not only offers the possibility to perform fast verification, but also realizes real-time circuit verification. Several considerable works have shown the capability of wireless communication system, both in block component level or full-system level, such as in [9]-[13]. Furthermore, FPGA-based verification is also able to perform simulation in cycle accurate. However, the signal observability and debugging instrument are very limited and not flexible. In order to provide high degree of observability, the designer should develop and add a dedicated circuit to capture various signals with considerable

efforts, such as Signal-Tracing Technique [].

From the observations of many cases, implementation of all component of a complex circuits into full hardware can not be performed straight forward since frequently the designers experience several problems, such as lack of resource block, timing constraint problem, or other technical issues. Furthermore, implementation of full prototyping into hardware platform limiting the intervention of user for assessing the system, unless the dedicated software or firmware have been implemented inside hardware circuit within a CPU module. To address this challenge, it needs experienced resource as well as extensive labor time for system development until the employed system are ready to be used for system verification. Hence, this verification platform is only suitable for final stage of system development as an product outcome, rather than integral part of system development.

2.3.3 Hardware Assisted Design Verification

With the increasing complexity of modern embedded system design, pure-software and full-hardware verification are limited used in system verification, particularly in system design with intensive HW/SW co-design process. As a consequence, the verification paradigm has shifted into HW/SW co-verification framework since the task processing is not only dominated by the hardware part, but also includes the software part. In order to leverage the capability of FPGA-platform, recently the combination of HW/SW framework is considered as an alternative solution in circuit verification, instead of using full-hardware platform, as illustrated in 2.6

Recently, HW/SW co-verifications become typical approach for system evaluation. The scopes of HW/SW co-verification mainly focus on: 1) mixed-simulation of components in the different abstraction level, 2) integration of various system-level simulator into unified environment, 3) simulation speed up by reducing the overhead communication between software platform and hardware platform.

Another relevant issue on HW/SW co-verification is related to HW/SW partition. The partition is the task of allocating system functions into set HW or SW resources. Various formulations for task partition can be carried out according to 1) **architectural assumption** : degree parallelism, the type of communication flow between each function, etc. 2)

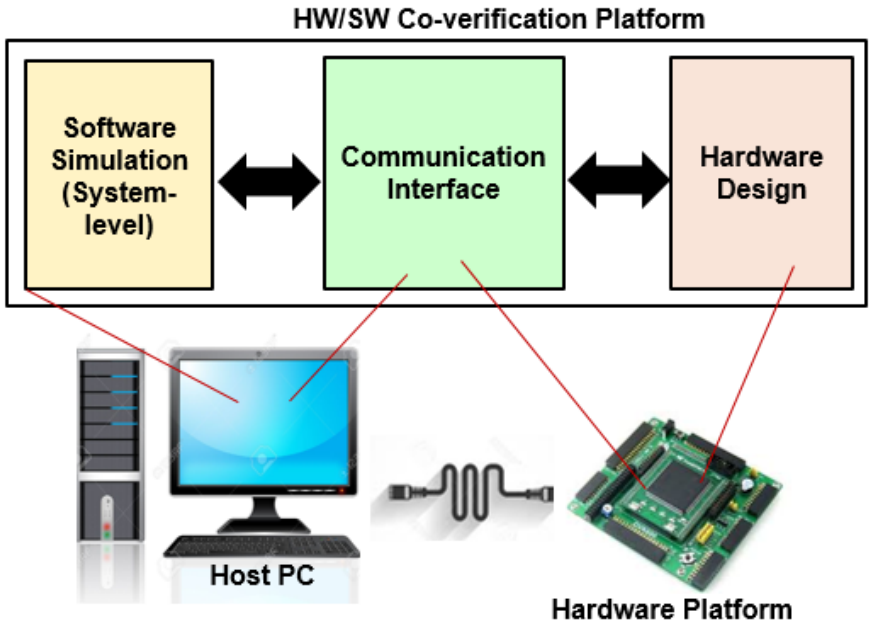

Figure 2.6: HW/SW Co-verification System

partitioning objectives : maximizing the overall speed up or minimizing the overall cost.

The HW/SW co-verification approach complements the conventional FPGA prototyping by offering more flexibilities for verification task, although the achieved verification time is not as fast as the performance of FPGA prototyping. These flexibilities include the facility for modification test scenarios, tight integration with system-level simulation in high-level abstraction, and ability to cover large coverage tests. Hence, the HW/SW co-verification can perform comprehensive system evaluation, particularly for complex circuit with various system parameters and many types of signaling between system layers.

The deployment of HW/SW framework recently is enabled by the availability of various interface platforms for connecting host PC and HW platform. Commercial EDA tool such as MATLAB provides Hardware-in-the loop option to perform co-simulation of HW design within Simulink Environment, as presented in [15] or [16]. Other platform such as TCP/IP based interface also one candidate for interfacing host PC and HW platform, as proposed in [17]. A HW/SW co-design framework with operating system support is also considered [18]

By employing HW/SW co-verification, it allows a mixed hardware and software based execution and offers several advantages:

- Avoid to develop different model (re-model) of the design which already available in early system-level modeling or RTL. This may reduce the development cost and enable hardware-assisted verification
- Hardware prototyping can be deployed in faster time because the system testbench has been created earlier, already verified and mature as it applied in previous stages.
- The trade-off between accuracy, speed and run-time can be managed more flexibly.

2.4 HW/SW Communication Interface

HW/SW communication interface is one of important component in HW/SW co-verification in order to allow concurrent design verification for both design domains, e.g HW and SW. The communication interface also bridges the gap between different abstraction layers and provide data adaption from one layer to other layers.

The abstraction of communication model should handle two different parts : one at SW side and the other one in HW side. In SW side the communication interface can be consisted two different layer. The upper layer is API model that serves as interface between user application and the lower layer is hardware-dependent software. The hardware-dependent software model will be different, depending on timing characteristic of hardware interconnection. In general, usually hardware vendor provide bus driver software for handling data transfer from API software to hardware platform. On the other hand, in HW side the communication interface hides the detail of bus protocol through adapter interface which is called as bus interface. This include I/O modules such as memory, FIFO, or registers.

In simple point of view, the communication interface between SW function and HW component provide a channel for data from source node to destination node. The communication channel can be modeled with different abstraction layer, depending on implementation of HW/SW co-verification platform. In data-flow simulation, this communication

interface could be directly mapped as argument for function call as an explicit connection. This communication as a port wiring in hardware side. In more detail abstraction, this HW/SW interface can be modeled as transaction based model in a dedicated function call. The function call usually represents data transfer from one node in SW to other destination node in HW or from one source node in HW to destination node in SW.

The generic architecture of HW/SW communication interface is shown in Fig. 2.7.

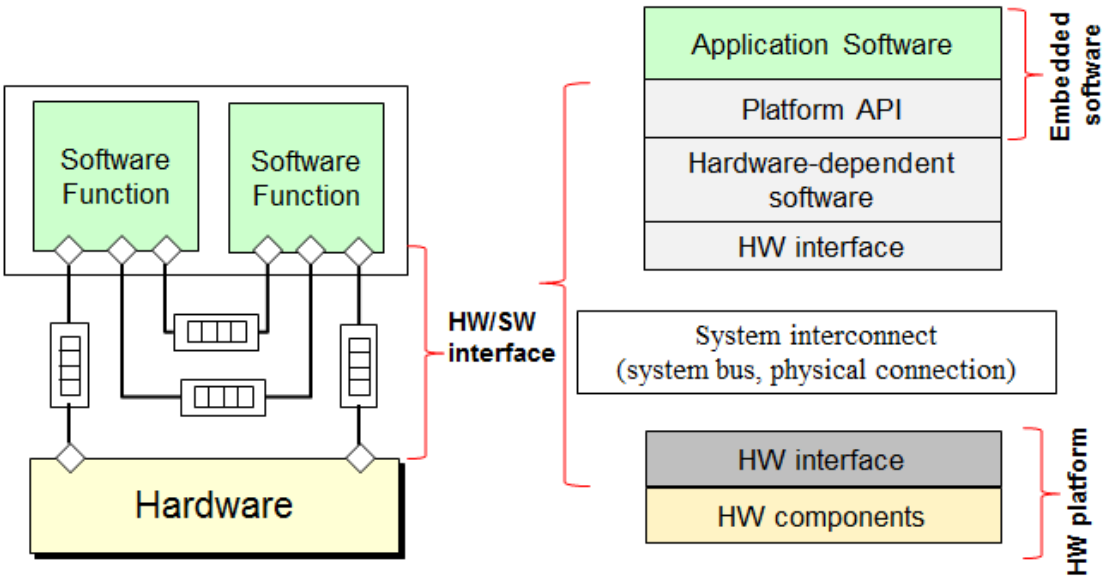

Figure 2.7: HW/SW Communication Interface

2.5 Summary

As can be observed in the previous description, the HW/SW co-verification method has many promising advantages compared to the conventional ones, which are fully-software simulation and FPGA prototyping. The properties of these methods are provided in Table 2.1.

The HW/SW co-verification methods is the optimal option for verification of complex circuits, considering the trade-off between system flexibility, verification run-time,

Table 2.1: Comparison Co-verification Methods

Properties	SW Simulation		HW Prototyping	HW/SW Co-verification
Model/Platform	RTL	C/C++,SystemC	FPGA	HIL
Run-time	Slow	Fast	Real-time	Fast
Accuracy	Cycle accurate	Cycle approximate	Cycle accurate	High
Flexibility	Moderate	High	low	High
Observability	Full	Full	Limited	High
Cost Development	Low	Low	High	Moderate

and the development effort. However, the existing HW/SW co-verification has limited the integration with system level simulation and suffer to be implemented as integral part of system development. Hence, the simulation cannot cover verification task in various design levels concurrently, which are: algorithm validation, RTL simulation, and physical verification (in circuit verification). Finally, in order to address these requirements, the unified HW/SW co-verification framework is proposed to leverage the existing HW/SW co-verification methods.

Chapter 3

Unified HW/SW Co-verification Methodology

3.1 Scope of the framework

In order to realize an efficient co-verification platform, this thesis proposes an effective approach to obtain reliable and efficient development of large scale systems. An efficient HW/SW co-verification platform should not only be capable of performing a fast simulation, but also at the same time it must have:

1. *Flexibility* : support quick turn-around design modification and design extension,
2. *Large coverage* of verification task : cover from algorithm development to hardware implementation,
3. *Tight-integration* with system-level simulation : maintain reliable system-level performance.

In general, VLSI design and verification flow consist of three design layers, as depicted in left side of Fig. 3.1. In the first layer, a complete system modeling is developed in algorithm/mathematical abstraction using system level language such as MATLAB, C/C++, etc. For example, in wireless communication system, it includes model abstraction of transmitter, channel model, and receiver. Since blocks in wireless communication system,

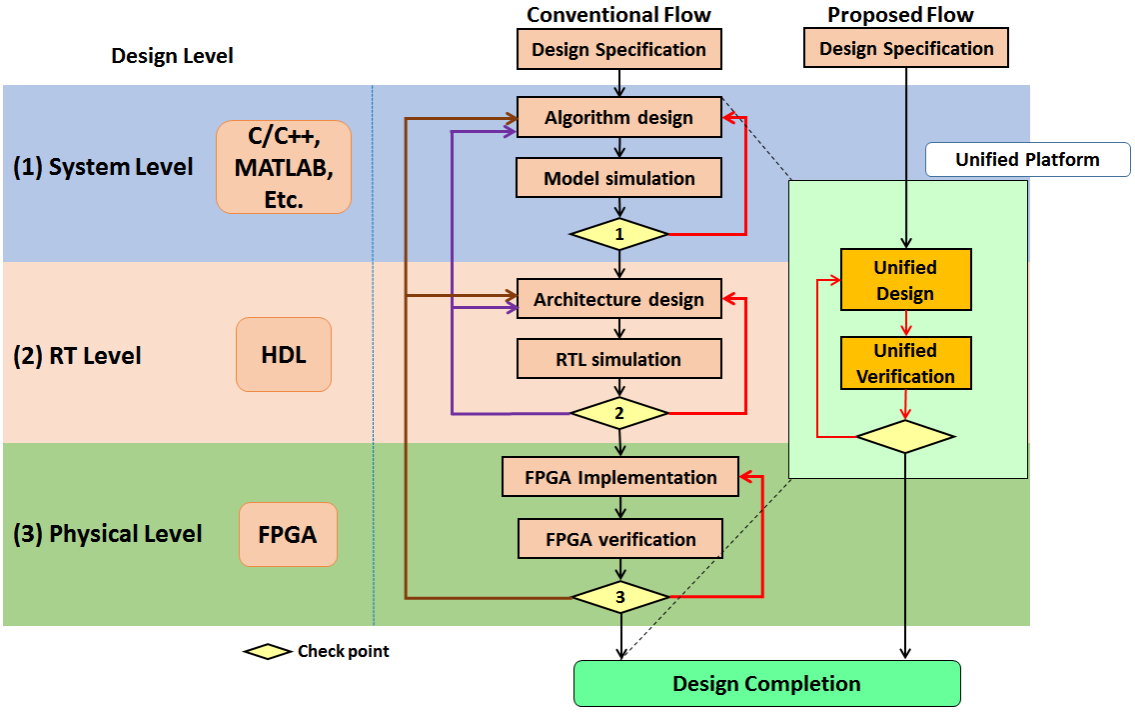

Figure 3.1: VLSI Design and Verification Flow

especially in receiver, are very sensitive to the employed algorithm and hardware optimization, before translating into hardware design, the designed algorithm should be verified in order to get realistic computational complexity and predictable performance degradation.

After algorithms have been validated, RTL design for hardware implementation can be created depending on hardware target. The process of algorithm transformation, recently, not become a difficult task since the availability of advanced High Level Synthesis CAD tools [19]. A study and evaluation of various HLS tools, including model based RTL design, are thoroughly presented in [20]. In this HLS methodology, the hardware abstraction is created using graphical form that facilitate system development conveniently. This method offers some features, including parameterized design, area and timing optimization options. These features allow the designers to carry out fast design exploration and lead to reduce development time significantly. In this RTL design stage, the verification is also performed to verify that hardware design in generated HDL code is still have same functionality as defined in system level simulation. However, verification time for bit true

model in this CAD environment is very slow. Once the RTL is obtained, hardware implementation can be carried out and once again the verification is carried out in order to ensure that final hardware implementation satisfy required performance.

In the conventional verification system, the verification process of each stage is carried out independently and is also not integrated to system level simulation. Additionally, to verify and to evaluate overall system performance, it needs to implement all blocks into hardware emulation in order to obtain fast verification results. With this approach, all hardware design of overall system should be completed before performing verification. Moreover, another potential problem will be faced when we directly implement a full system, such as lack of FPGA resource or timing problem.

As summary, there are several problems from the conventional verification methodology. The first one, conventional verification process may contain many iteration loops, either within same design layer or different design layer. This process takes longer verification time and gives slow feedback for design modification. As a result, development process requires longer time. The second one, because the verification process is independent between design layer, the verification environment in each layer cannot guarantee the consistency of performance in the point of view of system level simulation. Hence, the expected performance cannot be maintained from system level design into final system implementation.

In our proposed verification platform, the verification of complex system can be carried out efficiently from block component up to system level by employing unified HW/SW co-verification platform. In the proposed scheme, we can use multiple of abstraction levels of design. In particular, one component of whole system can be simulated in physical level, while the remain blocks are simulated in other levels, MATLAB or RTL level.

To realize such system, tight integration of hardware platform with system level simulation is a key element. The proposed verification platform can be used by hardware designers to design, implement, and verify related block concurrently. Furthermore, the verification of each block can be performed in the point of view of system level simulation. Hence, the final performance requirements of full system can be maintained and predictable. Moreover, the verification time can be reduced significantly.

3.2 Data-driven Simulation

A key feature in proposed verification method is data-driven simulation. In this approach, data processing could be performed in vector based (burst data). Vector based processing is considered since the provided data from system level simulation (e.g MATLAB) is matrix/array-based format. To realize this feature, we provide communication interface that support block data transfer. It includes SW interface and HW interface that support block data transfer. The SW interface can perform burst mode transaction, while HW interface able to handle block data transfer by employing block RAMs or FIFOs. By utilizing this approach, the interaction of hardware software only occur in the beginning and the end of verification run-time. Therefore, the overhead of HW/SW interaction can be reduced significantly, which results significant improvement of HW/SW co-verification run-time.

Furthermore, one of of significant benefit is we do not need intensive synchronization for data transfer in the communication channel. Since the data transfer is performed based on availability of data, it can be synchronized only by simple interrupt-based mechanism. Therefore, the implementation complexity in both software and hardware can be maintained.

This approach is different with HIL verification methodology in [15]. This work employ HIL within Simulink to verify a complex circuit (e.g FFT design). While the Simulink environment can provide more convenience verification, however, the data processing is carried out in cycle-based simulation and time-driven based. This means that the data processing is performed one-by-one data involving intensive interaction HW/SW communication. Thus, this method introduces significant overhead for HW/SW communication for each cycle and affects overall verification speed-up, particularly in high complex circuits.

As illustrated in Fig. 3.2, we can characterize the performance of data-driven simulation and time-driven simulation as follows. First, we assume that the data-driven simulation involves the interaction between HW and SW in the beginning and the end of verification run-time. On the other hand, the time-driven simulation involves the HW/SW interaction in each cycle. HW/SW interaction task takes T_{comm}. Therefore, the total HW/SW communication overhead for data-driven and time-driven simulation are $2T_{com}$ and $2NT_{com}$, respectively. The communication speed and penalty cycle of interrupt handling contribute to

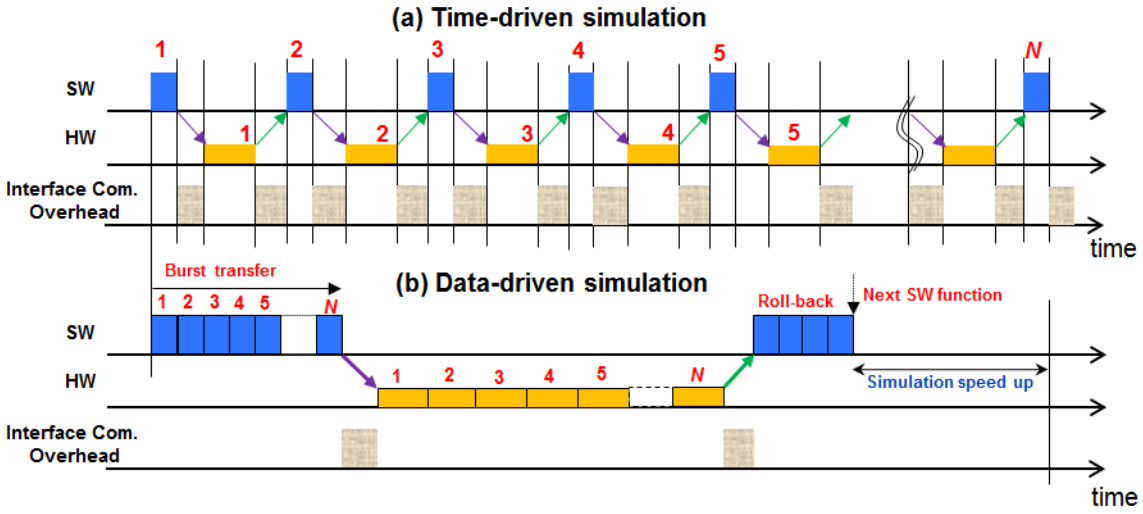

Figure 3.2: HW-SW interaction in verification process: Time-driven simulation (upper), Data-driven simulation (lower)

this communication overhead. For simulating N data, the total hardware processing for the both simulations are same, which is T_{HW}. Hence, the total of verification run-time for data-driven simulation, T_{DD}, and time-driven simulation, T_{TD}, can be calculated as provided in the following equations.

$$T_{DD} = N(T_{SW} + T_{HW}) + 2T_{com} \tag{3.1}$$

$$T_{TD} = N(T_{SW} + T_{HW} + 2T_{com}) \tag{3.2}$$

We can see that the HW/SW communication overhead in data-driven simulation does not longer depend on the number of simulated data (e.g cycle). Therefore, for simulating very large data, the data-driven simulation will offer significant advancement over time-driven simulation. Moreover, by employing our interface, SW data transfer can be performed in high speed transfer, achieving up-to 800 Mbps. Thus, the overhead for HW/SW interaction in the proposed method will also be reduced significantly. As a results, the verification time in the proposed method is much faster than the time-driven simulation.

The data-driven simulation might have a limitation of memory buffer size used for burst data transfer. Principally, the buffer size depends on specification of test vector and the number of I/O ports of DUT. In practical, the boundary limit of buffer size is the remaining

available memory blocks in FPGA target after implementation of hardware core of DUT.

3.3 Task Partition Methodology

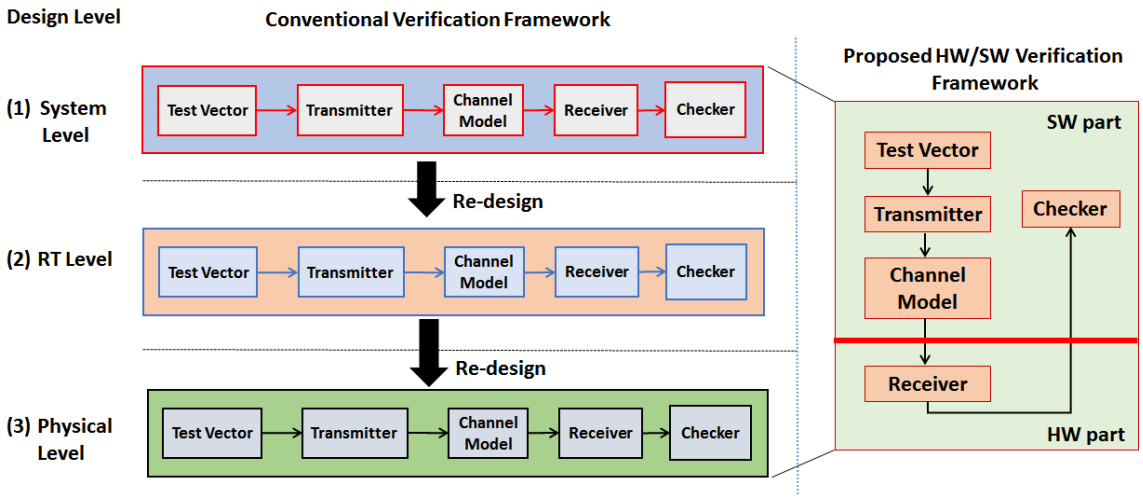

Figure 3.3: Example of verification framework

Typically, the design process of a complex signal processing system starts from system level algorithm description, such as MATLAB or C/C++. There are many possible algorithms for implementation, in order to fulfill system requirements, but they give different trade-off between area complexity, efficiency, flexibility, and design effort. Hence, design exploration is mandatory and should be performed quickly at the initial development. Once the algorithm is selected, a submodule can be transferred to hardware development and further verified in the point of view of system level simulation.

