TECHNIQUE OF PVT ANALYSIS ON SD CONTROLLER TIMING VALIDATION FOR 28NM SOC FPGA

By

NUR AMALINA AIZA BINTI YUSNI

A Dissertation submitted for partial fulfilment of the requirement for the degree of Master of Science (Electronic Systems Design Engineering)

August 2015

ACKNOWLEDGEMENT

First of all I feel so grateful and thanks to ALLAH whom giving me the opportunity to finish this thesis. I would like to convey my appreciation to my supervisor, Dr. Anwar Hasni Abu Hassan who patiently assisted and guided me along with the project. It is such a challenge to me for taking this path and I could not imagine what will happen without his support, advice and encouragement.

Secondly, I would like to thank both of my parents for their support, motivation and love from the first day of my project until the end. They always give support to me even when I felt so down. I am really appreciates their loves and cares.

Last but not least, great appreciation goes to my friends that help me in completing this project and thesis. Thank you also for their support and advices in order to motivate me. I would appreciate their kindness. Thank you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

SoC	System on Chip
IP	Intellectual Property
SD	Secure Digital
MMC	Multimedia Card
Tsu	Setup Time
Th	Hold Time
Tco	Output Delay
ATE	Automatic Test Equipment
IO	Input/Output
FPGA	Field Programmable Gate Array
DUT	Device Under Test
CV	Cyclone V
PVT	Process, Voltage, Temperature
HPS	Hard Processing System
PLL	Phase Locked Loop
PCIe	Peripheral Components Interconnect Express
DMA	Direct Memory Access
BIU	Bus Interface Unit
CIU	Card Interface Unit
CSR	Control and Status Register
FIFO	First In First Out
CLK	Clock
DAT	Data
CMD	Command
Clkdiv	Clock Divider
Clksrc	Clock Source
Clkena	Clock Enable
CRC	Cyclic Redundancy Check
SS	Slow Corner Unit
TT	Typical Cornet Unit
FF	Fast Cornet Unit

TEKNIK PVT ANALISIS BAGI PENGESAHAN MASA PENGAWAL SD UNTUK 28NM FPGA SOC

ABSTRAK

SoC adalah system di dalam satu cip yang terdiri daripada memori, pemproses dan peranti. Di dalam SoC terdapat banyak blok IP seperti blok pengawal memori digital, SD. Hari berganti hari, SoC mempunyai peningkatan permintaan di dalam industri. SD telah digunakan secara meluas sebagai penyimpan data kerana ia mempunyai saiz yang mudah untuk dibawa dan mempunyai kapasiti yang besar. Bagi mengekalkan kualiti IP, selain mengesahkan fungsinya, ia juga adalah perlu untuk mengesahkan bahawa pengawal SD telah memenuhi spesifikasi masa. Kebiasaannya, untuk mengukur masa bagi IP yang mempunyai protokol adalah sukar kerana ia perlu mengukur masa untuk parameter IO (Tco, Tsu dan Th). Tetapi, masalahnya ialah kebanyakkan alat tidak sesuai untuk mengukur masa. Analisa logik hanya mampu menangkap masa tetapi tidak boleh melaraskan masa. ATE mungkin dapat melaraskan masa tetapi untuk menyediakan protokol pengaturcaraan mengambil masa berbulan-bulan. Tambahan lagi, ATE tidak boleh menampung protokol dengan unsur-unsur masa yang tidak tentu. Bagi menyelesaikan masalah ini, mengurangkan masa untuk pengesahan dan meningkatkan pencirian masa yang berkaitan dengan protokol IO masuk, tesis ini akan memperkenalkan kaedah baru yang menggunakan konfigurasi, kandungan yang berkualiti dan unsur kelewatan yang mempunyai protokol. Kaedah tambahan yang akan digunakan didalam projek ini adalah menggunakan papan kelewatan masa yang mengandungi cip Max II. Max II cip akan mempunyai reka bentuk kelewatan untuk memberikan kelewatan kepada jam masuk. Pengguna boleh mengawal penetapan kelewatan dengan mengubah pin pemilih pada papan kelewatan masa yang mempunyai tiga bit penetapan berbeza jalan kelewatannya. Bagi memenuhi spesifikasi lembaran data, kelewaan keluaran juga diukur menggunakan protocol arahan. Berdasarkan analisis yang dilakukan, unit SS pada kondisi Vmin dan suhu sejuk memberikan kes terburuk bagi pencarian parameter untuk masa.