The first step to build an efficient HW/SW co-verification system is performing task partitioning of all system process. The task partitioning can be carried out under consideration of area complexity, timing processing, requirement of quick algorithm evaluation, or any design metrics that are determined by system requirements with subject to maximize the verification speed-up and to minimize design effort.

For example, as depicted in Fig. 3.3, signal processing blocks of wireless communication system consist of several consecutive processes: input data and parameter (referred

as Test Vector), Transmitter, Channel Model, and Receiver. Assumed that we perform task partitioning to the system by selecting a complex process (e.g Receiver process) to be simulated in hardware platform, and the rest of processes are simulated in software platform. Furthermore, the transmitter, channel model, and output results checking could be implemented into different software abstraction language. The transmitter and channel model process are implemented in MATLAB, while output checking is implemented using C/C++. Both software platforms, which are MATLAB part and C/C++ part, communicate each other through custom transparent layer communication (API). On the other hand, receiver process, as considered the high complexity system, is implemented in hardware emulated platform.

At the initial stage design, it is possible to implement receiver partially in hardware and allocate other receiver processes in software. When the development stage is growing, more tasks in software processing could be added up to the hardware target and also can re-utilize the verified hardware blocks, achieving a complete full hardware emulation.

3.4 HW/SW Co-verification Platform

3.4.1 General Architecture HW/SW Co-verification

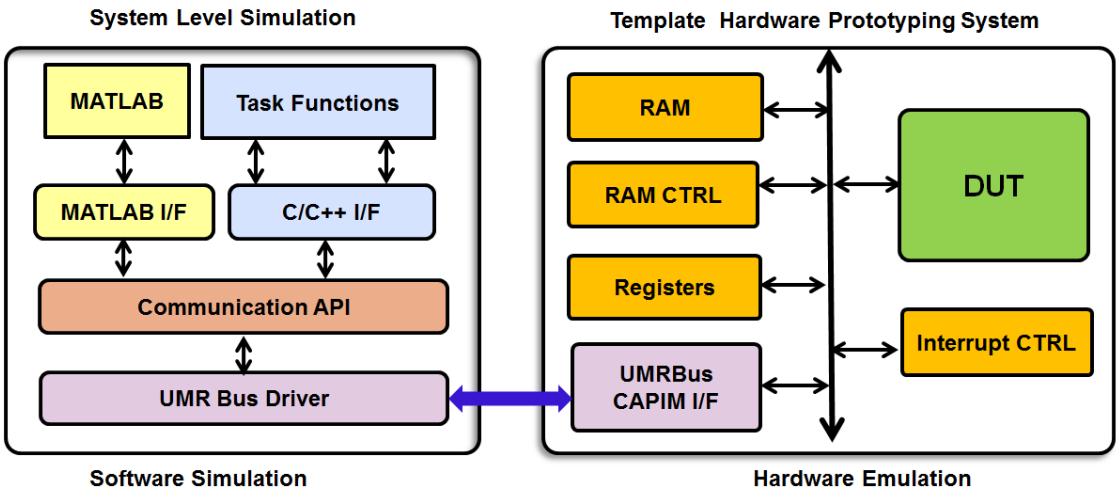

Figure 3.4: General HW/SW Architecture

To implement a complete design of HW/SW co-verification, first we have to provide a generic architecture for HW/SW implementation, as shown in Fig. 3.4. The HW/SW design should accommodate flexibility and reconfigurability purpose. Hence, it could be reused for other design targets or applications. This generic architecture also reflects typical SoC point of view, that is constructed from master CPU (host PC), bus (communication interface), and slave as the hardware design target. The hardware design mainly consist of three building blocks, which are: (1) bus interface module for receiving data from end-point physical connection link (e.g. PCIe cable), (2) memory banks for storing input and output stream, and (3) hardware target that is being verified (Design Under Test). On the other hand, the software design consist of 3 main blocks, which are: (1) system level design that performs system level simulation, (2) communication API that handles data communication between software layer, and (3) bus driver that connects data communication of software part and hardware part.

The employed HW/SW platform uses HAPS board from Synopsys [21]. Basically, the original Synopsys HAPS (UMR bus) [22] is aimed for prototyping platform and is not intended for hardware-assisted simulation. Since it cannot be directly connected to higher abstraction of simulation, we provide the API design to extend the UMR Bus function in order to allow the data flow between MATLAB and bus driver software. The second one, the nature of UMR bus is single data transfer. Therefore, we improve the interface, both in hardware in software, to support burst transfer and also capable for parallel I/O connection.

On the other hand, the main objective of our proposed methodology is for unified verification, that covers all design abstraction layers. Our proposed methodology seems similar with Synopsys Hybrid Prototyping Platform [23]. It can also perform system level simulation by utilizing virtual prototyping. However, our hardware platform does not include virtual prototyping packages. Therefore, system level simulation could not be carried out in employed HAPS board. Furthermore, the TLM verification flow in virtual prototyping is being a commercial package which is not an open access package. Additionally, the TLM based verification concept primarily suitable for System on Chip (SoC) prototyping case, where the data communication through on-chip bus among various modules are very important.

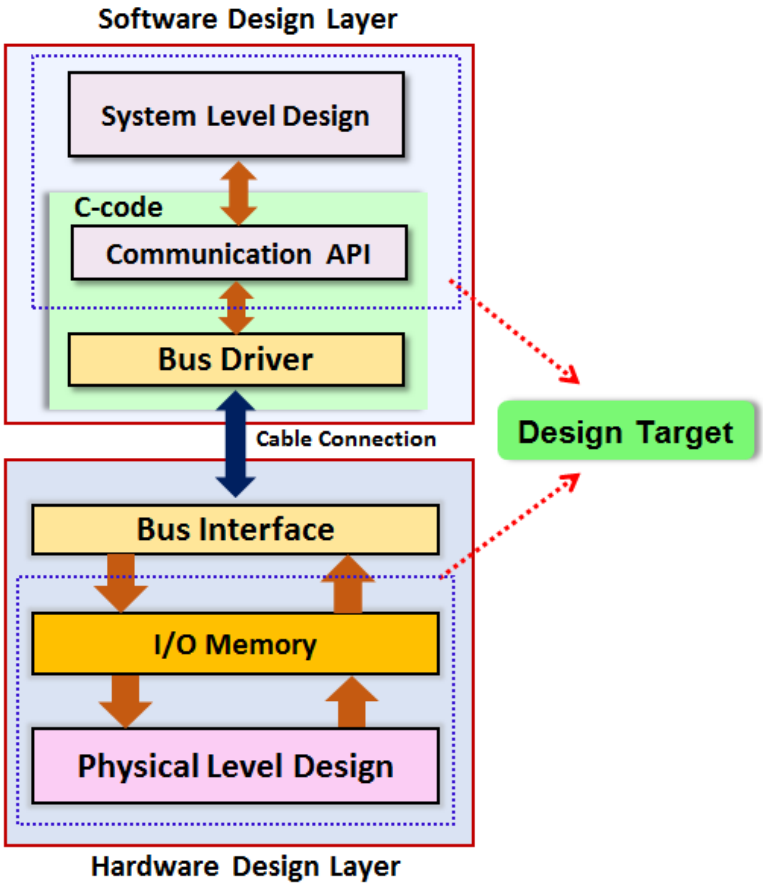

Figure 3.5: Unified HW/SW design layer

3.4.2 Flexible and Configurable HW/SW Architecture

In order to address the limitation of basic HAPS platform, we propose unified hardware and software design approach to realize unified HW/SW co-verification, which layer structure is depicted in Fig. 3.5. We also propose flexible and scalable interface, both in hardware and software part.

In hardware side, the flexible interface is employed to handle various transfer modes, for example stream-based mode or pass-through mode. Additionally, to support various interface type in different applications, we use configurable architecture for I/O management. The hardware interface, e.g CAPIM, receives the data from software driver through physical link and multiplexes this data based on specified address. If the address tag of received

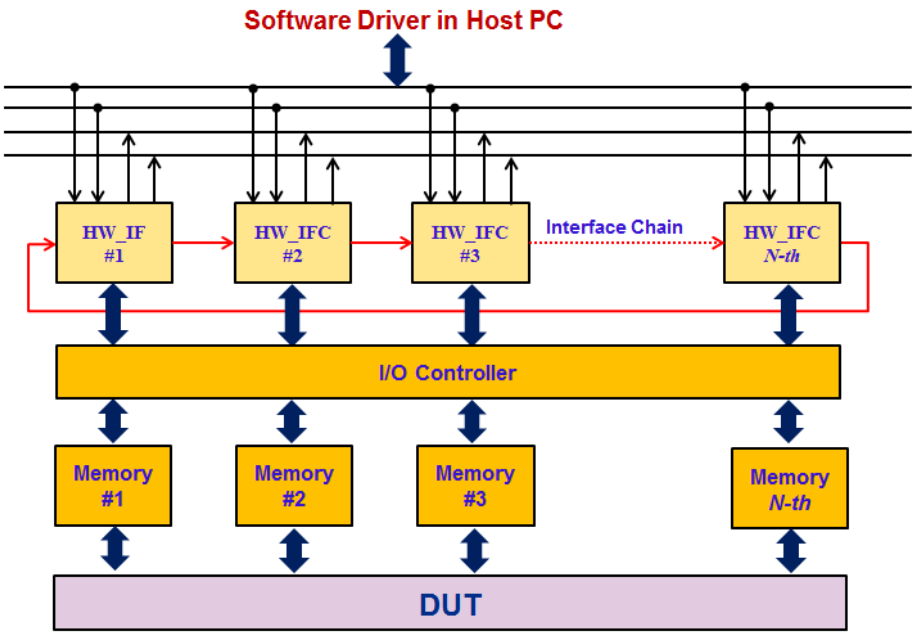

Figure 3.6: Hardware Interface Structure

data is identical with interface address identity (address ID), then the data will be written to connected memory or register. Otherwise, the data will pass through connected interface chain. Moreover, we also exploit bandwidth transfer by employing full-matrix cross bar connection. This allows the implementation of hardware interface module in fully parallel architecture. The generalized structure of hardware interface design is depicted in Fig. 3.6.

In software side, to adapt with high level system simulator, we employ customized API software as communication interface. This interface manages the received data from high level simulator or hardware circuit. Specifically, this API has to allocate the data that will be transferred to hardware from host PC. The API also receives and collects the data that are received from hardware for further processing in system level simulation. The designed API considers the flexible length of data transfer in order to enable burst data transfer. Furthermore, the designed API collaborates with software driver to perform data transfer into specific memory (or vice versa) by pointing designated hardware interface address of memory. API requires length of burst data as argument input, while SW driver uses the interface address as additional input argument. Several API codes are constructed into main program to perform whole co-simulation testbench program. Following listing

shows the pseudo-code of generic main testbench software.

```
// Pseudo code software Testbench
umrbus_init();
/* Write vector data to UMRBUS */
for idx=1:CAPIM_NUM
  write_data = read_vec_data(data_length, vecin_filename);
  umrbus_write (write_data, data_length, CAPIM_address);
end
/* Waiting for Interrupt */
if (interrupt_handler()==1)
  // Read data from hardware and send to Simulator
  read_data = umrbus_read(data_length, CAPIM_address);
  write_vec_data(read_data,data_length, vecout_filename);
end
```

3.4.3 Hardware-In-The Loop Co-verification System

In this work, the hardware platform uses HAPS platform (High-performance ASIC Prototyping Systems) from Synopsys [21], as depicted in Fig. 3.7. The HAPS system consists of 4 FPGA chips of Xilink Virtex6 family. It can occupy upto 7.5M gates for each FPGA chip. The HAPS system is connected to a host computer (PC) through PCI-e cable link.

The complete HW and SW design finally build a closed form of co-verification system, which is called as hardware-in-the loop (HIL) co-verification system. The generic structure of HIL system is depicted in Fig. 3.8.

By utilizing this HIL scheme, a system-level verification can be carried out in the following steps:

1. First, the system-level simulation will perform complete system simulation, as specified by system parameters from user. In the same time, the simulation will provide intended input data for DUT block.

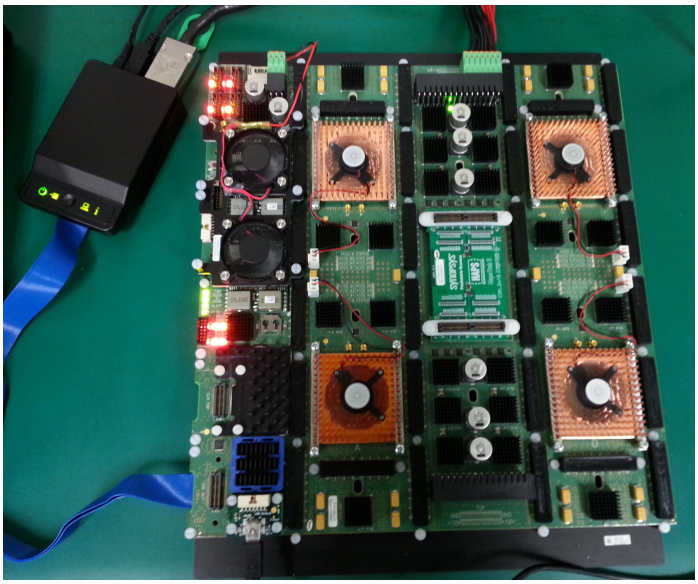

Figure 3.7: HAPS Hardware Platform

2. The input data for DUT is passed through vectors generator to create set of data vectors as required by DUT.

3. When generated data vectors are available, API software passes the re-formatted data to bus driver software and latter sends to the HW platform.

4. While the HW performs data processing, the testbench software waits for the interrupt signal from intended test point. As the interrupt signal is available, the software testbench reads out simulation data from the memory, as pointed by specified address. This simulation results are then sent back to system-level simulation for further system-level evaluation.

3.4.4 Methodology Comparison

In order to quantify the effectiveness of our proposed methodology, we summarize the comparison of several important features from different methodology/verification environment, as shown in Table 3.1.

Although high level simulation can provide all abstraction of system functionality and also requires low-effort for environment setup, however, the simulation results are too far

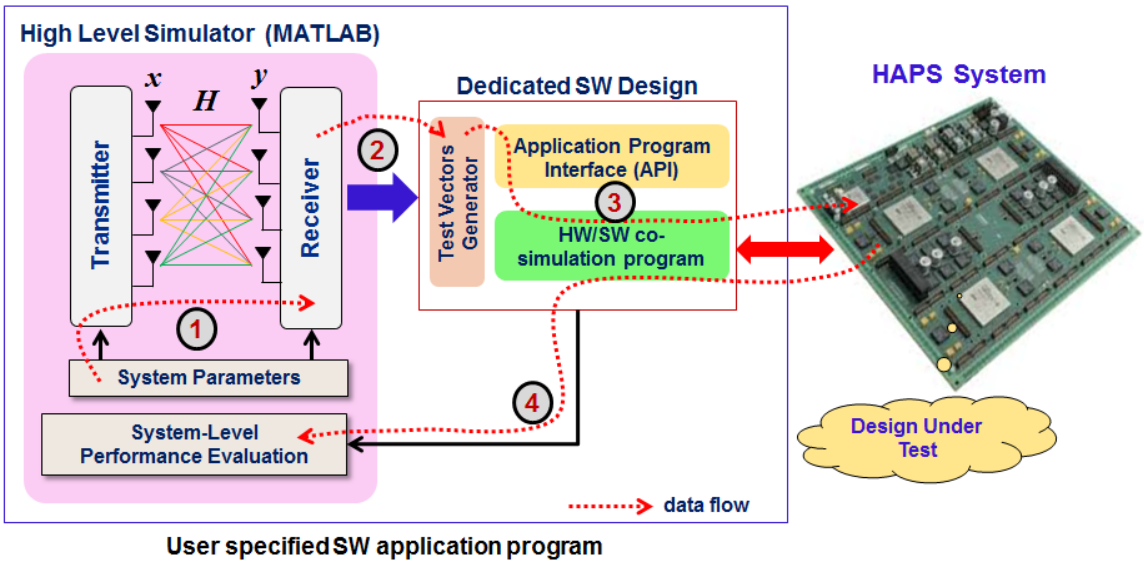

Figure 3.8: General HW/SW Architecture

from real hardware performance. In order to address this limitation, RTL simulation is carried out to obtain more accurate evaluation. However, performing verification of complex circuit and extensive computation using RTL simulator is prohibited due to very long runtime simulation, particularly for exhaustive verification involving various system parameters and many different design versions. To accelerate design verification and to achieve the optimum performance, the standalone FPGA prototyping is employed, such as in [9]. Unfortunately, the standalone FPGA implementation needs huge effort since it must implement all system into hardware part. Hence, the verification task could only be performed at very late stage of design development. Furthermore, this simulation approach is also less flexible for various test scenarios and difficult to maintain system level performance.

Recently, the hardware-in-the-loop is a promising solution for fast verification. It also has flexibility to support various system parameters. However, there is significant difference between our proposed work and the previous work of HIL system [15]. The HIL method in [15] employs Simulink environment that performs cycle-based simulation. Thus, the simulation is carried out as a time-driven simulation. This methods will introduce large overhead due to HW/SW interaction in each cycle simulation. On the other hand, our proposed verification is performed in data-driven based simulation that requires small overhead

Table 3.1: Comparison of Co-verification Methodology

Objectives/Metrics	**Synopsys Hybrid Prototyping [23]**	**HIL [15]**	**Proposed**
High Level Simulation			
a) Time-driven (cycle-based)	✓	✓	✓
b) Data-driven (vector-based	×	×	✓
RTL Simulation	×	✓	✓
FPGA simulation	✓	✓	✓
Unified HW/SW verification	✓	✓	✓
Verification speed	moderate	moderate	fast

on the HW/SW interaction. Therefore, the proposed method can achieve fast verification speed .

3.5 Summary

In this chapter, the co-verification methodology in unified framework is proposed. The proposed co-verification methodology includes:

1. The generic structure of HW/SW co-verification : reflects the recent SoC architecture and has flexible HW/SW interface.

2. Task partitioning methodology : allocates the task/function of overall system into implementation platform, either HW platform or SW platform.

3. Data-driven simulation within Hardware-in-the Loop co-verification : improves verification time significantly over the HIL platform in [15].

The proposed unified methodology has several main features, which are: (1) high flexibility to support fast turn-around on design modification and design extension; (2) ability to cover various stages of verification task (from algorithm validation to hardware implementation); (3) tightly-integrated with system-level simulation to maintain reliability of system performance.

Chapter 4

Fast Co-verification and Design Exploration in Complex Circuits

In this section, we describe one application example of the proposed unified framework discussed in Chapter 3, particularly for supporting fast co-evaluation and design exploration of complex circuit. The description covers algorithm-architecture translation, efficient architecture design, and building efficient verification framework. The MIMO decoder circuit of high throughput wireless system is selected to represents complex circuit. Furthermore, the relevant performance metrics are also presented.

4.1 Overview of MIMO Decoder in High Throughput Wireless Communication System

Multiple input multiple output (MIMO) wireless communication system is one of technology breakthrough in wireless communication system. Recently, this technology has widely adopted for providing high-data transmission rate. In WLAN standard family, the MIMO technology has been introduced since the deployment of 802.11n standard. Using MIMO transmission scheme the physical layer transmission rate can achieve upto 600 Mbps. MIMO transmission scheme utilizes multiple antennas both in transmitter and receiver to obtain the performance gain, which are spatial multiplexing gain and diversity

gain.

In order to increase system reliability, the MIMO scheme employs spatial diversity. In this scheme, the copy of data are transmitted through multiple antennas and will experience different paths. In the receiver, these multiple independent signals are utilized in decoding process in order to improve transmission reliability. On the other hand, in order to increase transmission rate (e.g. throughput), MIMO scheme sends the different data into each transmit antenna by using multiplexing scheme. Hence the obtained performance gain refers to spatial multiplexing gain. For single link communication with the number of transmit antennas N_T and the number of receive antennas N_R, the maximum diversity gain is $N_T N_R$, while the maximum spatial multiplexing gain is $\min\{N_T, N_R\}$. However, we cannot achieve both the maximum diversity gain and maximum spatial multiplexing gain in the same time. Hence, there is a trade-off between these two performance gains [24].

In this thesis, the main objective is to implement high throughput wireless system. Hence, the MIMO system is realized through spatial multiplexing scheme. We consider a MIMO wireless communication system with N transmit antenna and N receive antenna. The transmit symbol is taken from a quadrature amplitude modulation(QAM) which has 2^M constellation points. M denotes the modulation order. For simplification, the employed MIMO system is shown in Fig. 4.1.

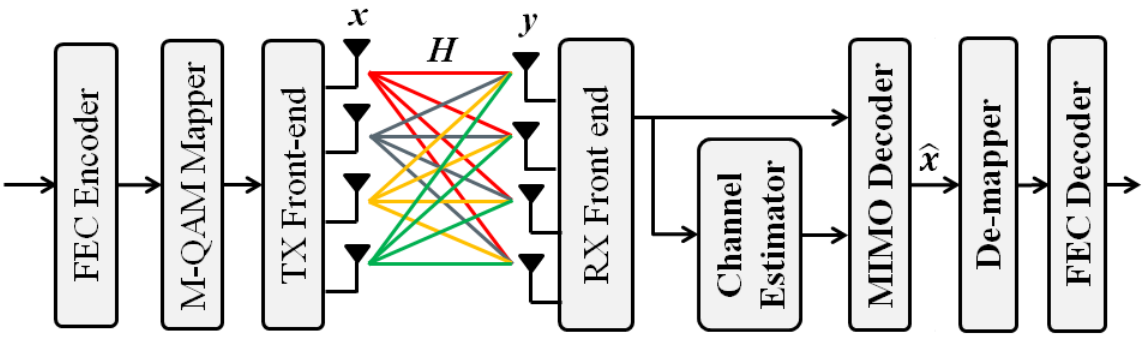

Figure 4.1: *NxN* MIMO Communication System Model

The transmission of each vector **x** over flat-fading MIMO channels can be written as

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{n} \tag{4.1}$$

where $\mathbf{x} = [x_1, x_2, ..., x_N]^T$ is the transmitted signal vector, $\mathbf{y} = [y_1, y_2, ..., y_N]^T$ is the received signal vector, $\mathbf{H}$ is the $N\,x\,N$ channel matrix, and $\mathbf{n} = [n_1, n_2, ..., n_N]^T$ is independent identically distributed Gaussian white noise vector.

The MIMO decoder block plays an important role in MIMO wireless communication system because the BER performance is highly dependent on employed MIMO decoder algorithm. Many researchers have investigated several techniques that are feasible for practical implementation, such as Zero Forcing (ZF) [25], maximum likelihood detection (MLD) [26], linear minimum mean square error (LMMSE) [27], Bell Labs layered space-time MMSE (BLAST MMSE) [28], and lattice-reduction aided MMSE (LRA MMSE) [29].

ZF algorithm and MMSE algorithm estimate the received signals based on the inverse matrix, $\mathbf{H^{-1}}$. In the case of the inverse matrix can not be found, the MIMO detection could employ the pseudo inverse matrix, $\mathbf{H^{+}_{ZF}}$ and $\mathbf{H^{+}_{MMSE}}$. As ZF MIMO detection doest not consider the contribution of noise, the MMSE MIMO detection includes the noise variance, σ_n^2, as the correcting factor for weight calculation in MIMO detection process. The ZF and MMSE MIMO detection algorithms can be expressed as follow.

$$\hat{\mathbf{x}} = \mathbf{H^{-1}y} \tag{4.2}$$

$$\mathbf{H^{-1}} \approx \mathbf{H^{+}_{ZF}} = (\mathbf{H^{H}H})^{-1}\mathbf{H^{H}} \tag{4.3}$$

$$\mathbf{H^{-1}} \approx \mathbf{H^{+}_{MMSE}} = (\mathbf{H^{H}H} + \mathbf{I}\sigma^{\mathbf{2}}_{\mathbf{n}})^{-1}\mathbf{H^{H}} \tag{4.4}$$

where, $\mathbf{I}$ is the identity matrix.

In order to improve the detection capability, BLAST-MMSE and LRA-MMSE have been proposed. The BLAST-MMSE performs a first detection of the most powerful signal using MMSE scheme, and then considers the estimated signal as an already-known noise before detecting the next signal. Because the most powerful signal is selected to detect first, the probability to detect received data correctly is high. Thus, the performance of this scheme is better than ZF and MMSE.

On the other hand, LRA-MMSE MIMO detection attempts to find the orthogonal form of the channel matrix $\mathbf{H}$ before applying MMSE for detection. By performing such calculation, the input channel matrix of the MMSE detection is expected to be orthogonal, thus the detection capability of the MMSE is improved as compared to case of non-orthogonalization.

4.2 MIMO MLD Algorithm

The MLD technique is considered as the optimal technique for MIMO Decoder. In this algorithm, estimated transmitted signal is calculated among all the candidates, as denoted in Eq. 4.5

$$\hat{\mathbf{x}} = \underset{\mathbf{x}\in\mathbf{\Omega}}{\operatorname{argmin}} \quad \|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2 \tag{4.5}$$

Hence, with the increasing of constellation point of the modulation scheme and the number of spatial streams, the computation complexity in MLD MIMO decoder becomes extremely high and increases the hardware complexity. Additionally, this requires huge effort on the verification. To deal with this issue, some studies, including its verification, have been proposed. In [9], the authors propose FPGA Implementation of real time MLD MIMO decoder that support for QPSK modulation in 4x4 MIMO system. In higher modulation order, e.g. 64-QAM MIMO system, ref. [10] describe FPGA prototyping of quasi MLD MIMO Decoder. Another implementation approaches of MIMO decoder, such as A GPU-based implementation and Application Specific Instrument-set Processor (ASIP) are also considered, as presented in [30] and [31], respectively.

In order to prove applicability of the proposed verification platform into complex circuits, we consider Full-MLD MIMO decoder for IEEE 802.11ac WLAN system, as a case study. The main motivation relies on several aspects.

1. First, we need to verify the complex circuits where the simulation time in SW based simulation takes excessive time (very long or impractical). This is to judge the efficiency of the proposed methodology in terms of verification time speed-up. Hence, the proposed methodology is eligible. The designed MLD MIMO decoder should support up to 256-QAM modulation scheme. Because of its huge complexity, design a real-time implementation and make efficient verification is still a big challenge. For example, we have carried out the calculation of timing processing of highest complexity MLD parameter in 4x4 MIMO carefully. It is found that the verification of one packet data, consisting 2 OFDM symbols, verification using MATLAB tools takes around 70 days. On the other hand, assisted hardware platform only takes 4

minutes.

2. One of the most complex circuit in MIMO WLAN system is MIMO decoder, where the most optimum algorithm is Full MLD algorithm. Since, we also need to know the performance of such algorithm in a real hardware implementation for a benchmark (reference), we have to implement this algorithm and have to evaluate its performance. For higher order modulation and large antenna numbers, it is seem impractical to simulate this algorithm in full SW simulation. However, in the final product deployment other algorithms such as K-best and Sphere decoding would be more attractive, considering the trade-off between performance and cost.

To reduce computation complexity, we employ QR decomposition into channel matrix, **H**, that are provided by Channel Estimator block. Firstly, we decompose the matrix **H** into two matrices **Q** and **R**, where **Q** is the unitary matrix and **R** is the upper triangular matrix. With **H = QR**, Eq. 4.5 can be written as

$$\mathbf{z} = \mathbf{R}\mathbf{x} + \mathbf{n}' \tag{4.6}$$

where $\mathbf{z} = \mathbf{Q}^{\mathbf{H}}y$ and $\mathbf{n}' = \mathbf{Q}^{\mathbf{H}}n$

Then, the output MLD, $\hat{\mathbf{x}}$, can be calculated by searching among all candidates such that resulting minimum magnitude of error signal, as provided by Eq. 4.7.

$$\hat{x_E} = \arg\min \sum_{i=1}^{\Omega} ||D||^2 = \arg\min \sum_{i=1}^{\Omega} ||z - Rx||^2 \tag{4.7}$$

where Ω is number of all candidate which is 2^{2MN} and x_E referred as Euclidean distance calculation. In this paper, we consider MIMO system with parameters as provided in Table 4.1.

Table 4.1: System Parameter

Parameter	**Value**
Wireless System Standard	IEEE 802.11ac
Number of Antenna (N)	4
Modulation Type (M)	QPSK (2), QAM16 (4), QAM64 (6) and QAM256 (8)
System Bandwidth	80 MHz

4.3 Design Exploration in Wireless Communication System

When we consider employing multi metrics in system development, several design versions or configurations should be evaluated in order to obtain optimum implementation [32], [33]. The evaluation are carried out by examining the trade-off among several design constraints. This process is called as design space exploration. In the latter, we use design exploration term instead.

The process of design exploration can be illustrated in Fig. 4.2. In developing LSI circuits for wireless communication systems, design exploration is unavoidable task. Since the hardware implementation of communication system is highly dependent to various constraints, design exploration is an enabler to obtain optimum design [34]. For example, several performance indexes such as throughput, hardware complexity, and performance can be considered as the inputs for assessing design exploration, with the limitation constraints are resource area and acceptable development time (include verification time). Recently, a hardware-assisted platform for design exploration has been considered to address these issues [35]. [36]

Formally, for given k design metrics (parameters) P_k, the cost function of each design configuration, C, can be formulated by

$$C = \sum_{k=1}^{K} \alpha_k P_k \tag{4.8}$$

where α_k is weighting factor for k^{th} metric (priority of design target). The lower α_k represent

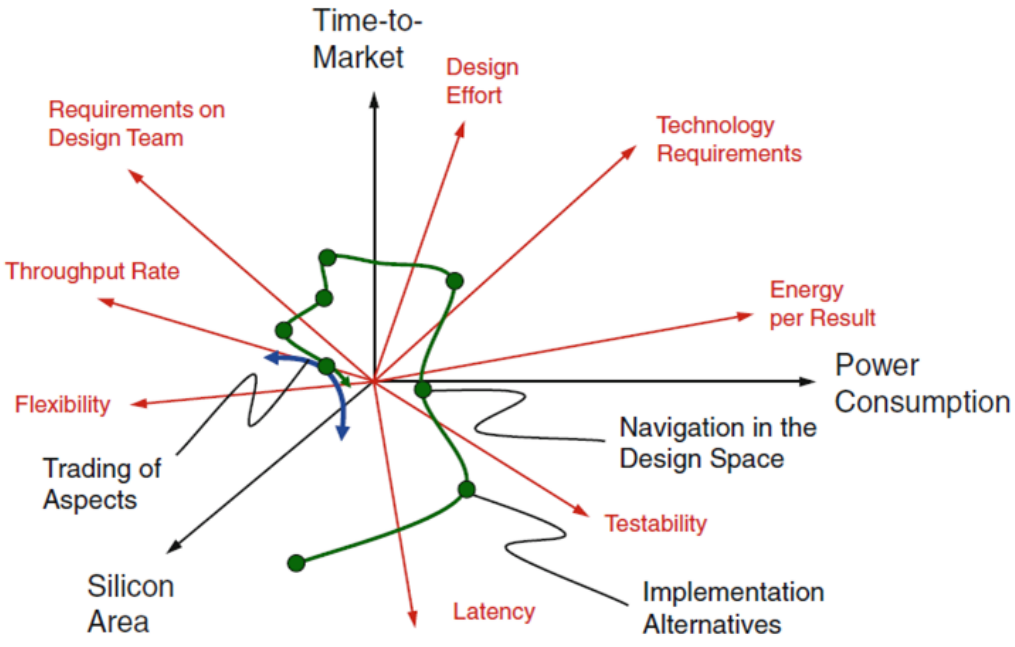

Figure 4.2: Multi-metrics Design Exploration

the higher priority of design metric.

The objective of design exploration is to minimize the cost function, C, subject to the available design constraint, P_k. The design metric is limited by the bound (capacity), denoted with b_k. Hence, the design exploration problem can be written as follow.

$$\begin{aligned} &\underset{P}{\text{minimize}} \quad C(P) \\ &\text{subject to} \quad P_k \leq b_i,\ k = 1, \ldots, K. \end{aligned}$$

Additionally, to evaluate the design exploration the designer should perform verification in different abstraction layers, which are: algorithm evaluation in system level, RTL simulation and hardware verification. In conventional approach, verification of those layers of abstraction are carried out independently. As a result, development process takes longer time. Moreover, since the verification process is carried out independently among design layers, the performance results from each layer cannot guarantee the consistency of performance in the point of view of system level simulation. Hence, the efficient verification time

is critical when involving design exploration.

To clarify the efficiency of design exploration task, we can evaluate the design and verification cost. Let the total of design versions to be evaluated are D . The cost of development, C is characterized by total design time T_{design} and verification time T_{verif}.

Since we employ model-based design, it is not difficult task to generate different version of design. Therefore, we can eliminate the variation of design times for several design versions, particularly if no major differences. Later, for this work we may use development cost as $C = kT_{verif}$, with k is constant factor. In the conventional way, for evaluating D different designs, we have to perform D times verification, which results development cost of design exploration as $C_{DSE} = kDT_{verif}$.

With the proposed verification framework, we can deploy several design version into hardware target and verify these designs concurrently. In the best case, D can be implemented in hardware target. When the each design version has same I/O structure, we only need to multiplex the I/O data for each design. This task can be addressed by the proposed flexible interface. Hence, the cost of the verification task can be significantly reduced only $C_{DSE_prop} = kT_{verif}$. This results a gain of design productivity :

$$G_{DP} = \frac{1}{\frac{C_{DSE_{prop}}}{C_{DSE}}} = D \tag{4.9}$$

This assumption is valid only for complex circuit, when the computation time of hardware processing is much longer than the software processing for data transfer.

In this thesis, we consider design exploration which the the objectives are : (1) finds out the optimum hardware usage, (2) takes efficient verification time, and (3) satisfies the system-level performance error rate. The design exploration task can be categorized into 2 main tasks.

1. *Algorithm co-exploration* : The typical design exploration in wireless communication system is algorithm co-exploration. Since the implementation of signal processing task varies in different algorithms, many existing algorithms can be selected as an optimum solution, regarding to design constraint. For example, in MLD MIMO Decoder, to implement distance calculation, the Euclidean metric is the most optimum

solution. However, in limited logic resource, the use of multiplier can significantly takes hardware resource. As an alternative, the approximation approach of distance calculation, such as Manhattan distance can be employed. In practical approach, we may employ simplified distance calculation by using a Manhattan distance, as given by:

$$\hat{x_M} = \arg\min \sum_{i=1}^{\Omega} |Re[D]| + |Im[D]| \tag{4.10}$$

Comparing Eq. 4.10 and Eq. 4.7, it is clearly shown that the Manhattan metric no longer uses multiplier for calculating metric value. The distance metrics is calculated as addition of real part and imaginary part of distance value, instead of using of square-value of distance. Hence, when a number of distance calculation unit are large (e.g in parallel processing MLD MIMO decoder) it can reduce significantly area resource as well as reduce critical path delay as compared to Euclidean metric. The evaluation of this exploration will be discussed in later section.

2. *Bit Length Decision* : As the data quantization on wireless signal processing will significantly affect overall system performance, the decision of bit width should be carried out carefully, in order to obtain acceptable performance loss. Hence, the evaluation of selected bit width must be performed in each stage design to ensure that performance of fixed point hardware design satisfy required performance. To provide a fast exploration of bit width decision, our proposed HW/SW co-verification can be utilized for faster bit-width decision by performing simulation various version of design. Furthermore, the simulation results are analyzed to determine the most optimum bit width regarding to system performance and hardware cost (logic area usage).

4.4 HW/SW Architecture for MIMO Decoder Implementation

In this section, we will describe HW/SW design for co-evaluation MLD-MIMO Decoder. This includes description of HW architecture design, FPGA implementation, HW/SW Hardware-in-the loop verification system.

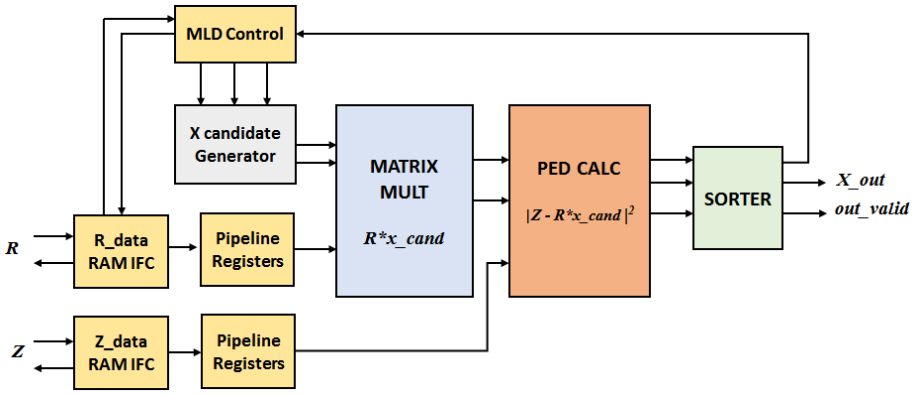

Figure 4.3: MLD MIMO Decoder architecture

4.4.1 MIMO MLD Architecture

The architecture of designed MLD decoder is derived based on MLD signal processing in Eq. 4.6. The overall block diagram of MIMO Decoder is shown in Fig. 4.3. While the main objective of this work is to provide verification framework, however, we also consider the low complexity design of Full MLD MIMO decoder. The low complexity design implementation is important in order to obtain efficient design, for both area usage and timing processing. The smaller area resource can give more opportunity to employed many design core in hardware target, while the efficient timing processing (i.e. faster maximum clock freq.) offers the faster run-time of simulation. Particularly, in an intensive DSP computation, such as Full MLD processing, the bottleneck of simulation processing is related to hardware processing. Therefore, implementation of efficient architecture implementation is critical, instead using direct implementation of calculation.

The MLD MIMO Decoder block mainly consists of:

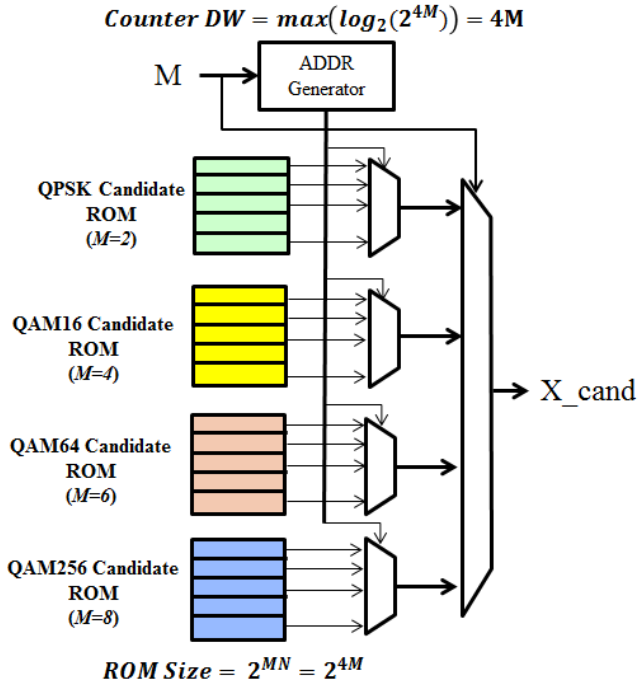

Figure 4.4: Conventional architecture of ROM candidate generator

1. **Candidate Generator**: This provides all possible transmitted signals. The number of reference transmitted signals are determined by the employed modulation scheme and the maximum number of transmit stream. Conventionally, the generation of candidate transmit signal is implemented by ROM. However, the direct implementation of unoptimized ROM structure will have huge cost of area resource. The ROM size will increase double exponentially as the number modulation scheme is higher. For example, in implementation of 16-QAM needs ROM structure with total size of 2^{4*4}, while the 64-QAM requires 2^{6*4}. The implementation of higher depth-ROM will results complex 2^{4M}-to-1 multiplexer and address decoding. Furthermore, this complex structure limits maximum processing speed.

   To reduce complexity of ROM structure, we can exploit the characteristic of value of constellation points regardless in-phase (I) or quadrature (Q) axis. Since the constellation points are constructed from unique values for each level point (L), we may only use the unique value and modify the structure of address decoding to implement the

ROM structure. The total number of L in each modulation scheme is $L = sqrt(2^M)$, where M represent modulation order. Furthermore, since we employ upto 4 streams, we also can separate the ROM upto 8 individual ROMs. Hence, the implementation only requires 8 ROMs with the each ROM consists of L words. The address for each ROM is selected from main address counter which the selector size for each counter is only $log_2 L$.

To clarify this idea, we present the modified structure of ROM for 16-QAM case as shown in Fig. 4.5.

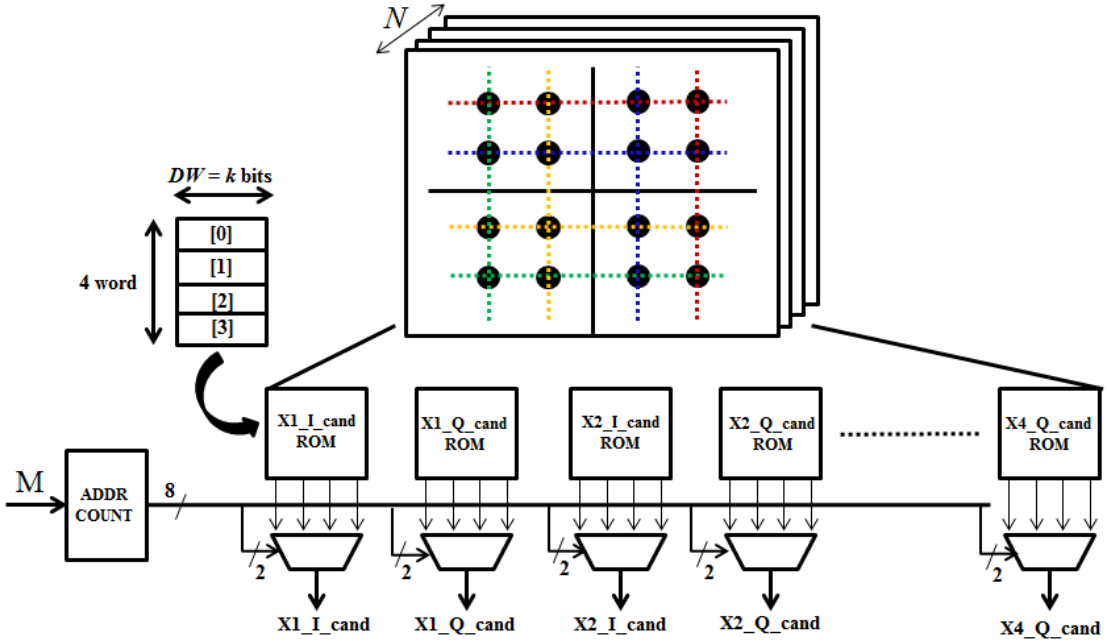

Figure 4.5: architecture of 16-QAM ROM candidate generator

In summary, the modified architecture will reduce the ROM size from 2^{4M} into $2^{M/2}$ and multiplexer complexity from 2^{4M}-to-1 multiplexer into $M/2$-to-1 multiplexer

2. **MATRIX MULT**: performs matrix multiplication between received signals and candidate of transmitted signal. In order to reduce multiplication number, we use QR decomposition pre-processing into channel matrix, **H**. With this pre-processing, the channel matrix form is modified into unitary matrix and provides 10 non zeros value.