TECHNIQUE OF PVT ANALYSIS ON SD CONTROLLER TIMING VALIDATION FOR 28NM SOC FPGA

ABSTRACT

SoC is a system on chip that consists of memory, processor and peripherals. Inside SoC there are many IP blocks such as secure digital (SD) controller block. As time fly, SoC has increase their demand in the industries. SD was widely used as the data store since it is compatible and have a large capacity. To remain the quality of the IP, beside functional validation, it is necessary to ensure the SD controller meet its timing specification. Typically, for protocol timing characterization it is difficult to adjust the timing as it needs to properly measure the input IO timing parameters (Tco,Tsu and Th). However, the problem is most tools do not adjust timing for measurement. Logic analyzers can only captures timing, but do not have the capabilities for user to adjust the timing. Full size ATE testers might be able adjust timing, but programming protocol aware test vectors takes many months of development per protocol. Additionally, ATEs cannot accommodate protocols with non-deterministic timing elements. To eliminate these problems, reduce characterization time, and improve timing related to IO protocol characterization, this thesis will introduce a new methodology using a configurable, high granularity, protocol aware delay element. The additional methodology that will be used in this project is using an additional timing delay board which consist of Max II device. The Max II device will have a delay design inside to provide delay to input clock. User can control the setting of the delay by changing the external selector pin on the timing delay board. The external pin is three bit setting from 000 to 111. Each of the setting will have a different delay path depends on the design in Max II. Using this methodology, an analysis on PVT has been conducted to get the worst case of the minimum setup time and hold time. To complete the timing datasheet for SD controller, output delay has been measure using command protocol. Based on the analysis, SS corner at Vmin and cold temperature condition gives the worst minimum setup time (2.56ns), minimum hold time (1.5ns) and output delay.

CHAPTER 1

INTRODUCTION

1.1 Project Background

With the technology advancement in today's embedded system such as mobile and electronic devices marketplace, which is smartphone, tablets, notebooks, digital camera, and video camera, there is increasing demands on SoCs (System on a chip). SoC is a system in a single microchip that has a package of all computer components such as processor, peripheral and memory. Incrementing on SoC demand has led to the SoC FPGA development where the IP blocks can interface with FPGA fabric.

SoC FPGA chips have been used widely and it takes the advantages on the rising of mobile and electronic devices demand which used high speed speech and image processing. SoC FPGA consists of SoC component and FPGA fabric in a single chip and this is why it is gaining its own admire from time to time because while designing a design using FPGA, user can interface the design with the SoC component without a need to have a separate circuit for SoC.

In order to process the speech and images, it is important to have storage device to store the voice and image data. SD (secure digital) card is a non-volatile flash memory data storage device used to store digital information that popular among semiconductor industry for memory devices. To manage the SD card, SD host controller is needed. SD host controller is one of a must peripheral that need to

be supported in SoC chip. Since SoC chip is quite new in market, typical practice to validate the working of SD controller in SoC chip is through functional validation only. However, this will only approved the functionality of SD controller where it can write and read data to SD card but not proving the timing performance of the controller. Timing validation is important to make sure the function acts properly within correct timing spec. The normal timing specifications for SD consist of output delay, setup time and hold time.

Validation process on the bench level needs a board which called characterization board. As in early development of SoC chip the focus are more on functional validation so the characterization board are lack in terms of timing validation characteristic on the board. The SD timing delay board was designed, therefore in this thesis the timing validation methodology for three timing parameter which is output delay time, Tco, setup time, Tsu and hold time, Th will be created and the measurement results will be analysed to make sure the chip is in a good performance. The chips used in this project are Cyclone V SoC FPGA chip.

1.2 Problem Statement

SD controller in SoC is not similar as SD controller design inside FPGA. The SD controller in the SoC is a hardware that already implement in a single chip. The hardware is already fixed and cannot be change as in FPGA. Since the hardware already there, it is difficult to do timing validation instead, all the attention that manufacturers company put focus are on verifying the functionality of the SD controller in SoC chip. But, they had forgotten about the timing specification. Typically, for protocol timing characterization it is difficult to adjust the timing as needed to properly measure the timing parameters. Most tools do not adjust timing for measurement. Logic analyzers can only capture timing, but user cannot adjust the timing.