Hence the total complex multiplications and additions are reduced from 16 to 10 units and from 12 to 6 units, respectively . The matrix multiplier block results 4 PED values which later will be used for distance calculation. The final structure of matrix multiplication is depicted in Fig. 4.6.

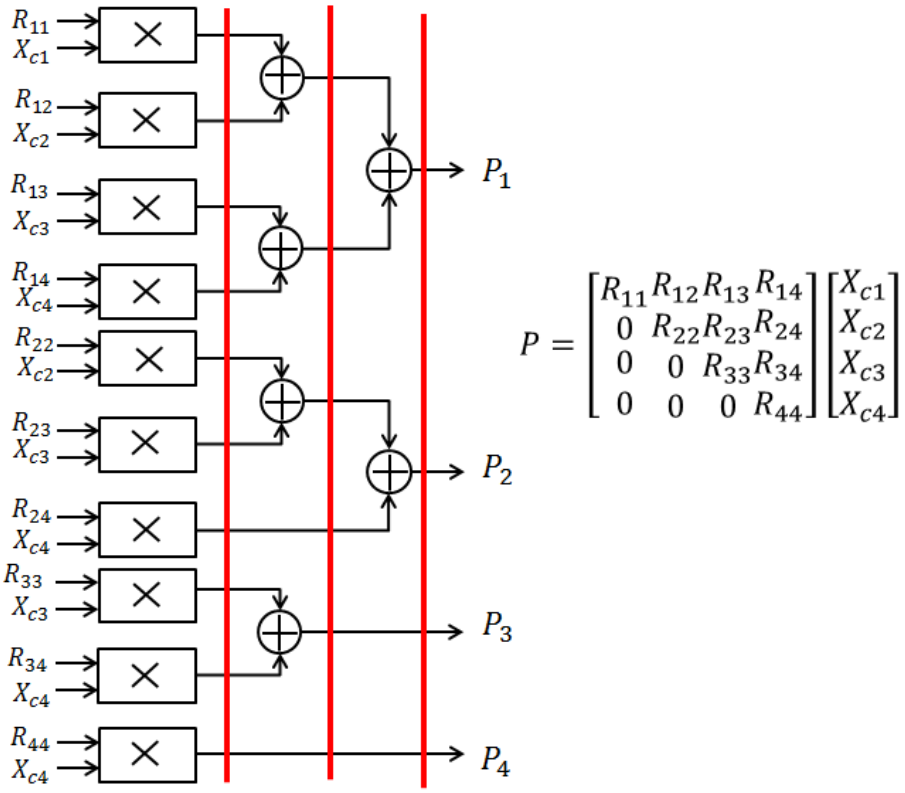

Figure 4.6: architecture of Matrix Multiplier Block

3. **PED CALC**: performs distance calculation, recursively along all possible number of candidate. The distance calculation is calculated as total distance from all PED value. The calculation of PED metric could be performed using Euclidean method or Manhattan method, as provided in 4.7 and 4.10. In Euclidean method, to obtain one PED value it is required 2 multipliers and one adder. On the other hand, Manhattan method does not require multiplication blocks. It only employ one adder to obtain PED value. The structure of distance calculation is shown in Fig. 4.7.

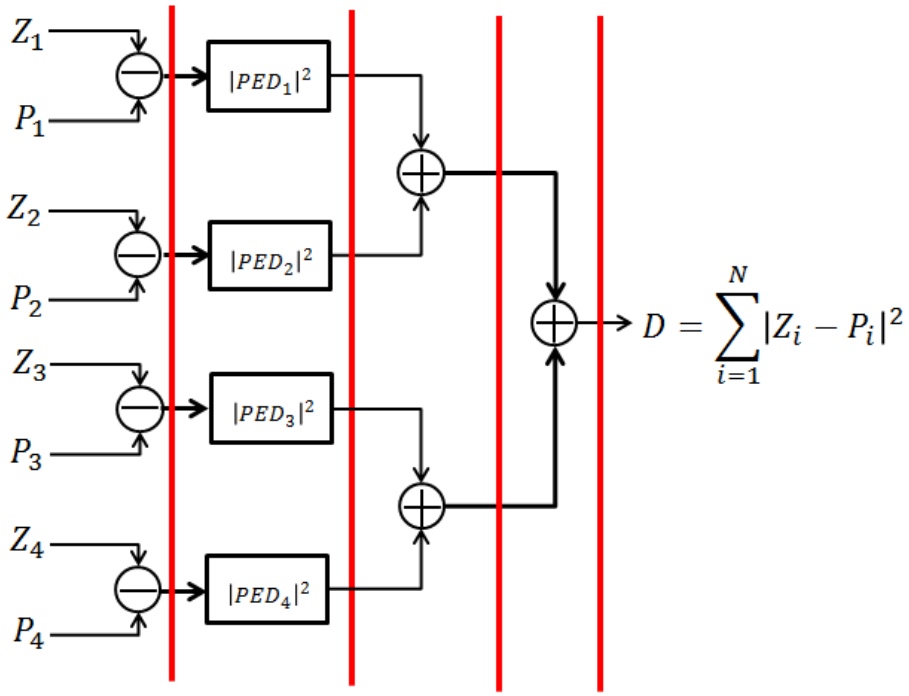

Figure 4.7: architecture of Distance Calculator Block

4. **SORTER**: determines the minimum error distance and selects the estimated transmitted signal. Since the calculation of distance is carried out sequentially, the sorter is implemented with simple sorting method that only employs one comparator and register for buffering the minimum value.

5. **MLD control**: provides control signal and parameter in appropriate timing. The baseline design with non-pipeline control, the distance data will be provided in every 8 clock cycles. Since the calculation of distance and updating minimum value can only provide one data in each cycle calculation, it introduces some idle clock cycles. This results a low-throughput MLD processing. Therefore, the control block is employed in order to manages the process of MLD calculation can be performed with zero-stall. Following are the timing chart of baseline MLD processing and non-stall MLD processing.

   It can be shown that the total processing timing can be reduced from $8K$ into $K + T_{latency}$. The K value represent the number of distances should be calculated for determining

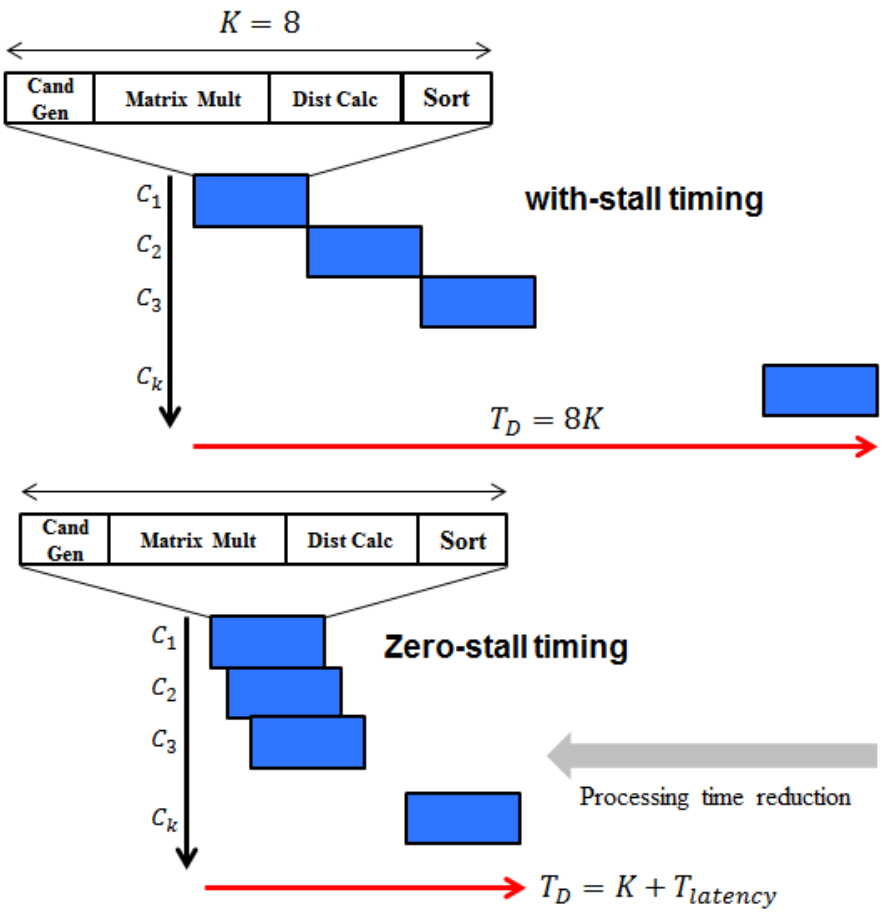

Figure 4.8: Timing Diagram MLD calculation

estimated transmit signal. For higher order modulation, which K is very high, the reduction is very significant.

6. Some **Pipeline registers**. These pipeline registers is inserted to break up complex processing within several stages. Hence, the critical path delay can be reduced and the achievable maximum clock speed is increased. By operating the circuit in higher speed, the verification task can be performed in faster time.

Furthermore, if the processing time are required to be improved the parallel processing core could be employed. However, the verification efficiency trade-off should be considered. This performance metric will be discussed in later section of this chapter.

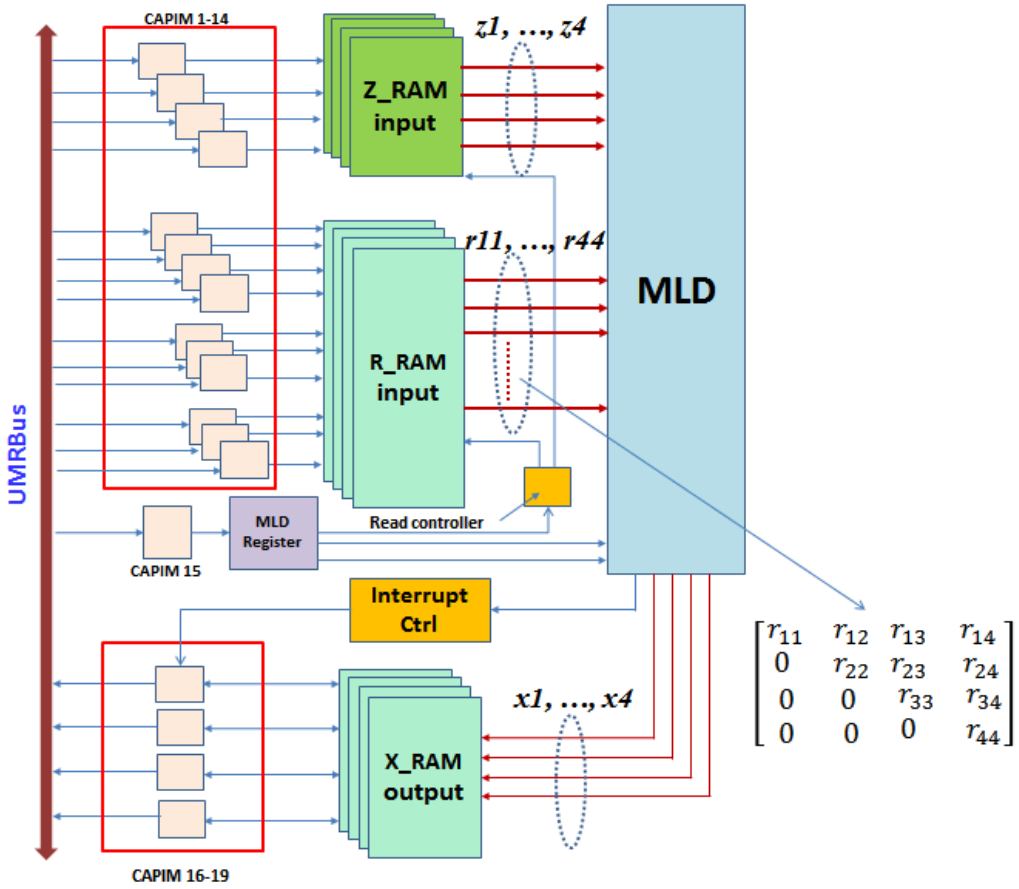

Figure 4.9: FPGA Architecture for MLD MIMO Decoder Implementation

4.4.2 Architecture for FPGA Implementation of MIMO MLD

When the DUT circuit is completely designed, the following task is FPGA implementation. The FPGA implementation we need to integrate the processing core (e.g DUT) and the hardware interface. The hardware interface consist of several CAPIMs and memories as well as the control circuit for address decoding within bus interface. The number of required CAPIM unit depend on the structure of input/output deployed system.

In order to implement MLD MIMO Decoder into template FPGA architecture, first we have to consider the structure of input-output of DUT. From 4.6, it is clear that MLD design has 14 inputs, which are R data matrices and Z data matrices. Additionally, we also will reserve one CAPIM to handle data input for system parameter and control, such as

modulation type, number of stream and start/enable signal.

According to the requirements of MLD MIMO decoder timing, the input of MLD MIMO decoder for each decoding process should be available at the same clock cycle. For simplification, we store each element of matrices input into different memory. Hence, we map each input data R and Z to one index CAPIM. In the same approach, we can assign each data output to one CAPIM unit. In case of MLD MIMO decoder, we need 14 CAPIM units for input buffer, 4 CAPIM units for output and 1 CAPIM unit for register control. In this FPGA implementation, the buffer size is determined according to the following reasons: (1) maximize parallel I/O and (2) support burst size for one packet frame data (e.g Low order modulation require larger amount of data). The buffer size for each I/O is considered as 32x14 bits RAM. The overall FPGA architecture for MLD implementation is shown in Fig. 4.9.

Moreover, the building of CAPIM structure is easily modified and configured for other DUT. Hence, the effort of configuring the interface design between FPGA and host PC for other applications is relative small.

4.4.3 Building Hardware-in-the Loop Co-verification System

After the hardware design of FPGA system has been implemented, we can employ this DUT and also the SW design into template architecture, building a complete HIL co-verification system. The DUT design is developed based on derived MIMO MLD decoder algorithm, as presented in previous section. On the other hand, the SW design is developed according to task partitioning process.

From the wireless block diagram, in Fig.4.1, we further could employ task partitioning and also task mapping for MLD MIMO decoder co-evaluation, as depicted in Fig. 4.10.

The high-level simulation software (e.g MATLAB) implement the rest of wireless signal processing tasks, except for MIMO decoding processing. These include transmitter process, channel model, receiver front-end, MIMO pre-processing (e.g. channel estimator and QR decomposition), data demodulation dan FEC Decoding. Other software tasks, such as performance evaluation and data transfer transfer, are natively performed by MATLAB and API software, respectively. Furthermore, the whole software tasks are constructed into

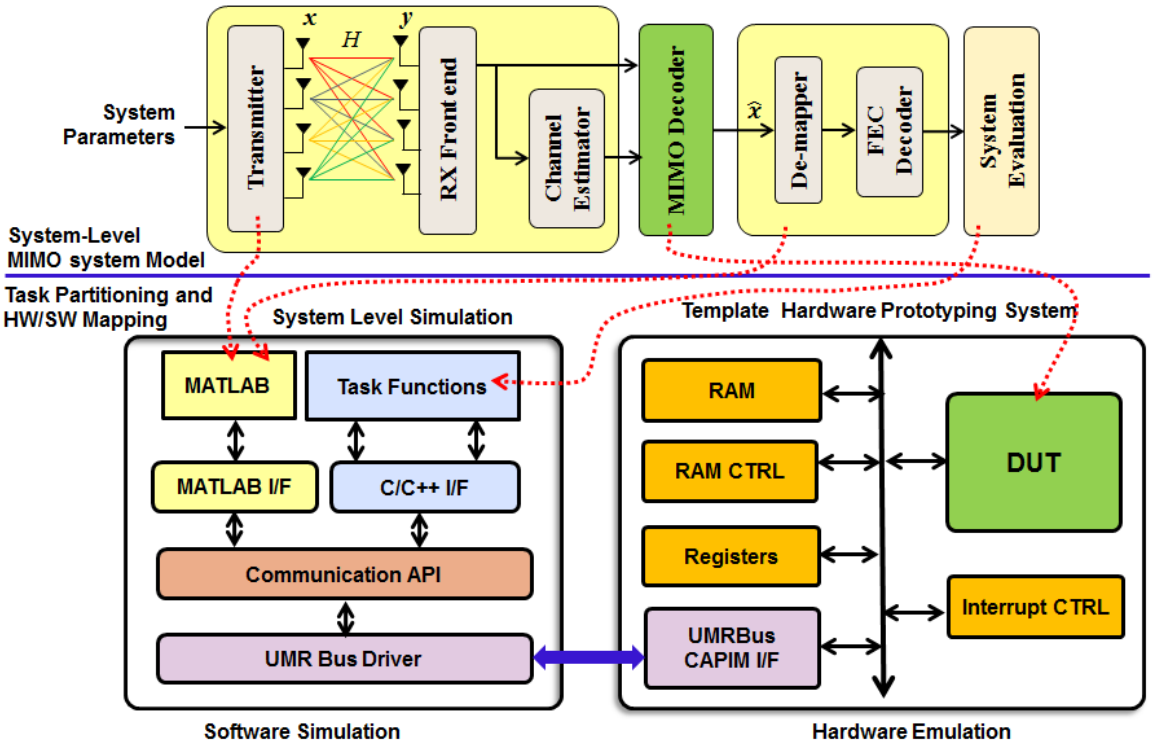

Figure 4.10: Task Partitioning and Mapping for Implementation of Co-evaluation MLD MIMO Decoder

main testbench software and will be executed in host PC for performing HIL co-evaluation. The complete structure of HIL system for MLD MIMO decoder co-verification is shown in Fig. 4.11.

4.5 Evaluation of Performance Metrics

In this section, we elaborate several performance metrics in order to evaluate the effectiveness of the proposed HW/SW co-verification, particularly in large scale circuits.

4.5.1 Estimation of Timing Processing

In order to obtain a valid verification time, estimation of processing time is the most important factor. Therefore, we make an accurate approach by abstracting each computation

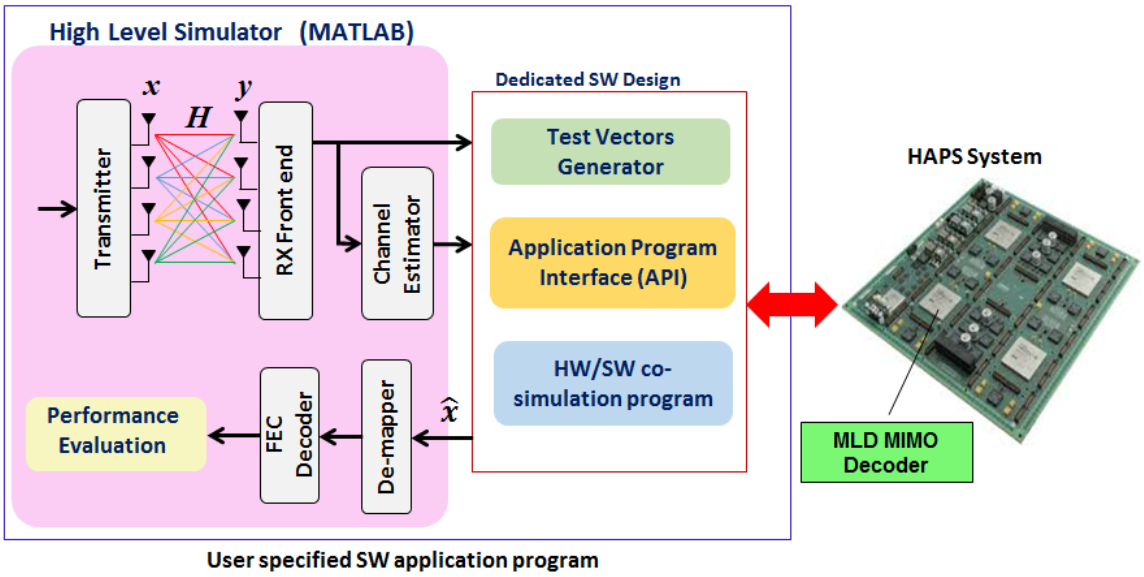

Figure 4.11: Hardware-in-the loop for MLD MIMO Decoder co-verification system

task, as shown in Fig. 4.12.

Note that, the verification time consists of two main parts, which are: software processing and hardware processing. Furthermore, execution time of software processing has several tasks: (1) initial setup, (2) data transfer from host PC to FPGA target in the beginning of simulation time, (3) interrupt handling, (4) data transfer from FPGA target to Host PC, and (5) post processing analysis for final performance evaluation after hardware process. The software execution time depends on the number of input and the length of burst data. In our case, software execution time only varies with the number of CAPIM and the length of simulated data. Variation of modulation type parameter does not affect software execution time.

On the other hand, the hardware processing mainly consists of several iterations of processing element (PE) execution, in which the number of iterations depend on employed modulation type. In our simulation case, to decode one subcarrier data, the PE will perform its process iteratively, as many as the number of candidates. Moreover, each processing of subcarrier also takes additional cycle for memory access.

The estimation of processing time for each modulation type is given in Table 4.5.1. This result is obtained by performing software profiling, which the software takes around

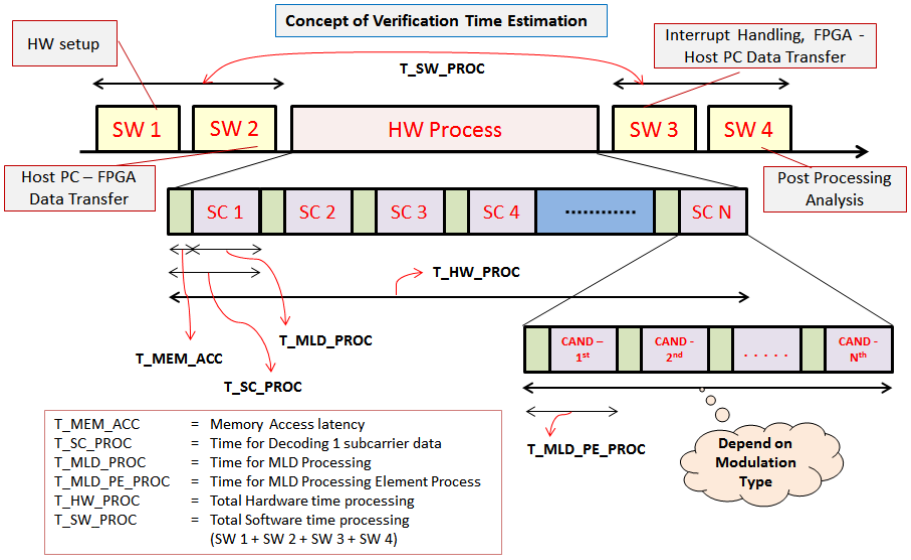

Figure 4.12: Illustration for Estimating Verification Run-time

0.02 second for every run time. For any target applications, we also can employ the same approach of estimation time to obtain estimation of execution time accurately.