The purpose of timing validation on communication between SD controller and SD card is to guarantee that the time for data in and out is within certain range of value. These can assure that the chip's performance is good and lead to widen the market share and increase producer net profit.

Furthermore, during past few years in characterization board design for early SoC chip validation, the board was designed to do a functional test validation which is write and read function to or from SD card only without timing characteristic on the board. So, current characterization board need to be modified to have this capability.

1.3 Objective

Referring to the problem statement, the main objective for this project is to get the timing performance of the SD controller using SoC CV chip across PVT. Timing performance characteristic that will be analysed are output delay, Tco, setup time, Tsu and hold time, Th. To achieve this objective, an additional SD timing delay board is needed in order to do the timing validation for setup and hold time. The SoC FPGA chip was chosen to be tested using normal characterization board combine with SD timing delay board. The other objectives of this project are:

i. To develop a methodology to do timing validation to get output delay, setup time and hold time for SD controller in SoC chip.
ii. To analyse the performance of SD controller in terms of the output delay, minimum setup time and hold time specification across PVT.

1.4 Scope of work

The scope of this project is it only covers the timing validation process part on the SD controller of a Cyclone V SoC chip. The validation part is including the methodology used to measure the timing parameter, the data collection and data analysis. The timing parameter measured is output delay, Tco, setup time, Tsu and hold time, Th. The test will be conducted on the bench using Altera Cyclone V characterization board with additional timing delay board to measure setup and hold time. This thesis will not discuss about the additional timing delay board design as it is confidential. All tools and equipment used are Altera lab tools and equipment.

1.5 Thesis Organisation

This thesis is organized into five chapters. Chapter 1 of this thesis will cover the introduction of the SoC FPGA and the effect of existing validation plan. It briefly discussed the needs on the new methodology and the reasons of the needs.

Chapter 2 focuses on the literature review on topic regarding SD controller block understanding, the typical functional validation, overview on the bench level testing, the challenge on the bench level testing and ATE testing and some works on the tuneable timing control. Since SD controller is a protocol IP, this chapter also describe in details what protocol SD controller used.

Chapter 3 encompasses the development methodology proposed as the solution for current workaround. The new methodology to find minimum setup and hold time requires an additional timing delay board that consist of Max II device. This chapter will not review the board schematic design but will describe the features of the delay board that related to this project. Moreover, in this chapter also will describe about the delay design inside Max II which are used to give a delay to the input clock of the SD controller for timing input parameters measurement. Furthermore, the overall setup needed for this methods also will be review.

Chapter 4 presents the data collected during the experiment. Data collection has been collected across PVT which is process, voltage and temperature. Data collection for minimum setup and hold time plotted in a scatter graph and linear equation has been generated to find the minimum setup and hold time for each process. While, for output delay data, table was used for analysis. This chapter also show the effect of the PVT in measurement.

Finally, in chapter 5 covers the conclusion of this project based on the results and finding in chapter 4. This chapter also include suggestion for future development.

CHAPTER 2

LITERATURE REVIEW

2.1 Cyclone V Hard Processor System (HPS) Overview

Cyclone V SoC FPGA device is a single-die system on chip (SoC) that contains of two parts, a hard processor system (HPS) portion and FPGA portion[1]. Figure 2.1 below show the block diagram of the Cyclone V SoC device.

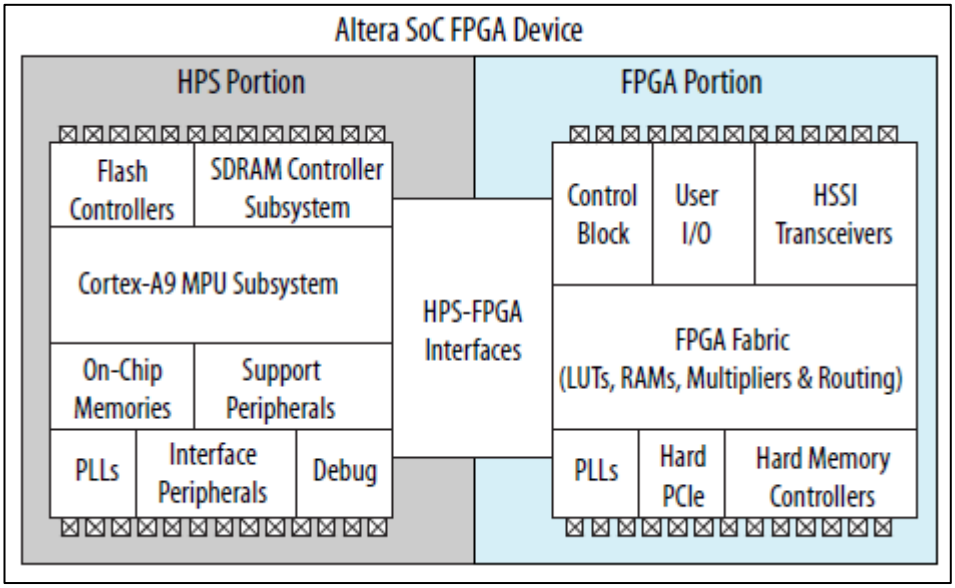

Figure 2.1 Altera SoC FPGA Device Block Diagram [1]

As shown in the diagram, HPS portion consist of microprocessor unit (MPU) subsystem with ARM Cortex A9 MPcore processor, SDRAM controller subsystem, on-chip memories, interface peripherals, debug capabilities and phase-locked loop (PLLs). While FPGA portion consist of phase-locked loop (PLLs), a control block, FPGA fabric and depending on the device variant, high-speed serial interface (HSSI) transceivers, hard PCI Express (PCIe) controllers and hard memory controllers. Having all the processors, peripherals and controllers in one chip does help to reduce the board space and the system cost.

Both HPS and FPGA portions have their own pins in which the pins cannot be shared between HPS and FPGA. Below are the I/O pins types.

- Dedicated I/O – I/O that is only for external non-volatile storage device (for boot flash), HPS clock and resets.
- Shared I/O – I/O that can be assigned to peripherals in the HPS or FPGA.
- FPGA I/O – I/O that is for FPGA fabric. Some of the peripherals in HPS can route signals to FPGA fabric by connecting to the FPGA I/O instead of using HPS I/O.

The following are the features covers in HPS. The main related features to the project is the secure digital/multimedia card (SD/MMC) controllers and a part of SD/MMC sub features in system manager features.

- MPU subsystem featuring a single or dual ARM Cortex-A9 MPCore processors
- General-purpose direct memory access (DMA) controller
- Two ethernet media access controllers (EMACs)
- Two USB 2.0 on-the-go (OTG) controllers
- NAND flash controller
- Quad SPI flash controller
- Secure digital/multimedia card (SD/MMC) controller
- Two serial peripheral interface (SPI) master controllers

- Two SPI slave controllers
- Four inter-integrated circuit (I2C) controllers
- 64 KB on-chip RAM
- 64 KB on-chip boot ROM
- Two UARTs
- Four timers
- Two watchdog timers
- Three general-purpose I/O (GPIO) interfaces
- Two controller area network (CAN) controllers
- ARM Coresight™ debug components:
- Debug access port (DAP)
- Trace port interface unit (TPIU)
- System trace macrocell (STM)
- Program trace macrocell (PTM)
- Embedded trace router (ETR)
- Embedded cross trigger (ECT)
- System manager
- Clock manager
- Reset manager
- Scan manager
- FPGA manager
- SDRAM controller subsystem

Figure 2.2 shows the HPS block diagram that contains all the features. The main feature that will be discuss in this project is SDMMC controller which is a flash controller inside Cyclone V SoC FPGA chip as shown in red box in Figure 2.2.