Table 4.2: Estimation of Verification Time

Mod Type	SW proc (sec.)	HW proc (sec.)	Total proc(sec.)
1 (QPSK)	0.02	0.00261	0.02261
2 (QAM16)	0.02	0.65541	0.67541
3 (QAM64)	0.02	167.77221	167.79221
4 (QAM256)	0.02	42,949.67301	42,949.69

4.5.2 Verification Efficiency

To further speed up hardware processing, it is possible to change the design architecture. For example, utilization of parallel architecture of MLD can speed up hardware processing and finally results an improvement of system throughput. However, the cost of are resource and the achievable speed-up gain should be evaluated. In order to quantify this problem, we introduce verification efficiency metric. This metric assess the efficiency verification task when we employ parallel processing element. The verification efficiency, η, is determined

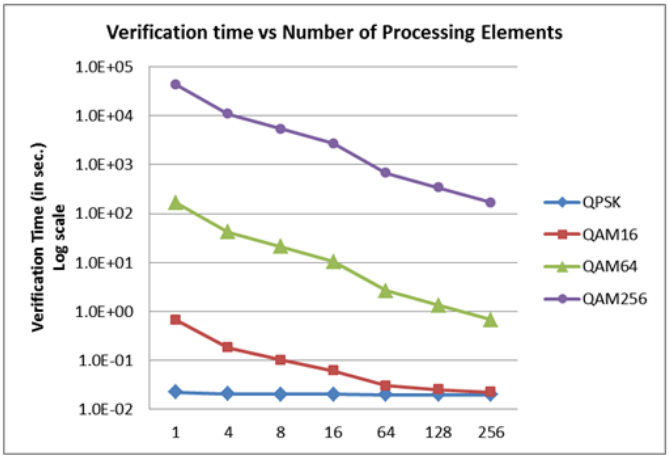

Figure 4.13: Verification time for various PE numbers

based on Eq. 4.11:

$$\eta = 1 - \frac{T_{paraPE}}{T_{singlePE}} \tag{4.11}$$

where, $T_{singlePE}$ is verification time carried out in single processing element architecture, and T_{paraPE} is verification time carried out in parallel architecture.

From this equation we further can characterize the effect of number processing core into simulation time. Let the simulation time for SW part takes T_S and HW processing time takes T_H for single unit of processing element. In the case of employing M processing element, the HW simulation time will be reduced into $\frac{T_H}{M}$. The SW simulation in parallel processing element takes the same time as in the single processing time. Hence, Eq. 4.11 can be rewritten in Eq. 4.12.

$$\eta = 1 - \frac{T_S + \frac{T_H}{M}}{T_S + T_H} \tag{4.12}$$

Moreover, if we assume the processing time of HW task is k times of the processing time of software, the verification efficiency, η, can be expressed as:

$$\eta = 1 - \frac{T_S + \frac{k.T_S}{M}}{T_S + k.T_S} \tag{4.13}$$

The latter, it can be derived into compact form as given by

$$\eta = \left(1 - \frac{1}{M}\right)\left(\frac{k}{1+k}\right) \tag{4.14}$$

The latest equation reveals that the efficiency is affected by two factors:

1. Performance gain of parallel processing (M)

   The efficiency gain of parallel processing is shown in the first term of equation. We can see that the efficiency gain will increased as the number of employed processing element is higher.

2. Hardware computation complexity (k)

   The verification efficiency is also mutually dependent on the complexity of hardware processing. It can be seen from the last term of equation 4.14, the verification will be efficient only when the complexity hardware processing time is much larger than software processing time (e.g k is larger).

In order to evaluate this metric, we calculate required verification time for different number of parallel processing elements. Fig. 4.13 shows the verification time for several number of PEs, from 4 PEs until 256 PEs. From evaluations, we found that increasing the number processing element, up to 256 units, will give another increasing of verification time about 100 times faster than using single PE. This result is obtained by employing the same simulation parameters in Table 4.1.

Therefore, the total speed up of verification time can reach up to 10^5 time, compare to pure software verification. However, increasing the number of processing element no longer give high benefit for speed up verification time. As shown in Fig. 4.14, the verification efficiency for the number of parallel PE area more than 64 units, the efficiency not increased significantly regarding to hardware cost. It remains constant for around 95% in 16QAM and approximately 99% in 64QAM and 256QAM. However, for low order modulation scheme, such as QPSK, the parallel architecture does not give significant gain for

verification efficiency. As shown in Fig. 4.14, the verification efficiency for QPSK mode, verification efficiency relatively constant around 12%. Hence, the optimum parallel PE is considered as 64 units and particularly effective in higher order modulation scheme.

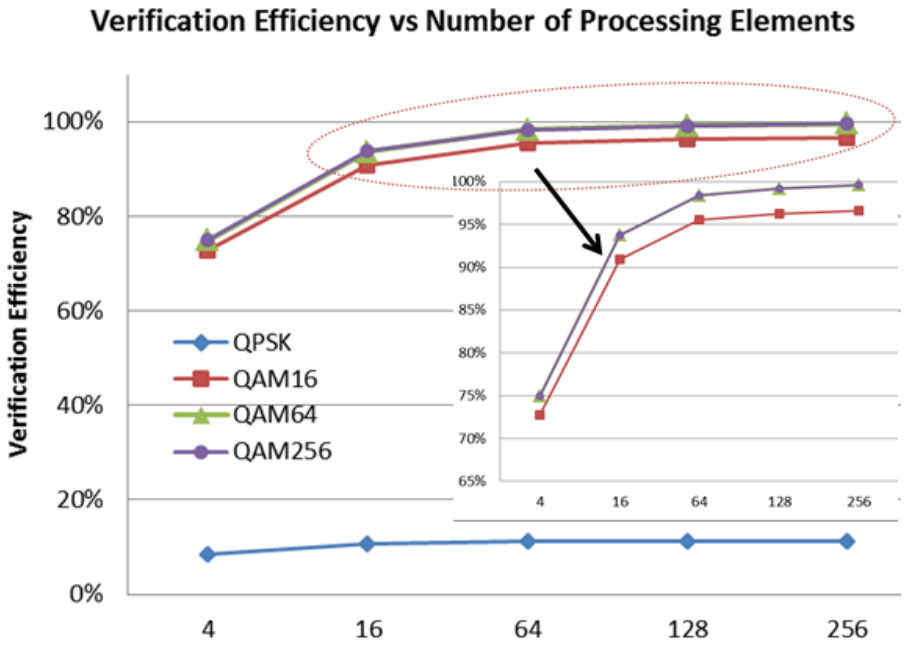

Figure 4.14: Verification efficiency for various PE numbers

4.5.3 Verification Runtime Speed-up

The simulation speed of the different environment verification has been measured in order to present benefit of proposed verification platform. The verification speed-up is defined as a comparison of verification time between proposed verification framework, T_{prop}, and other verification environments, which are MATLAB-Synphny HLS environment T_{SHLS} and Modelsim environment, T_{SW} . The MATLAB-Synphny HLS environment is selected to represent high-level abstraction framework, while the Modelsim represents full-Software verification framework. The verification speed up for two reference benchmarks are provided in Eq. 4.15.

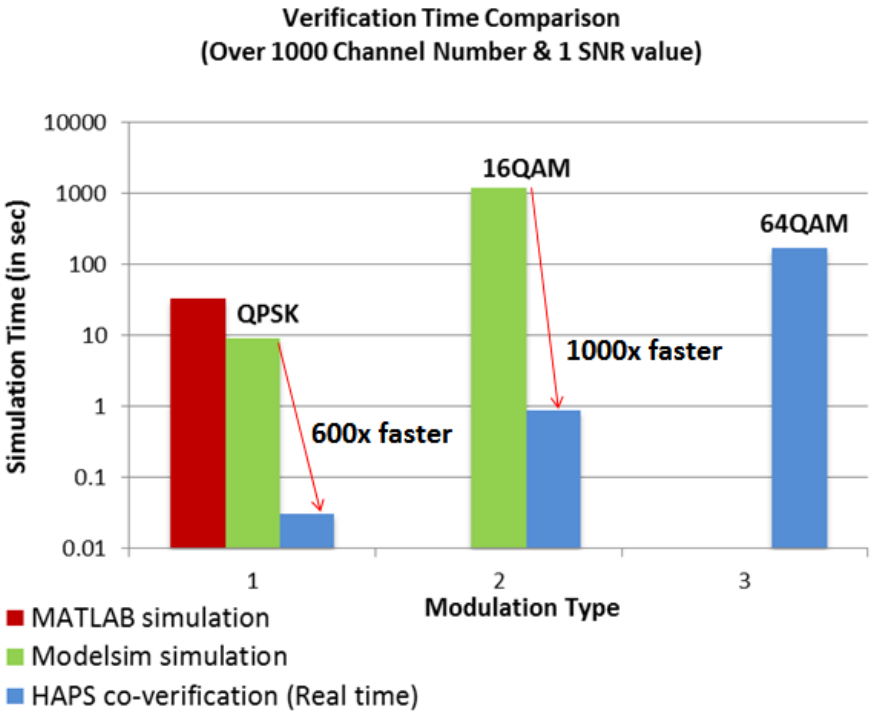

Figure 4.15: Verification Time Comparison

$$\alpha_1 = \frac{T_{prop}}{T_{SHLS}} \qquad \alpha_2 = \frac{T_{prop}}{T_{SW}} \tag{4.15}$$

where, α_1 is verification speed-up against MATLAB-Synphny HLS, while α_2 corresponds to verification speed-up against full-Software simulation (Modelsim),

Figure 4.15 shows the results of verification time for different modulation scheme. These results represent single processing element of MLD. It is also should be noted that the results of MATLAB-Synphny HLS can not be obtained for higher order modulation which are 16QAM and 64QAM. On the other hand, the modelsim simulation is not applicable for 64QAM modulation.

As shown in Fig. 4.15, the speed up of verification time is more than 3 times magnitude (1000 times) compare to software simulation, such as Modelsim and MATLAB. It also can be seen that the simulation time is close as predicted by pre-calculation.

Furthermore, from experimental simulation it is shown that the bottleneck of data transfer is no longer occurred. For instance, when evaluate transmitting large data (1000 Byte) using high order modulation, the software processing for data transfer between host PC and HAPS system is negligible and the limitation lied in hardware processing. Because the software processing part took a constant time, while the hardware processing part depend on employed modulation type.

4.6 Experimental Evaluations

4.6.1 FPGA Implementation Results

FPGA implementation is carried out on hardware target HAPS system with FPGA device is Xilinx-virtex6. Implementation results of MIMO MLD decoder for single PE and parallel PE (64 cores) are summarized in Table 4.6.1.

Table 4.3: FPGA Implementation Results of MIMO MLD Decoder

Resource Usage	**1 PE**	**64 PEs**
LUTs	18,024	140,851
Registers	7,989	32,827
RAM Blocks	576	576
Gate Count	218,139	1,497,447

It is found that the designed MIMO MLD Decoder can achieve clock frequency upto 180 MHz. Furthermore, it is also possible to implement a number of parallel processing elements or even several cores of different version of design in order to speed up hardware processing time since our hardware platform have big capacity of logic resources.

From synthesis result, the MIMO Decoder design occupies 18,024 look-up tables (LUTs), 7,989 registers, 576 blocks of RAM. On the other hand, 64-PE units of MIMO MLD decoder requires 140,851 LUTs, which is almost 8 times compared to single PE. The parallel design also requires 35,827 registers, which is equivalent to 4.5 times of single PE usage. Additionally, the RAM design occupies the same blocks with single PE implementation, which is 576 resource blocks. Since each block RAM is constructed from 36x1 Kbits RAM, the total employed memory bits is equivalent to 30,736 Kbits.

It is should be noticed that implementing parallel processing element does not change the design of bus interface and memory structure. Hence, we can utilize the rest of the LUT resource, as well as the DSP block for implementing the parallel processing element or other cores. The

In order to provide a clear comparison corresponding to area complexity of LSI design, we have converted the resource usage into gate count. Hence, we can use this value for comparison between design target. Essentially, the task of gate translation between FPGA and ASIC is difficult and cannot give exact gate count due to several reasons, such as: coding style, synthesis optimization, Placement and routing algorithm, library quality, etc. However, to give estimation of gate count, we use approximation value which each LUT is constructed from 9 gates and one register employs 7 gates [37]. Additionally, this assumption also considering the fact that the number of gate counts in one LUT vary depend on the number of inputs used.

According to our estimation constraint, the total gate count of MLD design in single PE and 64 PEs are around 218 K gates and 1.5 M gates, respectively. This result reflect that the MLD MIMO decoder design has high complexity in term of gate count (e.g. more than one million gate)

4.6.2 Example Evaluations

1. **Bit Error Rate Performance**

   The performance of proposed MIMO Decoder design has been evaluated regarding to various channel characteristics and modulation types, as shown in Fig. 4.16 and Fig. 4.17 respectively.

   Figure 4.16 shows BER performance for various modulation types, from QPSK until 64QAM. The results also include MATLAB simulation results that provides performance benchmark. The MATLAB simulation is carried out in floating point simulation. On the other hand, the HW/SW simulation employ MLD decoder as DUT. In this simulation, the MLD MIMO Decoder applies Euclidean metric for distance calculation and uses datapath length of 16 bits.

   From simulation result, we can see that the performance of hardware implementation

of MIMO decoder is close to MATLAB simulation for QPSK and 16 QAM schemes. The very small performance lose is only affected by bit quantization. However, for high order modulation such as 64QAM, it is impractical to obtain the result since its excessive run-time. The simulation results reveal that the task of algorithm transformation, from floating-point format into hardware fixed-point format is sufficient.

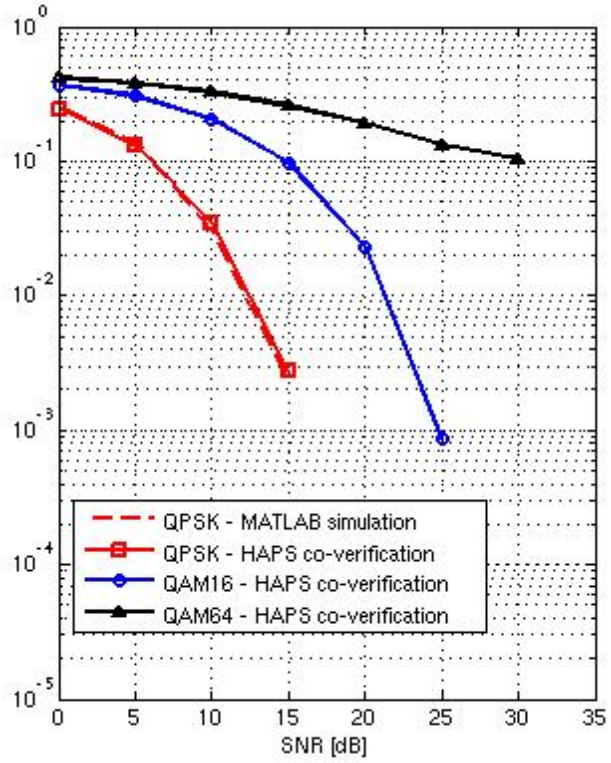

Figure 4.16: BER performance on various modulation types

Furthermore, we also evaluate the performance of DUT for for various channel type, as shown in Figure 4.17. In this simulation, the DUT employs QPSK modulation scheme. The performance of MIMO decoder is evaluated under three types channel, which are : Identity channel, TGac AWGN, and TGac Channel D.

From Fig. 4.16 and Fig. 4.17, we can prove the capability of our verification platform for comprehensive evaluation of wireless communication system, that covers various scenarios. This result also confirm that the verification platform can perform

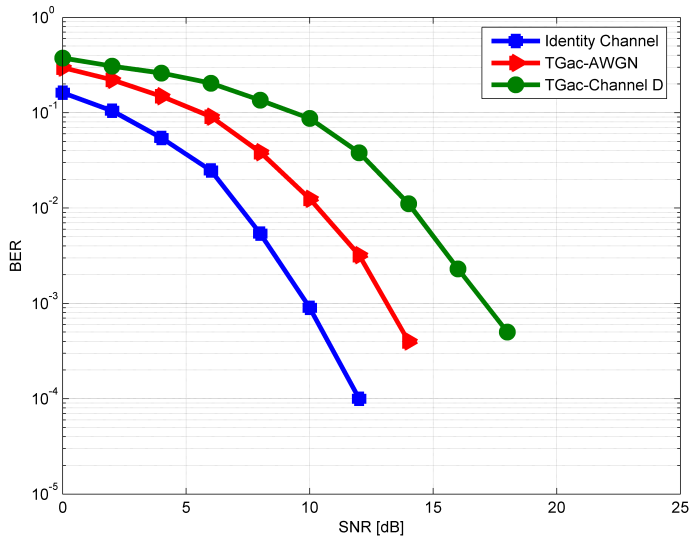

Figure 4.17: BER performance on various channel types (QPSK Mode)

evaluation in different abstraction layers, which are system level design (MATLAB) and physical implementation (FPGA).

2. Algorithm Co-exploration

   Fig. 4.18 shows comparison of performance of MIMO MLD Decoder for Euclidean distance and Manhattan calculation method.

   The performance results in low order modulation such as QPSK is almost same for various SNR. On the other hand, in higher order modulation such as 16-QAM, the performance of Manhattan distance calculation is slightly below from Euclidean distance calculation, particularly in higher SNR (above 20 dB). These results give insight to the designer to evaluate if the selected algorithm is sufficient or not, according to error rate performance requirement. However, selected algorithm should be decided as early as possible during design development. Thus, the fast design exploration as well as fast verification is a key element in order to validate that algorithm approximation gives acceptable performance loss. By evaluating two design versions concurrently, the verification time ca be reduced become half of conventional verification scheme. This result corresponds to design productivity gain of 2.

   As summary, we can show these results into DSE chart, as shown in Fig. 4.19. The

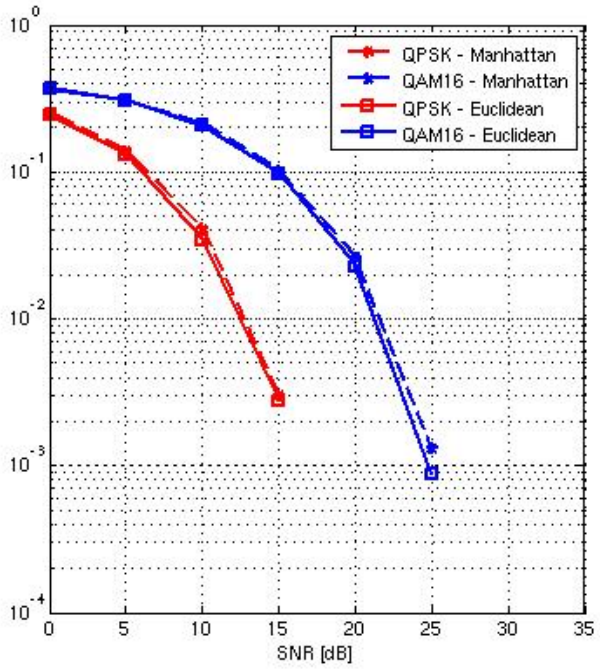

Figure 4.18: BER performance for different types of distance calculation

more optimum of design alternative is selected with the direction of navigation curve (dash line).

3. Optimum Hardware Complexity

   In Table 3, we provide logic resource utilization from the implementation of parallel PE of MLD decoder. This table confirms that increasing the bit length results exponential growth of logic resource usage in term of gate count. For example, by increasing bit-length two times (e.g from 16-bit into 32 bits) results in almost 4 times increasing of gate count, from 1.5 M gates to 5.2 M gates. Hence, we further can make trade-off this area resource results with achievable error rate performance.

   The evaluation results of BER performance for various bit lengths are shown in Fig. 4.20.

   From the evaluation results, we can see that the error performance of MIMO MLD

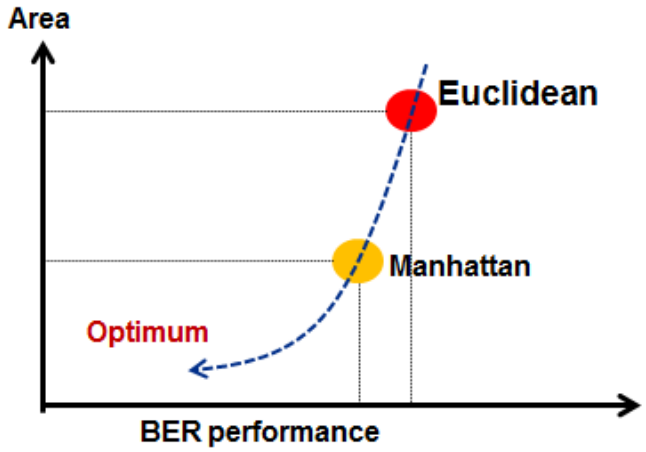

Figure 4.19: DSE chart for algorithm exploration

Table 4.4: Hardware resource usage for different bitwidth

Bit length	LUTs usage (%)	Register usage (%)	Gate Count
8	26,140 (5.5%)	8,768 (0.9%)	296,636
11	54,760 (11.5%)	14,553 (1.5%)	594,711
16	143,753 (30.3%)	28,496 (3.0%)	1,493,249
24	277,321 (58.5%)	33,959 (3.6%)	2,733,602
32	535,798 (112%)	60,252 (6.0%)	5,243,946

Decoder will improve as the bit length is increased, as its nature. The 8-bit implementation gives the worst performance and it is not acceptable. On the other hand, the 11-bit implementation is can enhance an error rate performance, particularly in lower SNR. However, when the channel condition is better, error rate performance is degraded and will not satisfy the error rate target. The implementation of 16-bit, 24-bit and 32-bit can approach the upper error rate bound, which is floating point software simulation. However, the performance improvement when the bit length is higher than 16 bits is relative small from floating point simulation of MATLAB and no longer significant. Hence, trading-off the performance and hardware usage, the efficient hardware for final implementation is considered as 16-bit.

To summarize the performance results and hardware implementation results, we can represent these data into DSE chart, as hown in Fig. 4.21. From the DSE chart, we

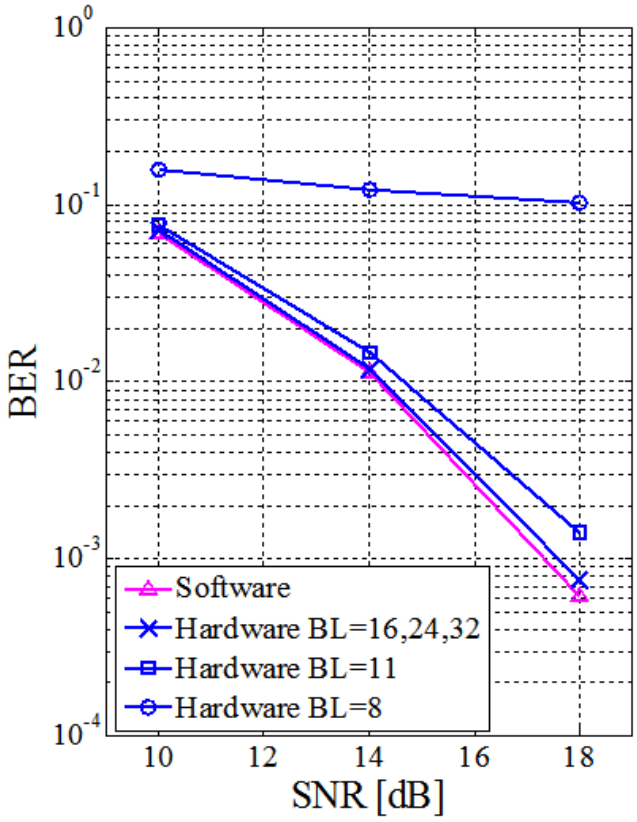

Figure 4.20: BER for various datapath bit length

have two design options that satisfy the allowed DSE space, which are 11-bits design and 16-bits design. Since we consider the error performance is the higher priority, the optimum of 16-bits design option is selected.

4.7 Summary

In this chapter, we have presented an application example of the proposed co-verification methodology in order to prove its effectiveness on speed-up verification time. This evaluation also involves the evaluation of design exploration. The MLD MIMO Decoder for high throughput tireless system is selected, as a representative of the complex circuits. The implemented MLD MIMO decoder employs approximately 1.5 M gates count. The performance metric in term of verification efficiency, achievable run-time verification, and design

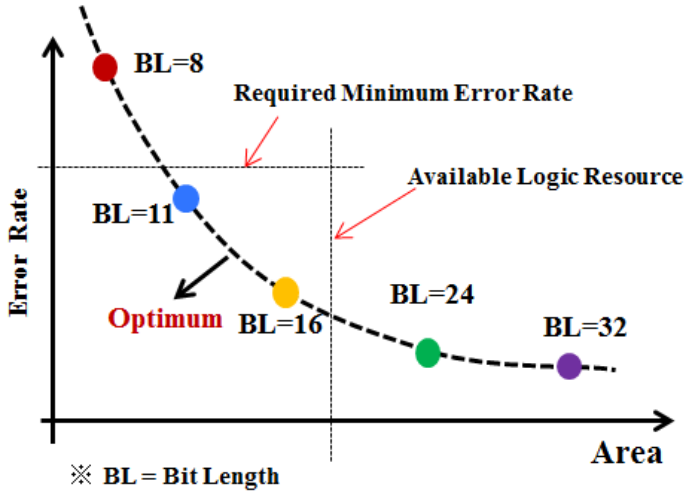

Figure 4.21: DSE chart for MLD MIMO decoder implementation

exploration evaluations are presented. The experimental evaluation using hardware-in-the loop verification scheme is also presented. Experimental results with MLD-MIMO Decoder case show that the verification speed up can achieve upto 1,000 times in single PE and upto 100,000 times in 64-parallel PE, with the verification efficiency almost 99 %. By using this scheme, the proposed frame work is able to perform comprehensive evaluation for various test scenarios in the point of view of system-level simulation, as shown by BER performance of whole wireless system.

Chapter 5

Unified System Level Simulator for Very High Throughput Wireless Systems

5.1 System Level Simulator in Wireless Communication System

Over one decade, wireless technology has evolved rapidly with the adoption several important technologies by commercial standards (e.g. WLAN, WiMAX, LTE, DVB/DVB-T, etc). Unfortunately, the introduction of new standard always results in a significant increasing on system complexity and signaling. Therefore, it increases the difficulty for system evaluation. To deal with this issue, the availability of a platform for system-level evaluation in wireless systems development is very important. This tool can help to respond the fast evolution of adopted technologies and allow to anticipate the up-coming standard, such as 5G networks. The main requirements for realizing such tool are: fast run-time evaluation, high-flexibility platform (easy for modification or extension), large-coverage of use cases of evaluations (functional, timing, multiple levels of design abstraction).

Several platforms for system-level evaluation have been presented, including software-based simulation [42]-[41], FPGA-accelerated[43]-[44], and SDR-based platform[45]. In [39], MATLAB-based system-level simulator for LTE system is proposed while ref. [42]

presented a simulation and emulation for MIMO Wireless Baseband Transceiver. The full-software system-level simulation framework such as ref. [39] has main advantage on its flexibility of design modification and extension. However, the achievable run-time in this framework is limited and timing-related evaluation could not be performed. On the other hand, the simulation framework in ref. [42] offers real-time evaluation on full-hardware platform. However, it requires huge cost and efforts for development and limited flexibility for system modification. Considering the trade-off between run-time, flexibility, and test coverage, we extend the proposed unified verification framework for application of system-level simulation.

The proposed system-level simulation has to cover several tasks in the development and verification process, particularly in early stage of system development, which are:

1. **Evaluates algorithms**

   This task is carried out in order to investigate feasible algorithm for design target such as a wireless transceiver implementation. Practically, each wireless communication standard only defines clearly the processing or computation of transmitter part. Therefore, the hardware implementation of transmitter can be realized more rapidly. However, in the receiver side, the implementation cannot be done straightforward as in transmitter. The selected algorithms are left to the designers. When designing components of receiver system, there are no trivial solutions, particularly when the performance is highly depend on impairments or interferences. Many feasible solutions can be selected among available candidates. Therefore, several trade-off between error-performance and design cost should be investigated carefully.

2. **Provides reproducibility and controllable data generation**

   Reproducibility becomes even more important when the system complexity is growth, as in the case of evaluation of wireless system [39]. Although there are available resources of database in some applications, however in the field of wireless signal processing, it is not practical for providing such simulation resources. By providing reproducible data, we can perform thorough checking, particularly in large test scenarios and corner-test evaluation.

3. **Provides reference data**

Although there are many references and various literatures for wireless simulation case, however, there are some ambiguities. It is not always clear which part of specifications were actually implemented and which part were omitted for the sake of simplicity. Hence, it is difficult to confirm that the evaluated design satisfies the standard-compliant or not.

4. **Evaluates cross-layer system performance**
   When dealing a comprehensive evaluation of wireless system, it is mandatory to consider the interactions data flow and signaling between layers (cross-layer). The performance of one layer will directly affect the performance of its boundary layer. Therefore, the development PHY layer (e.g transceiver system) has to consider the system protocol that is defined in the upper layer (e.g MAC layer). In order to address this issue, the proposed co-verification system should cover the evaluation for system function for both MAC and PHY related functions.

In this work, the evaluation scope will cover two layers of 802.11 reference model, which are PHY layer and MAC layer. The evaluations include link level error performance, achievable timing processing, low-complexity and transceiver algorithm for PHY layer evaluation. On the other hand, for MAC layer evaluation, it include evaluations of signaling methods, packet data transfer, payload length allocation, and protocol timing requirements. The simulation framework is shown in Fig. 5.1.

5.1.1 Cross-layer communication protocol

In order to realize efficient SoC implementation and to provide comprehensive system-level simulation, it is important to understand deeply how the physical layer operates and how it interfaces with the MAC layer.

Data Flow Operation : The MAC layer communicates with the upper half of PHY layer, which is Physical Layer Convergence Protocol (PLCP) sublayer. It communicates through a service access point (SAP) commands. When the MAC layer instructs the PHY layer, the PLCP prepares MAC protocol data units (MPDUs) for transmission by taking the frame from the MAC sublayer and creating PLCP Protocol Data Unit (PPDU). The PPDU data is constructed from a preamble, PHY header, and PLCP Service Data Unit

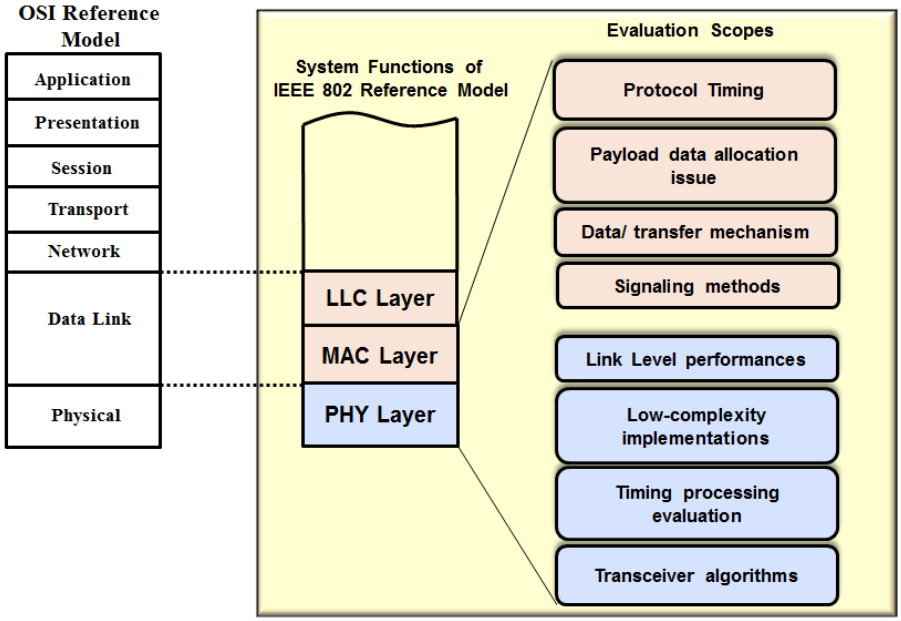

Figure 5.1: System Level Framework of Simulation for WLAN system

(PSDU) from MAC layer. Under the direction of the PLCP, the bottom half of PHY layer, which is Physical Medium Dependent (PMD) sublayer provides transmission and reception of PPDU data units via the wireless medium. The diagram in Fig. 5.2 shows the data information moving between the MAC layer and PHY layer.

PHY Operation and Frame Structure : For transmission process, the PHY operation is dictated by service parameter from MAC layer, which is called as TXVECTOR and RXVECTOR. These parameters setting include several necessary configuration of PHY to generate, transmit, and receive data frame. The TXVECTOR provides the PHY with the transmission parameters, while the RXVECTOR assists the MAC with received parameters of PHY. The TXVECTOR and RXVECTOR are constructed into compact form (word-size boundary) to adapt with memory transfer size. In our simulation case, the TXVECTOR and RXVECTOR consists of 5 data words.

Transmission and reception process of WLAN system are carried out in packet based (frame). The 802.11ac standard which is referred as Very High Throughput (VHT) system. The VHT PHY frame consists of a legacy preamble, a VHT preamble and the data payload, as depicted in Fig. 5.3. The legacy preamble consists of a Legacy Short Training

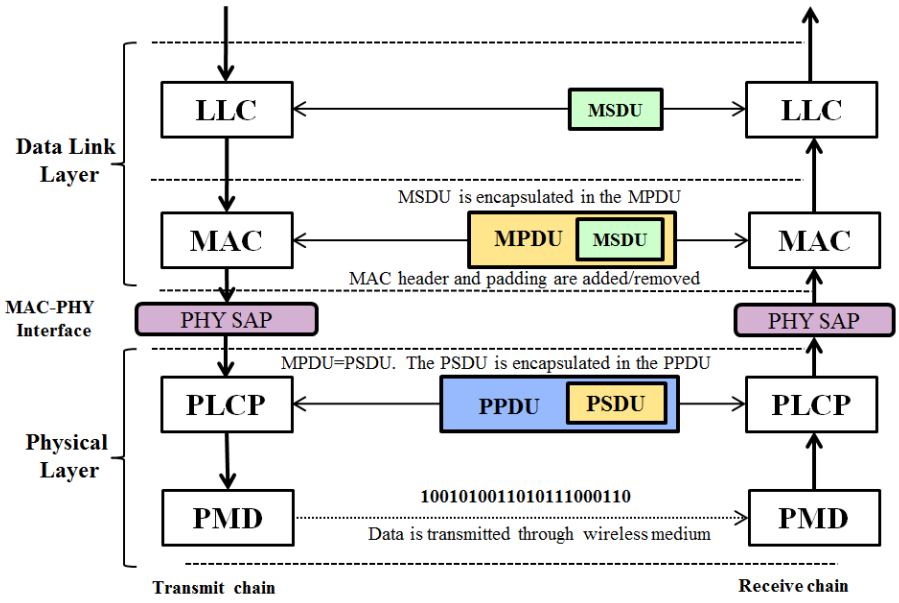

Figure 5.2: MAC-PHY data flow

Field (L-STF), a Legacy Long Training Field (L-LTF), and a Legacy Signal (L-SIG) field. These fields are the same as the ones in 11a and 11n (legacy and mixed formats) preambles. They allow all 802.11 devices to perform initial timing and frequency synchronization to the received data frame, and to avoid interference of other stations. Then follows the VHT Signal-A (VHT-SIG-A) field, VHT Short Training Field (VHT-STF), VHT Long Training Field (VHT-LTF), VHT Signal-B (VHT-SIG-B) field and finally the DATA symbols. VHT-SIG-A field carries MCS informations and VHT-specific parameters. VHT-STF and VHT-LTF include the information for channel estimation and MIMO configurations. VHT-SIGB field contains informations of length parameters and MU-MIMO support. These informations are utilized by PHY to determine how to decode and how much data to decode. The detail explanation of frame structure and PHY operation can be found in [4].

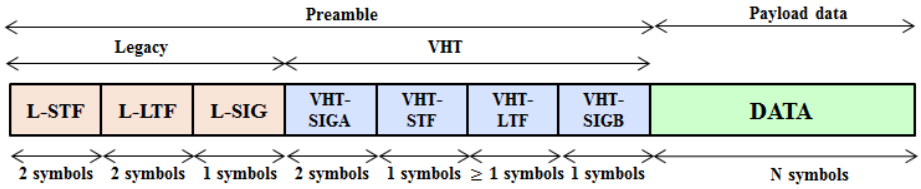

Figure 5.3: VHT Frame structure

5.2 PHY Transceiver of Very High Throughput Wireless Communication System

5.2.1 Multi User Wireless System

One of technology advancement in the latest WLAN standard, IEEE802.11 ac is the adoption of Multi User (MU) communication for downlink path. A complete review of MU transmission scheme in 802.11ac WLAN system is presented in [47]. By adopting MU scheme, one access point (AP) is able to use multiple streams simultaneously for transmitting data to several users. This scheme is employed to further increase throughput performance and to satisfy user's experience demands. The MU MIMO wireless system incorporates one AP terminal and several k users (STA), as depicted in Fig. 5.4.

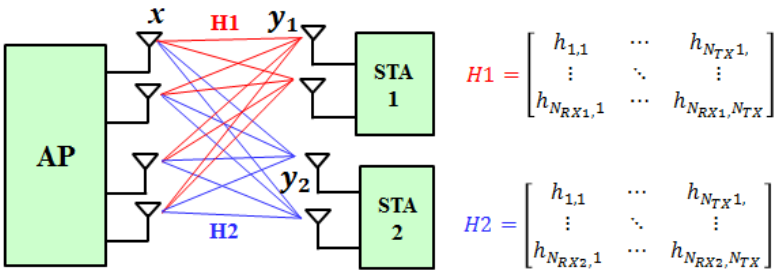

Figure 5.4: Multi User MIMO Wireless System

The AP with pre-coding scheme transmits $\boldsymbol{x}$ signal through N_{TX} transmit antenna over wireless channel H_k to the users. The received signals y_k in N_{RX} antenna of k-th STA is

represented by

$$y_k = H_k x_k + n_k \tag{5.1}$$

where n_k is the noise vector of user k.

The received multi-stream data in each user, $\hat{x_k}$, is detected using linear minimum mean square error (MMSE) MIMO detector, denoted by

$$\hat{x_k} = {H_k}^H [\sigma^2 \mathbf{I} + H_k {H_k}^H]^{-1} y_k \tag{5.2}$$

Here, σ^2 is the noise power, $(.)^H$ stands for Hermitian transpose and $\mathbf{I}$ for the identity matrix.

In this work, we employ 1 AP terminal and 2 STA users with channel link of each user is represented by H_1 and H_2. The transceiver specification adopts IEEE802.11ac standard [4], as summarized in Table 5.1.

Table 5.1: System Parameter

Parameters	**Value**
Packet Mode	VHT Mode
System Configuration	4x[2 2]
Number of Spatial streams	2
System bandwidth	80 MHz
FFT point	256
Guard Interval	800 ns
Modulation Coding Scheme	0-7 (BPSK - 64QAM)
FEC	Convolutional Coding
MIMO Detection	MMSE

5.2.2 Transceiver Structure

The transmitter follows the data flow as shown in Fig. 5.5. During data transmission, the allocated payload data for each user is scrambled, encoded and punctured independently. The scrambler avoids long sequences of bits with the same value, and adds desired properties to the transmitted data stream. The data field is scrambled using specified generator

polynomial. The DATA field in the 11ac system should be encoded using one of two types of FEC encoders: BCC and LDPC. However, in this thesis we employ only BCC encoder in PHY transceiver. Puncturing is a process to remove redundant encoded bits with a specified pattern. This reduces the number of transmitted bit and increases the coding rate. In the receiver side, the dummy zeros bits are inserted to replace the removed bits.

This punctured data, then is parsed into several streams in order to increase the transmission rate. The stream parser divides the puncture data into small blocks of bits, and then re-arranges into spatial streams, which represents the MIMO streams. The following process are interleaving and constellation mapping according to employed modulation scheme. Interleaver helps addressing the problem of burst error by shuffling the data bits in different code words. This creates an improved uniform distribution of the errors. In 802.11ac system, the interleaving is performed independently in each OFDM symbols by performing three steps of permutations. The equations for permutation are described in [4]. The interleaved serial data bits are arranged in groups of 1, 2, 4, 6, or 8 bits and mapped into complex numbers representing a BPSK, QPSK, 16-QAM, 64-QAM, or 256-QAM constellation points respectively. Each constellation point is mapped into one sub-carrier data.

In order to leverage spatial and frequency diversity, the data is mapped onto spatial transmit chain by spatial mapper block, and furthermore is interleaved across sub-carrier index. The spatial mapper block also perform beamforming process, where the transmit data is multiplied with weight matrix during pre-coding stage. Then, the pre-coded data, in form of OFDM symbol, in each transmit chain is modulated by IFFT and applied with guard interval insertion and windowing. Finally, the resulting signal is transmitted over wireless channel.

The corresponding signal processing block of receiver is depicted in Fig. 5.6. Most of signal processing block perform opposite process as in transmitter, except in receiver front-end part such as synchronization, CFO correction, and AGC for adjusting power of received signals. The first parts of receiver are mainly to detect the burst, synchronize, estimate the channel, and equalize the symbols, while the remaining of blocks reverse the processes of the transmitter.

In order to achieve the target data rate in Giga bit per second (Gbps) order, the transceiver

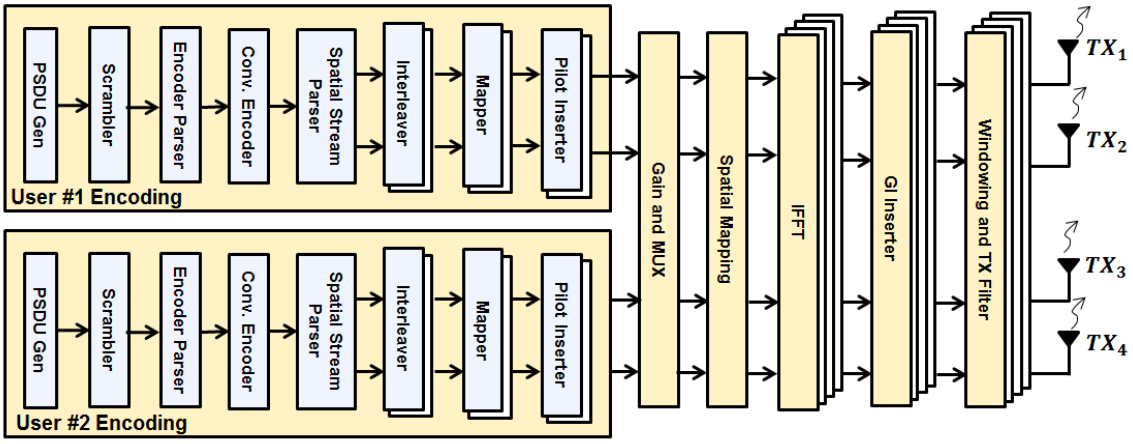

Figure 5.5: Transmitter Block Diagram

design consider some strategies. These include: (1) parallel architecture oriented, (2) exploiting the pipeline depth, and (3) employing multi clock domain. These approaches are applied to time-critical processing, for example spatial stream de-parser (SSDP) and viterbi decoder. In the SSDP block, highly parallel architecture is employed in order to provide high throughput data for viterbi (parallel-data). Since we keep the usage of radix-4 viterbi decoder, the design of viterbi module exploits the pipeline level and thus can employ higher clock frequency. Hence, we can decode received data with the improvement of 2 times of throughput. In the same time, this design strategy also maintains the usage of hardware resource. As a result, the processing time requirement can be satisfied with efficient hardware implementation when the primary implementation constraint is limited available resource area.