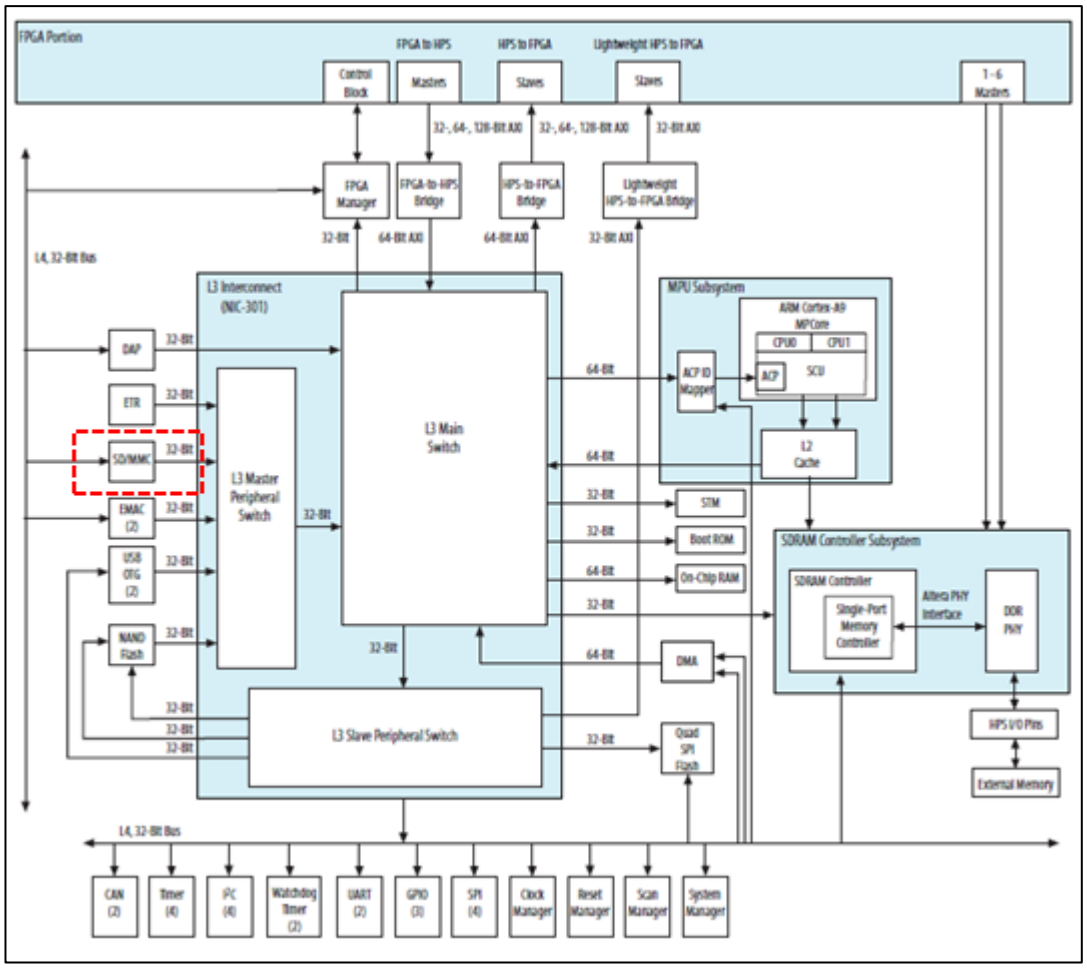

Figure 2.2 HPS Block Diagram[2]

2.2 SD/MMC Controller Overview

As advancement of SoC FPGA chips rising and widely known by the industry, SD/MMC card was used widely especially in speech and image processing application. Secure Digital (SD) card and Multimedia card (MMC) is a low cost and removable flash memory device that have a large storage capacity. Since the speech and image processing demands for a large storage to store large voice and image data with low-cost and micro-power requirements, SD or MMC card is used to expand the on-board storage capacity and the most suitable option for these applications.

SD/MMC card needs a controller to manage the interfacing of the cards. Hard processor system (HPS) will provides this controller to manage communication to the external SD and MMC cards. Besides write and read function, the SD/MMC controller also enables user to store boot images and boot the processor system from external flash card.

2.2.1 Bus Interface Unit (BIU) and Card Interface Unit (CIU)

The SD/MMC controller consists of two interfaces, a bus interface unit (BIU) and a card interface unit (CIU). The BIU provides a slave interface for a host to access the control and status registers (CSRs). Furthermore, this interface also provides independent FIFO buffer access through a DMA interface. The DMA responsible for data exchange between FIFO buffer and system memory. While BIU support more on the interfacing of the processor and SD/MMC controller, CIU support more on the interfacing of external card. CIU supports the SD and MMC protocols on the controller and also provides clock management through the clock control block.