5.2.3 System Level Issue

While, the final implementation of SoC has not carried out yet, however, in order to make seamless integration and final verification, we should take into account SoC-like structure for system-level evaluation.

As the transceiver was completely designed, it will be integrated with MAC layer block, as well as the application processor within SoC Architecture. Basically, the WLAN design

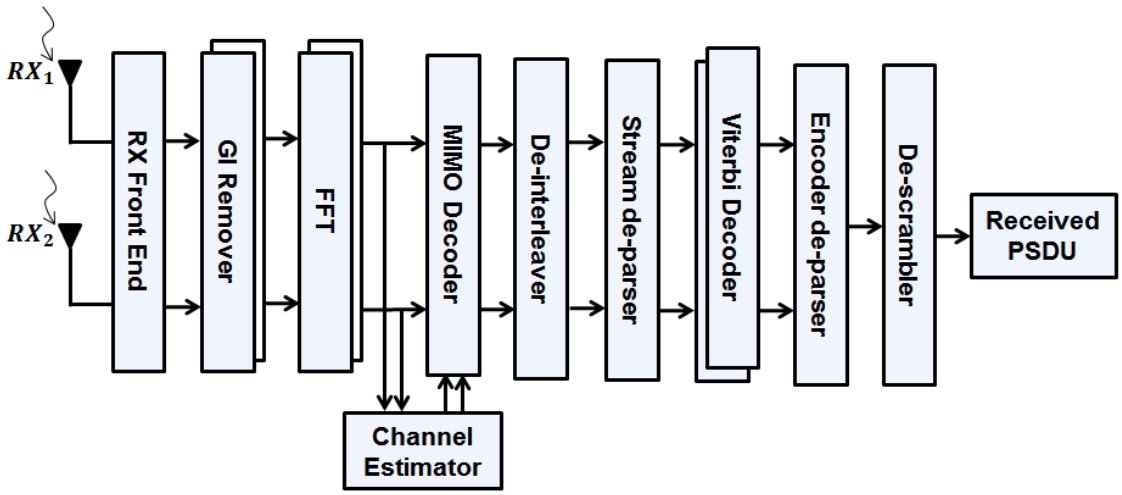

Figure 5.6: Receiver Block Diagram

(PHY and MAC) and application processor will be attached into specific bus (e.g AMBA bus). These components with several I/O peripherals and memory systems construct a complete SoC system. In the SoC point of view, the interactions between PHY, MAC, and application software in processor are carried out through memory system and interrupt service. Hence, we also need to provide an interface system that realizes data transfer and handles the interrupt. By using the provided template generic architecture, we can easily map the WLAN SoC structure into proposed HW/SW co-verification architecture. Figure 5.7 shows the minimal WLAN SoC design in the proposed unified system-level simulation.

In this work, we make task partitioning for the sake of simplicity and the limitation of hardware platform. First, we omit the MAC protocol function and only consider the SAP procedure of bottom layer of MAC. Hence, we only implement the MAC interface design. Additionally, the RF block is also eliminated in hardware implementation. However, the impact of RF processing could be model in MATLAB simulation. The task of processor in SoC is mapped into host PC by employing MATLAB code and C program to provide data from upper layer and to model cross-layer interaction (SAP procedure). The Host PC also handles the interrupt function from interrupt control circuit in hardware design.

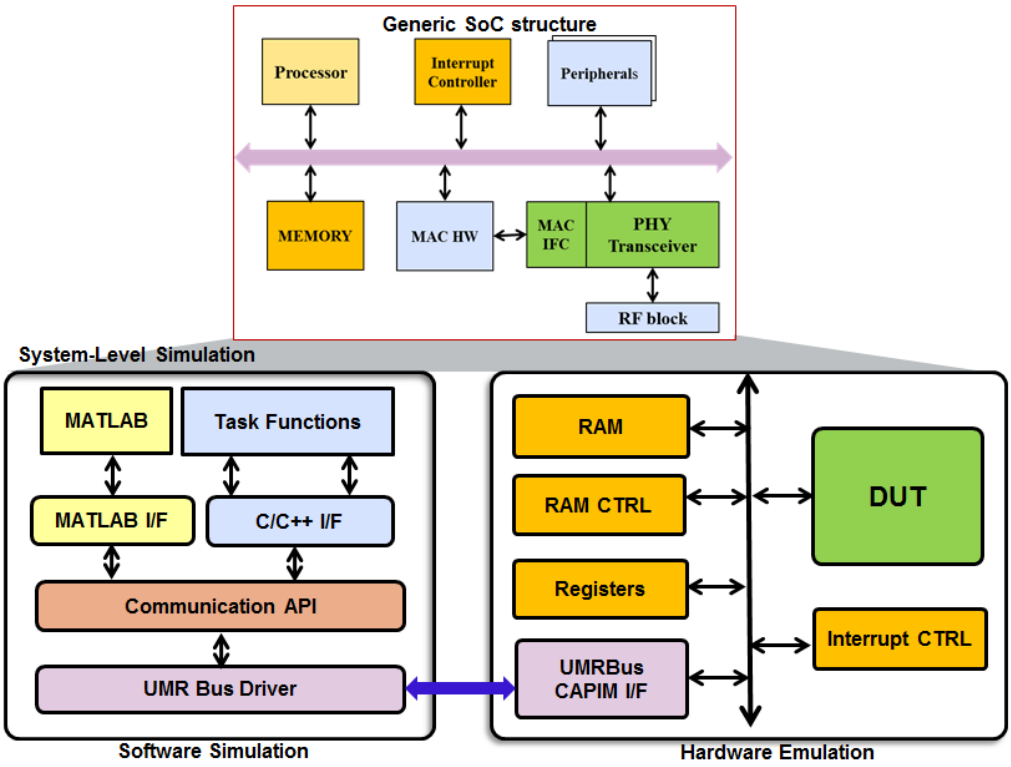

Figure 5.7: Minimal WLAN SoC structure in Unified System Level Simulator

5.3 FPGA Architecture and Implementation Results

For the implementation of HW/SW co-evaluation, the fully-compliant 802.11ac standard PHY transceiver is implemented in hardware platform. The design of MAC interface is also implemented in order to make realistic interface with MAC HW layer. The MAC interface process is directed by the control start from host procesor. This interface delivers the TXVECTOR content in TX Header Memory. Additionall, the FPGA implementation also includes the hardware interface that receives data from end-point bus system, I/O management block, as well as memory blocks. According to the verification requirements and to specified interconnection, we employ 13 units of CAPIMs. Each CAPIM is connected to every RAM of PHY transceiver. The address assignment of a pair CAPIM and RAM is provided in Table 5.2 and its corresponding FPGA architecture is shown in Fig. 5.8.

The hardware design is implemented using device target Xilinx FPGA Virtex6 XCVLX760

Table 5.2: CAPIM Address Assignment

CAPIM Address	**Usage**
1	PHY Configuration Registers
2	Transmit Header Memory
3	TX MSDU Memory for user 1
4	TX MSDU Memory for user 2
5-8	TX ADC Memory
9	Receive Header Memory
10	RX MSDU Memory
11-12	RX DAC Memory
13	General Purpose Debugging Memory
14	TX Beamforming Memory

within HAPS hardware platform. In this implementation, the PHY system employs multiple clocks processing, in order to achieve high throughput while maintain efficient implementation. In particular, the transceiver system is operated with the clock frequency 60 MHz, except for viterbi decoding and modules in front-end receiver, that use 120 MHz and 20 MHz clock frequency, respectively. However, in the final target implementation the designed PHY system will be operated with clock frequency of 240 MHz in order to achieve real-time system. The implemented design is feasible to achieve the target clock frequency of 240 MHz by considering the gap between FPGA and ASI implementation. As confirmed by study in [38], the standard-cell ASIC implementation is approximately faster between 3.4-4.6 times than FPGA design. Hence, the transceiver system could be directly implemented into ASIC design without major modification for architecture optimization.

The implementation results are provided in Table 5.3. It should be noted, the result in Table 5.3 is resource usage for each set of transceiver system. The transceiver system occupies 274,324 LUTs and 62,161 Registers. This results corresponding to gate count around 2.9 M approximately. Additionally, the total employed RAM block is equivalent to 19,656 Kbits RAM.

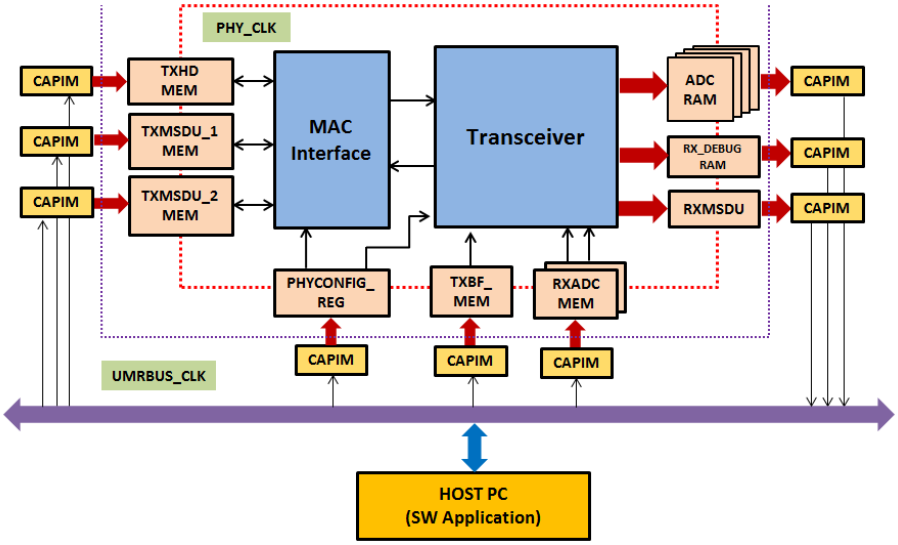

Figure 5.8: FPGA Architecture Implementation

Table 5.3: FPGA Logic Resource

Resource Block	**Usage (%)**
LUTs	274,324(57%)
Registers	62,161 (6%)
Block RAM	546 (Out of 720)
Gate Count	2,904,043

5.4 Unified HW/SW Simulator

The structure of HIL system for system-level simulator of MU-MIMO Wireless system is depicted in Fig, 5.9. In the previous chapter, we only employ the most critical block in receiver. On the other hand, in this chapter, we extend the hardware part into full system of transceiver. Then, the software task performs the data processing for channel coefficient generator, hardware impairment model, as well as performance evaluation. The MATLAB function also performs weighting matrix calculation for beamforming process.

Co-evaluation process is carried out in the following step:

1. The Host PC perform system simulation by executing MATLAB simulation of the transmission process according to given system parameter that selected by the user.

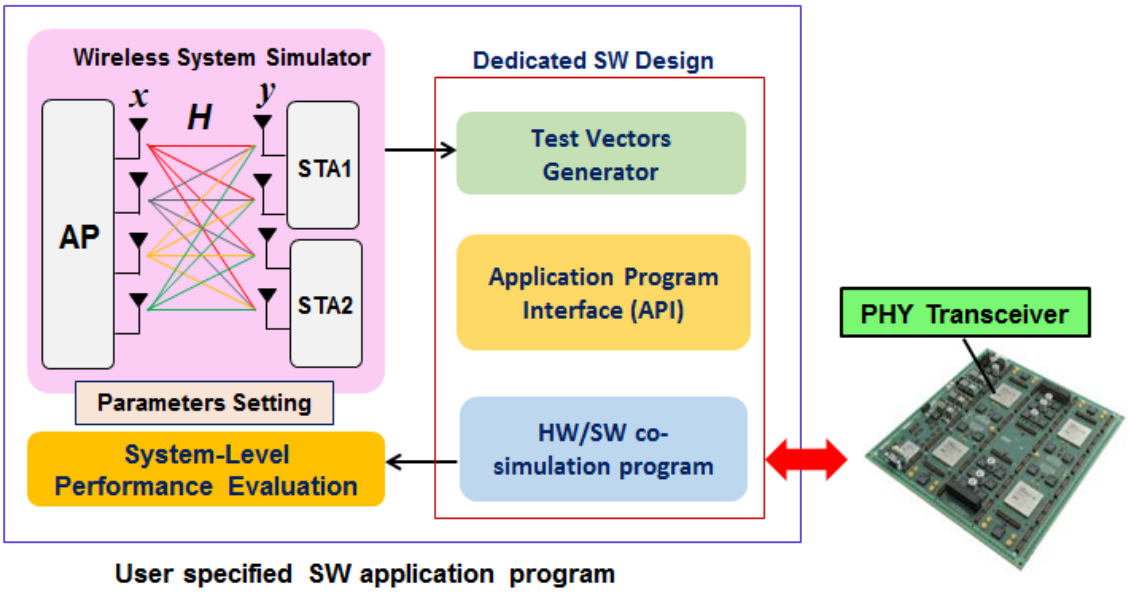

Figure 5.9: MIMO Multi-User Co-verification System

2. The transmitted signal that also affected by hardware impairments send to channel model to get the received signals for the receiver. In the same time, we also can evaluate the correctness of transmitted signal compare the floating point simulation. The evaluation of transmitter in term of MSE could be also performed.

3. This received data latter sent to the hardware platform through the API and software driver that is executed within the simulation testbench program. The testbench program will send the input vector for receiver block to the designed memory. The data transfer will be performed for all receiver input including the register control that provide configuration setting of receiver.

4. When the control start register of receiver is written the hardware will perform simulation, while the software hold its task and waiting the interrupt signal from hardware part.

5. As the interrupt signal is available and recognized by software, the software could access the hardware to read intended results, whether intermediate results of receiver in MIMO decoder output or final result of received data in form of MSDU packet. The

read data then collected by the API function to further return to MATLAB program for system evaluation. From the received simulation results, various performance evaluation could be carried out in software function as well as the behavior of upper layer.

5.5 System Level Performance Evaluation

In system-level evaluation, the performance evaluation is carried out according to the standard-compliant test. It includes (1) link-level performance such as BER and PER and (2) the achievable throughput performance of system-level. The system-level evaluation also assess system latency to investigate achievable inter-frame duration since the packet transmission in WLAN based network is strictly limited by defined SIFS duration.

The transmission scheme is carried out in packet transmission, where the data is sent in frame based transmission. System-Level evaluation is carried out according to the system parameters in Table 5.1. The simulation environments is designated by also taking care the hardware impairments. The summary of simulation conditions are presented in Table 5.4.

Table 5.4: Simulation Condition

Parameters	**Value**
Hardware Impairments	PA Nonlinearity, Phase Noise, CFO, ADC/DAC quantization
Channel Model	TGac D
Distance AP-STA	20 metes
Number of transmitted packet	1000 byte

5.5.1 Error Rate Performance

In this section, we provide simulation results in order to evaluate system-level performance of the employed PHY transceiver. We conduct the evaluation of end-to-end error performance of PHY transceiver system. The PER performance of VHT transmission scheme

mode is provided. These results will be used as a basis for further evaluation of system-level performance. Figure 5.10 shows PER performance for high order modulation, e.g 16-QAM and 64-QAM. From this result, we can conclude that the transmission will achieve

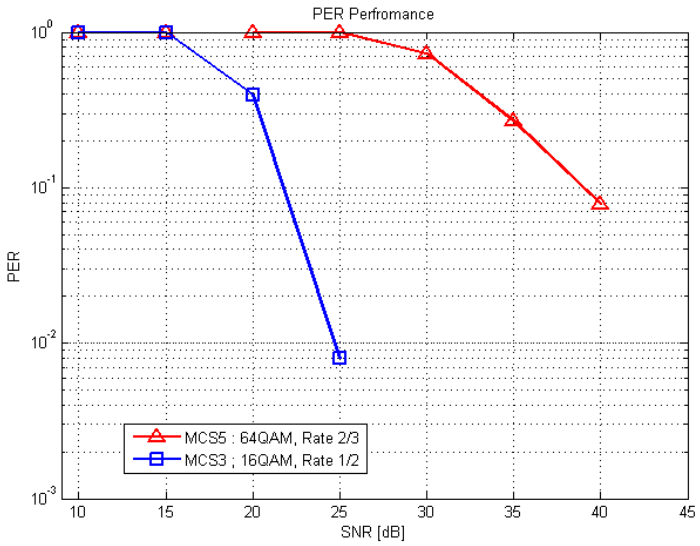

Figure 5.10: Error Rate Performance of transceiver

error-free in SNR condition above 25 dB and 40 dB in 16-QAM and 64-QAM, respectively. Furthermore, from the obtained PER results we may evaluate the throughput performance.

5.5.2 Latency Performance

The latency evaluation is also important in the development 802.11 based system, since the most difficult problem in implementing the 802.11 system is strict requirements of the Short Interframe Space (SIFS) which is defined as within 16 μs [46]. The SIFS performance will directly affect the protocol timing in MAC layer. The SIFS is the time counted from the reception of the last PPDU symbol to the transmission of the first symbol of the response PPDU, as showin in 5.11. Within the SIFS duration the PHY should complete its packet reception that also include the RF delay and PLCP delay (D), the time slot of data preparation for transmitting response (M), and turn-around time of Rx/TX switching. Since the response data (ACK frame) should be immediately performed after this SIFS duration,

achieving this timing constraint is a mandatory in order to satisfy the requirements of the upper layer protocol.

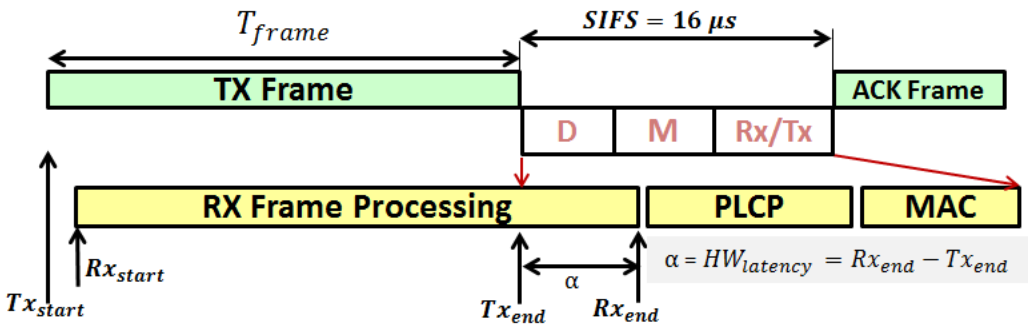

Figure 5.11: Transceiver processing latency illustration

From the observed simulation waveform, the introduced hardware latency take about 3,081 clock cycle, that corresponds to 12.84 μs. This amount reveals that the system has available time-slot for other processing (MAC processing, CPU cycles, etc.) and feasible to catch up the requirements of SIFS duration. Hence, we can conclude the designed transceiver satisfy the requirements of upper layer network (MAC layer).

5.5.3 Achievable Throughput Performance

The evaluation of throughput performance is carried out by also considering the interaction with upper layer, which is MAC layer. Additionally, we also include several hardware impairments as presented in Table 5.4. The throughput measurements is evaluated by transmitting numerous of random data packet over emulated channel. In the simulation, we employ 1,000 packets with each packet contains 1,000 bytes of MAC payload data (MSDU). The throughput, Γ, can be computed as follows.

$$\Gamma = \frac{L_{payload}}{T_{frame}}(1 - PER) \tag{5.3}$$

where $L_{payload}$ is length of MAC packet data payload in bytes, and the T_{frame} is the duration of transmitted frame that is constructed from transmission time from PHY header ($T_{PHYheader}$), SIFS duration (T_{SIFS}), and whole packet (T_{packet}) that follow Eq.

$$T_{frame} = T_{PHYheader} + T_{SIFS} + T_{packet} \tag{5.4}$$

The packet duration is determined by the employed transmission parameters, such as modulation scheme, transmission bandwidth, number of spatial stream, etc. The duration of packet is determined by the number of the transmitted OFDM symbol, while the OFDM symbol duration has been specified as 4 μs. Using the PHY header as 44 μs and SIFS durations 16 μs as well as the obtained PER value, the achievable throughput can be obtained as depicted in Fig. 5.12

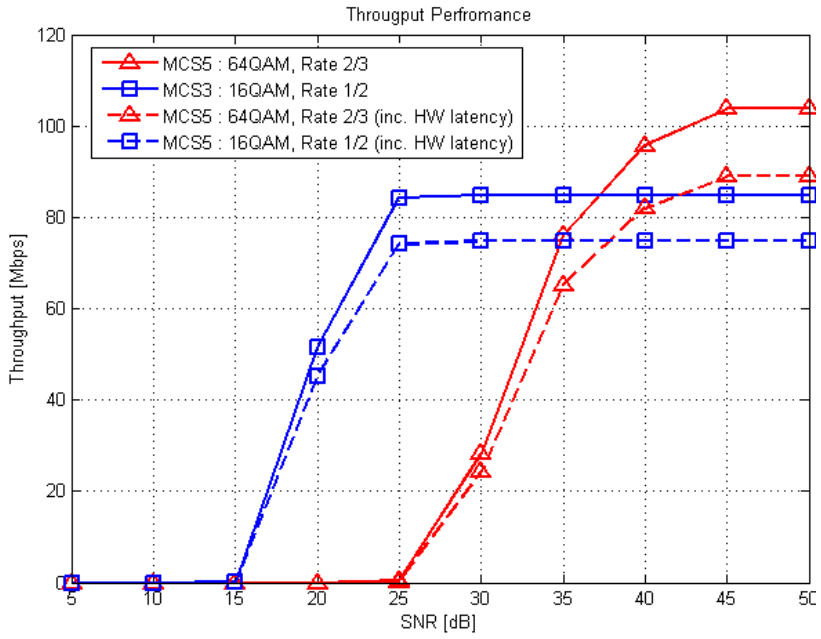

Figure 5.12: Achievable Throughput for Different MCS

In order to obtain realistic performance results ,we also evaluate the throughput from the real hardware design that includes timing processing of transceiver. Since the hardware processing results some latencies (e.g latency for data-buffering and pre-processing task), we add the correction factor, α, in calculating achievable throughput. The introduced hardware latency is calculated from the end of transmitted frame to the end of packet reception

process, as illustrated in Fig. 5.11. Hence, the corrected throughput performance could be calculated by the following Eq.

$$\Gamma_{HW} = \frac{L_{payload}}{T_{frame} + \alpha}(1 - PER) \tag{5.5}$$

Moreover, in order to gain more higher throughput, close to the PHY data rate, we can utilize a longer payload data, especially in error free channel condition. This scenario eliminates the overhead of PHY header and signaling field in transmitted packet frame. From simulation, the throughput will be improved when the length of payload data increases, as shown in Fig. 5.13. Thus, through this evaluation we can decide the optimum employed length of packet data.