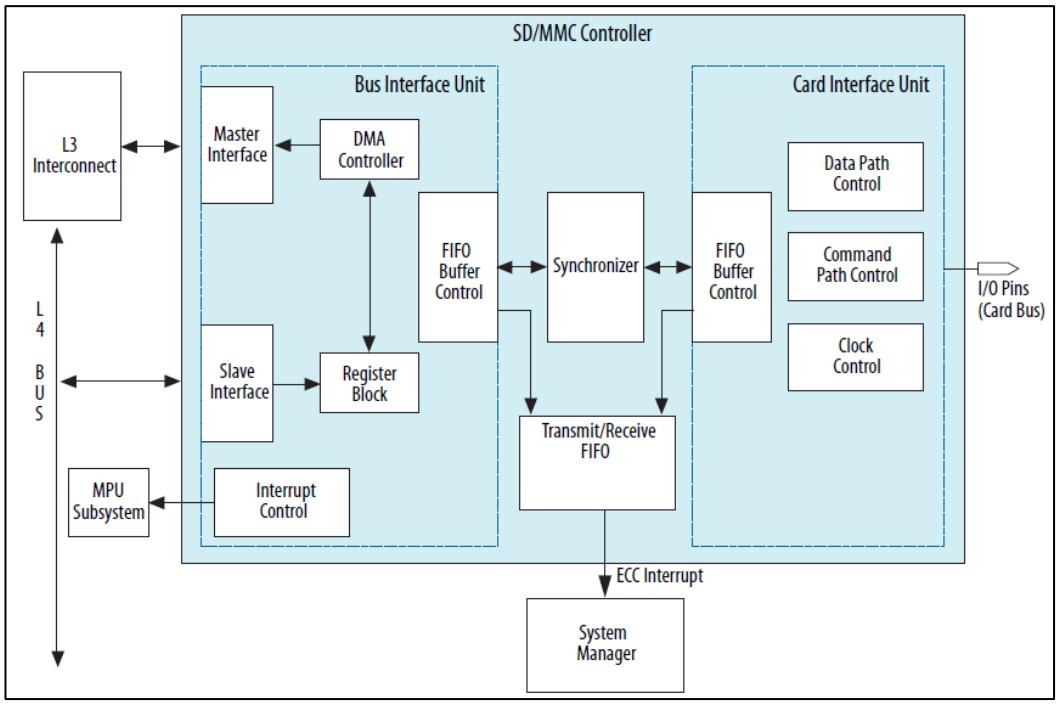

Figure 2.3 SD/MMC Controller Block Diagram

Figure 2.3 shows the SD/MMC controller block diagram. As in figure, the BIU interfaces with the CIU and connected to level 3 (L3) interconnect and level 4 (L4) bus. For this SD/MMC controller, the BIU consist of the following main functional blocks[2]:

1) **Slave interface:** This interface is used for the host processor to access the SD/MMC controller registers and data FIFO buffers.
2) **Register block:** Registers is used to provide the write and read function to CSRs. All the setting used for SD/MMC controller will be in the registers. All registers are inside BIU. All relevant registers needed for the CIU operation are copied to the CIU block once the setting of start command bit (start_cmd) of the command register (cmd) is 1 after a command was sent to a card.
3) **Interrupt controller unit:** The interrupt is generate depends on raw interrupt status register (rintsts), interrupt mask register (intmask), and the interrupt enable bit (int_enable) of the control register (ctrl). An active high level sensitive interrupts will be generated by the interrupt controller when at least one bit in the rintsts register is set to 1, the corresponding intmask register bit

is 1, and the int_enable bit of the ctrl register is 1. Once an interrupt activity is detected, the controller sets the related interrupt bit in the rintsts register. The software need to clear the bit by writing a 1 to the interrupt bit, otherwise the bit will remained set. Writing a 0 leaves the bit untouched.

4) **FIFO buffer:** This buffer is used to store transmit and receive data.
5) **DMA controller:** DMA controller is used to control data transmission between system memory and the SD/MMC card. Data can efficiently transfer from system memory to card with minimal host processor intervention by using DMA.

As mention above, CIU support more on interfacing with external cards. Figure 2.3 shows that CIU is interfaces with the BIU and external I/O which is SD/MMC cards or devices. BIU and CIU need each other. Once the communication between host and card exist, the host processor will writes command parameters to the BIU's control register and after that these parameter are then passed to the CIU. CIU then will take control and communicate with cards by generate SD/MMC command and data traffic on the card bus. All command and data is generated depending on the SD/MMC protocol. The CIU consist of the following main functional blocks:

1) Command path: Command path is mainly for send commands to the card and receiver response from card.
2) Data path: Data path is mainly to send and receive data. The data path block will read the data FIFO buffer and if there is data in the FIFO the data is transmit to the card bus during write operation and the data path will write the data received from card to the received FIFO during read operation.
3) Clock control: the clock control block provides different clock frequency to the SD/MMC cards. More detail in section 2.2.4