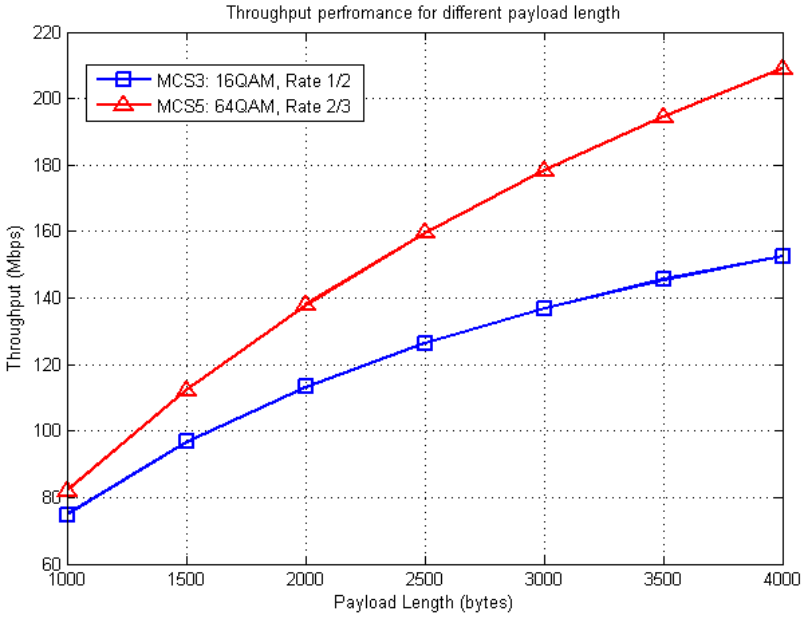

Figure 5.13: Achievable throughput for various payload lengths

5.6 Summary

In this chapter, we have shown that the proposed HW/SW co-verification framework can be extended for system-level by employing more tasks into hardware platform. This can

build a set of system-level simulator.

The full transceiver system based on 802.11ac standard that support multi-user transmission scheme was implemented as hardware part. The evaluation of the PHY transceiver is performed comprehensively: (1) by involving various system parameters, (2) by introducing the effect of hardware impairments, and (3) by considering the cross-layer interaction.

System-level error performance of standard-compliant transceiver system is carried out using the proposed HW/SW co-verification framework. Furthermore, the achievable throughput with the actual latency of hardware processing is performed. This evaluation is carried out in order to asses the total latency of system, that is critical to achieve the requirement of SIFS duration. The designed transceiver has been evaluated and give the total latency around 13 μs when operated in actual chip implementation. Clearly, for real-time SoC implementation of WLAN system, the remaining latency budget for other processing or tasks should be completed less than 3 μs. This result satisfies the upper layer protocol with the assumption that the upper layer can meet the available timing budget.

Chapter 6

Conclusion and Future Work

The main goal of this thesis is to provide a framework of co-verification methodology for large-scale SoC design. The proposed unified HW/SW co-verification framework is aimed to significantly improve the efficiency of development, design exploration, and evaluation of large scale LSI system, particularly for high throughput wireless communication system. Currently, the existing verification method is only suitable to be implemented in certain stage of the development. Therefore, the verification task should be carried out independently within each verification stage. It is also performed in different group of research and development team. This typical approach requires intensive turn-around design and verification process, and results longer cycle of design. Moreover, the comprehensive and reliable evaluations are difficult to be obtained.

The HW/SW co-verification methods is the optimal option for verification of complex circuits, considering the trade-off between system flexibility, verification run-time, and the development effort. However, the existing HW/SW co-verification has limited the integration with system level simulation and suffer to be implemented as integral part of system development. Hence, the simulation cannot cover verification task in various design levels concurrently, which are: algorithm validation, RTL simulation, and physical verification (in circuit verification). Finally, in order to address these requirements, the unified HW/SW co-verification framework is proposed to leverage the existing HW/SW co-verification methods.

The proposed methodology includes several contributions, which are:

1. The verification framework has flexible and scalable interface for data transfer between SW part (host PC) and HW part (FPGA). This feature enables design extension and could be applied to different system with only minor modifications.

2. The unified framework also allows verification for fast verification and design exploration by employing data-driven simulation method. This method can improves the run-time of verification and reduces the complexity of communication interface between SW and HW parts. This ability is particularly useful in verification of large scale system.

3. Proposed verification methodology supports the high level of design abstractions, such as MATLAB, C/C++, etc. This feature enables to perform system level simulation and allows for unified evaluation of various level of system design, from algorithm evaluation upto and physical level verification.

In order to show the applicability of the proposed framework, MLD MIMO Decoder and Full PHY transceiver for High Throughput Wireless Communication System are selected as application example. The Comprehensive evaluation includes the error performance and achievable throughput are carried out with hardware-in the loop fashion by employing various system parameters, channel conditions, as well as the impact of hardware impairments in the point of view system-level simulator. The performance metrics in term of verification run-time and verification efficiency are evaluated in order to show the effectiveness of proposed methodology where the speed of verification time achieve several orders of magnitude compare than pure software-simulation achieving near real-time simulation.

The MLD MIMO Decoder for high throughput wireless system is selected, as a representative of complex circuits. The implemented MLD MIMO decoder employs approximately 1.5 M gates count. The performance metric in term of verification efficiency, achievable run-time verification, and design exploration evaluations have been presented. The experimental evaluation using hardware-in-the loop verification scheme is also provided. Experimental results show that the verification speed up can achieve upto 1,000 times in single PE and upto 100,000 times in 64-parallel PE, with the verification efficiency almost 99%. By using this scheme, the proposed frame work is able to perform comprehensive

evaluation for various test scenarios in the point of view of system-level simulation, as shown by BER performance of whole wireless system.

In the second example, The full transceiver system based on 802.11ac standard that support multi-user transmission scheme was implemented as hardware part. The evaluation of the PHY transceiver can be performed comprehensively: (1) by involving various system parameters, (2) by introducing the effect of hardware impairments, and (3) by considering the cross-layer interaction.

System-level error performance of standard-compliant transceiver system is carried out using the proposed HW/SW co-verification framework. Furthermore, the achievable throughput with the actual latency of hardware processing is performed. This evaluation is carried out in order to asses the total latency of system, that is critical to achieve the requirement of SIFS duration. The designed transceiver has been evaluated and give the total latency around 13 μs when operated in actual chip implementation. Clearly, for real-time SoC implementation of WLAN system, the remaining latency budget for other processing or tasks should be completed less than 3 μs. This result will agree with the upper layer protocol requirements when the upper layer is already meet the available timing budget.

Although the proposed HW/SW co-verification is promising solution, however to achieve the user experience as offered by the latest commercial CAD tools, there are many open tasks and technical challenges that are feasible for future work:

1. Supporting multi-thread software execution to improve run-time simulation
   For verification various core processing with different software program, the multi-thread execution can be employed to allow the verification process can be performed concurrently. Hence, the simulation sequence can be done fully parallel both in the hardware side and software side. This later improves execution time. The implementation of multi-thread will be managed by scheduler which is a part of operating system. However, the synchronization issues in multi thread application will be more complex and the implementation must be careful to avoid race condition.

2. Providing debugging facility to observe various signals more flexible
   When the implemented hardware circuit is become more complex, the demand of debugging facility is increased. Essentially, in FPGA verification we can use its

inherit debugging facility, such as integrated logic analyzer (e.g. signal tap or chip scope) via JTAG interface. However, these debugging facilities have limited visibility and need to be defined before the implementation. Therefore, this approach is not flexible, particularly in a complex circuit. To address this issue, a multiplex-based test point can be included into design core in which it can tap-in various signals from the circuit. The selection can be controlled through register from software. With this implementation, we can choose any arbitrary signals during the runtime simulation without need to re-implement the design.

3. Integrating with Graphical user Interface to allow the user conveniently use the verification framework

   To more improve the benefit of framework, the availability of graphical user interface is considered. This front-end interface will allow the user to explore the co-verification framework and reduce a human-error in script-based simulation. This feature offers various user experiences in performing verification task, in particularly for user without comprehensive understanding on scripting language.

Acknowledgment

I am greatly indebted and would like to acknowledge the following individuals:

- Prof. Hiroshi Ochi for the guidance, supervision, and support during the doctoral course in Kyutech.
- Associate Prof. Masayuki Kurosaki, Assistant Prof. Leonardo Lanante, and Dr. Yuhei Nagao for providing assistance and giving insight comments in finishing this work.
- Prof. Akihiro Fujiwara, Prof. Yoshida Takeichi and Prof. Shingo Yoshizawa, who act as members of the graduation committee, that spend much time to read manuscript and give constructive comments.
- Fellow students, researches, and staffs : Khai, Khoa, Yokota, Nguyen, Chosokabe, Morita, Shinozaki, Haraguchi, Hongyo, Nico, Uwai, Kobayashi-san, Apiradee-san, Ogata-san, Urabe-san and many others with whom I was involved in many discussions and providing assistance in many ways during my stay in Japan.
- Fellow Indonesian students in Kyutech who provided me a good balance with out-of-work life.
- To whom undoubtedly I am indebted most; my wife and my family for the never-ending support and the extra bit of motivation to finish the doctoral course.

Bibliography

[1] C. Y Huang, et. al. ,SoC HW/SW Verification and Validation, *Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 297-300, 2011.

[2] H. Foster, Trends in Functional Verification, *Design and Automation Conference (DAC '15)*, 6 pages, June 2015.

[3] M. Pelcat, et. al., Design Productivity of a High Level Synthesis Compiler versus HDL, *2016 International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (IC-SAMOS 2016)*, July 2016.

[4] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification*, IEEE 802.11ac/D5.0, 2013

[5] J. Teich, Hardware/Software Codesign: The Past, The Present, and Predicting the Future, *Proceedings of the IEEE*, Vol. 100 Special Centennial Issue, pp. 1411-1430, 2012.

[6] L. Semeria, et/ al, Methodology for Hardware/Software Co-verification in C/C++, *Design Automation Conference*, 6 pages, 25-28 Jan. 2000 .

[7] Y. Nakamura, et. al., A Fast Hardware/Software Co-Verification Method for System-On-a-Chip by Using a C/C++ Simulator and FPGA Emulator with Shared Register Communication, *Design Automation Conference*, pp. 299-304, 2004.

[8] S. Hong, et. al, A Case of System-level Hardware/Software Co-design and Co-verification of a Commodity Multi-Processor System with Custom Hardware, *Proceedings of the 8th IEEE/ACM/IFIP International Conference on Hardware/software Codesign and System Synthesis*, pp. 513-520, 2012.

[9] T. Koike, et. al., Prototype Implementation of Real-Time ML Detector for Spatial Multiplexing Transmission, *IEICE Transaction of Communication*, Vol.E89-B, No. 3, pp. 845-852, Mar. 2006.

[10] L. G. Barbero and J. S. Thompson, Rapid Prototyping System for the Evaluation of MIMO Receive Algorithms, *The International Conference on Computer as a Tool (EUROCON)*, pp. 1779-1782, 2005.

[11] A. Alimohammad, et. al, FPGA-based Accelerator for the Verification of Leading-Edge Wireless System, *Design Automation Conference (DAC'09)*, pp. 844-847, 2009, 2015.

[12] M. Wenk, et. al., Hardware platform and implementation of a real-time multi-user MIMO-OFDM testbed, *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 789-792, 2009.

[13] Ishihara et al., Development and experimental validation of downlink multiuser MIMO-OFDM in gigabit wireless LAN systems, *EURASIP Journal on Advances in Signal Processing 2013*, 2013:123

[14] J. Goeders, et. al., Signal-Tracing Techniques for In-System FPGA Debugging of High-Level Synthesis Circuits, *IEEE Transaction on Computer-Aided Design of Integrated Circuit and Systems*, Vol. 36, No. 1, pp. 83-96, January 2017.

[15] G. Liang, et. al., A Hardware In The Loop Design Methodology for FPGA System and Its Application to Complex Functions, *International Symposium on VLSI Design, Automation, and Test (VLSI-DAT)*, pp. 1-4 2012.

[16] S. Skalicky, et. al, Enabling FPGA Support in MATLAB based Heterogeneous System, *International Conference on ReConFigurable Computing and FPGAs (ReConFig14*, pp. 1-6, 204. DOI: 10.1109/ReConFig.2014.7032515

[17] C. de Schryver, et. al., Lopy : An Open-source TCP/IP Rapid Prototyping and Validation Framework, *International Conference on Reconfigurable Computing and FPGAs(ReConFig)*, pp. 1-6, 2013.

[18] H. K. So, et.al, A Unified Hardware/Software Runtime Environment for FPGA-Based Reconfigurable Computer using BORPH, *Proceedings of the 4th International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS '06)*, pp. 259-264, 2006.

[19] J. Cong, et. al., High-Level Synthesis for FPGAs: From Prototyping to Deployment, *IEEE Transaction on Computer-Aided Design of Integrated Circuit and Systems*, Vol. 30, No. 4, pp. 473-488, 2011

[20] R. Nane, et. al., A survey and Evaluation of FPGA High-Level Synthesis Tools, *IEEE Transaction on Computer-Aided Design of Integrated Circuit and Systems*, Vol. 35, No. 10, pp. 15911604, October 2016,

[21] *HAPS-60 Series User Guide*, Synopsys, 2013.

[22] *UMRBus Communication System Handbook v3.11*, Synopsys, 2012.

[23] *Synopsys Hybrid Prototyping Solution*, Synopsys.
Available at http://www.synopsys.com/prototyping/fpgabasedprototyping/pages/hybrid-prototyping.aspx, accessed on Feb 2016.

[24] L. Zheng and D.N. C. Tse, "Diversity and Multiplexing: A Fundamental Tradeoff in Multiple-Antenna Channels", *IEEE Transaction on Information Theory*, Vol. 49, No. 5, pp. 1073-1096, May 2003.

[25] C. Siriteanu, et al., "MIMO zero-forcing detection analysis for correlated and estimated Rician fading", *IEEE transactions on Vehicular Technology*, vol. 61, no. 7, pp. 3087-3099, 2012.

[26] D.K.C. So and R.S. Cheng, "Layered maximum likelihood detection for MIMO systems in frequency selective fading channels", *IEEE Transactions on Wireless Communications*, vol. 5, no. 4, pp. 752-762, 2006.

[27] S. Yoshizawa, et. al., "VLSI implementation of a scalable pipeline MMSE MIMO detector for a 4×4 MIMO-OFDM receiver", *IEICE Trans. on Fundamentals*, Vol. E94-A, No. 1, pp. 324-331, Jan. 2011.

[28] C.Z.W.H. Swetman, et. al., "A comparison of the MMSE detector and its BLAST versions for MIMO channels", *IEE Seminar on MIMO: Comm. Sys. from Concept to Implementations*, 2001, pp. 19/1-19/6.

[29] R. Ragumadhavan, "Hardware-optimized Lattice Reduction Algorithm for WiMax/LTE MIMO detection using VLSI," *Inter. Journal of Computer Science and Information Technology (IJCSMC)*, Vol. 2, No. 4, pp. 146-154, Apr. 2013.

[30] M. Wu, et.al., A GPU Implementation of Real-Time MIMO Detector, *IEEE Workshop on Signal Processing Systems (SiPS)*, pp. 303-308, 2009.

[31] Z. Chai, et. al, A Low Complexity and High Throughput MIMO Detection VLSI Design for MIMO-OFDM Systems, *IEEE 83rd Vehicular Technology Conference (VTC Spring)*, 5 pages, 2016.

[32] T. Givargis, et. al., System-Level Exploration for Pareto-Optimal Configuration in Parameterized System-on-Chip, *IEEE Transaction on very Large Scale Integration (VLSI) Systems*, vol. 10, No. 4, pp. 416-422, Aug. 2002.

[33] S. Shibata, et. al, A Fast Performance Estimation Framework for System-Level Design Space Exploration, *IPSJ Transaction on System LSI Design Methodology*, Vol. 5 pp. 44-54, Feb. 2012.

[34] M. Gries, Methods for evaluating and covering the design space during early design development, Integration, the VLSI journal, Vol. 38 No. 2, pp. 131-183, 2004.

[35] A. Prost-Boucle, et. al., Fast and standalone Design Space Exploration for High-Level Synthesis Under Resource Constraints, *EUROMICRO Journal of System Architecture*, Vol. 60 Issue 1, pp. 79-93 , Jan. 2014.

[36] M. Kock, et. al., Hardware-accelerated design space exploration framework for communication system, *Analog Integrated Circuits and Signal Processing*, Vol. 78 Issue 3, pp. 557-571, March 2014.

[37] Xilinc Inc.,"FPGA and ASIC Technology Comparison", available online, accessed on Feb 2017.

[38] I. Kuon and J. Rose, "Measuring the Gap Between FPGAs and ASICs", *IEEE Transactions on Computer-Aided Design and Circuit and System (TCAD)*, vol. 26, no. 2, pp. 203-215, February 2007.

[39] C. Mehlfuhrer, et. al., The Vienna LTE Simulators - Enabling reproducibility in wireless communication research, *EURASIP Journal on Advances in Signal Processing*, Vol. 2011:29, 2011. DOI: 10.1186/1687-6180-2011-29

[40] Y. Saito, et. al., System-Level Performance Evaluations of Downlink Non-orthogonal Multiple Access (NOMA), *IEEE 24th International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC 2013)*, 5 pages, Sept. 2013.

[41] A. Benjebbor, et. al., System-level Performance of Downlink NOMA for Future LTE Enhancements, *IEEE Globecom Workshops*, pp. 66-70, Dec. 2013.

[42] P. Griesen, et. al., Simulation and Emulation of MIMO Wireless Baseband Transceiver, *EURASIP Journal on Wireless Communication and Networking*, Volume 2010, 12 pages, 2010. DOI:10.1155/2010/196796

[43] T. Wirth, et. al., Real-time Multi-user Multi-Antenna Downlink Measurements, *IEEE Wireless Communications and Networking Conference (WCNC 2008)*, pp. 1328-1333, March 2008.

[44] S. Haene, et. al., A Real-Time 4-Stream MIMO-OFDM Transceiver: System Design, FPGA Implementation, and Characterization, *IEEE Journal on Selected Areas in Communications*, vol 26. No. 6, Aug. 2008.

[45] H. Murata, et. al, Software Radio-Based Distributed Multi-user MIMO Testbed: Towards Green Wireless Communications, *IEICE Transaction Fundamentals Special Section on Wideband Systems (Invited paper)*, Vol. E96-A, No. 1, pp. 247-254, 2013.

[46] I. Lee, et. al., Effective co-verification of IEEE 802.11a MAC/PHY combining emulation and simulation technology, *Proceedings. 38th Annual Simulation Symposium*, pp. 1-9, 2005.

[47] T. Hiraguri and K. Nishimori., Survey of Transmission methods and Efficiency Using MIMO Technologies for Wireless LAN Systems, *IEICE Transaction on Communications*, Vol.E98-B, No. 7, pp. 1250-1267, July 2015.

Publication List

Journals

1. Nana Sutisna, Reina Hongyo, Leonardo Lanante Jr., Yuhei Nagao, Masayuki Kurosaki, and Hiroshi Ochi, *"Unified HW/SW Co-verification Methodology for High Throughput Wireless Communication System"*, IPSJ Transaction on System LSI Design Methodology, Vol.9, pp.61-71, Aug. 2016. DOI:10.2197/ipsjtsldm.9.61

International Conferences

1. Nana Sutisna, Leonardo Lanante Jr., Yuhei Nagao, Masayuki Kurosaki, and Hiroshi Ochi, *"Live Demonstration: Hardware-Software Co-verification for Very Large Scale SoC Using Synopsys HAPS Platform"*, 2014 IEEE Asia Pacific Conference on Circuit and Systems (APCCAS 2014), Okinawa-Japan, pp. 171-172, Nov. 2014. DOI: 10.1109/APCCAS.2014.7032746

2. Nana Sutisna, Leonardo Lanante Jr., Reina Hongyo, Yuhei Nagao, Masayuki Kurosaki, Hiroshi Ochi, *"Fast Design Exploration with Unified HW/SW co-verification Framework for High Throughput Wireless Communication System"*, 2016 IEEE International Conference on IC Design and Technology (ICICDT), Vietnam, pp. 1-4, March 2016. DOI:10.1109/ICICDT.2016.7542059

3. I. Syfalni, N. Surantha, D. K Lam, N. Sutisna, Y. Nagao, K. Wakasugi, Y. Tongxin, H. Ochi, T. Tsuchiya, *"Assertion-Based Verification of Industrial WLAN System"*,

IEEE International Symposium on Circuit and System (ISCAS), Montreal-Canada, pp. 982-985, May, 2016. DOI:10.1109/ISCAS.2016.7527407

4. Nana Sutisna, Leonardo Lanante Jr., Yuhei Nagao, Masayuki Kurosaki, Hiroshi Ochi, *"Unified HW/SW Framework for Efficient System Level Simulation"*, 2016 IEEE Asia Pacific Conference on Circuit and Systems (APCCAS 2016), Jeju-South Korea, pp. 518-521, Oct. 2016. DOI: 10.1109/APCCAS.2016.7804018

Seminars

1. Nana Sutisna, Leonardo Lanante Jr., Yuhei Nagao, Masayuki Kurosaki, and Hiroshi Ochi, *"Hardware-Software Co-verification for Very Large Scale SoC Using Synopsys HAPS Platform"*, International Symposium on Dependable Integrated Systems (DISC), Fukuoka, March 2015.

2. Nana Sutisna, Reina Hongyo, Leonardo Lanante Jr., Yuhei Nagao, Masayuki Kurosaki, and Hiroshi Ochi, *"Fast HW/SW Co-Verification Platform for High Throughput Wireless Communication System"*, International Symposium on Dependable Integrated Systems (DISC), Fukuoka, March 2016.