2.2.2 Clock Control Block

As mentioned in above, the clock control block provides different clock frequency for SD/MMC cards. There are three clock registers that control SD/MMC controller clock which are:

1) **Clkdiv register:** This register controls different clock generation that require by the cards. The value of the clock divider frequency can be control by setting the clkdiv register. The clkdiv register is a 8-bit value (N=8) that provide a clock division factor from 1 to 510. The value 0 means the clock is bypass which is whatever clock in is the output clock also, value 1 (N=1) means the clock is divided by 2 (2^1=2), value 2 (N=2) means the clock is divided by 4 (2^2=4) and so on.
2) **Clksrc register:** This is the clock divider source register. If the clock is going to divide by clock divider number 0, then set this to 0. If the clock divider number 1 is being used, set this to 1 and so on.
3) **Clkena register:** This is to enable and disable the clock output (clk_out) to the card. Set to 1 to enable and 0 to disable.

2.2.3 SD/MMC Controller Clock Connection

There are 6 clocks for SD/MMC controller shows in Table 2.1 below. The main sdmmc_clk is from clock manager and will go through a divider by four. There will be three outputs from divider by four which are sdmmc_clk divided and two clock input to phase shifter. Refer to Figure 2.4 for a clear explaination.

Table 2.1 SD/MMC Controller Clocks

Clock Name	Direction	Description
l4_mp_clk	In	Clock for SD/MMC controller BIU
Sdmmc_clk	In	Clock for SD/MMC controller
Sdmmc_clk_out	Out	The output clock for card
Sdmmc_clk_divided/ clk_in	Internal	Divide-by-four clock of sdmmc_clk
Sdmmc_sample_clk	Internal	Phase-shifted clock of sdmmc_clk_divided used to sample the command and data from the card
sdmmc_drv_clk	Internal	Phase-shifted clock of sdmmc_clk_divided for controller to drive command and data to the card to meet hold time requirements

The sdmmc_clk_divided clock also known as clk_in clock to the SD/MMC controller. The phase shifter is used to generate sdmmc_drv_clk and sdmmc_sample_clk. There are eight phase shift provide by the phase shifter which include 0, 45, 90,135, 180, 225, 270 and 315 degrees. The setting for the phase shifter degree has been done in system manager. These will be discuss in the next section. The sdmmc_clk_out is generated by the controller and this is the clock that is driven to the card.

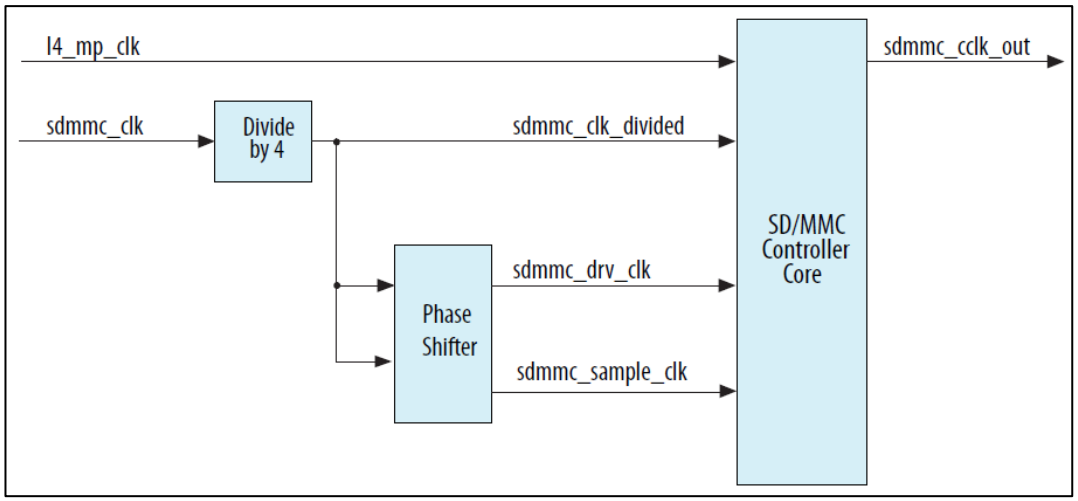

Figure 2.4 SD/MMC controller clock diagram[2]

2.2.4 SD/MMC Bus Protocol

The SD/MMC protocol is based on command, response and data bit. All of these are initiated by a star bit and terminate by a stop bit. The controller provide a reference clock to the card and only the master can initiate a transaction.

- Command: A token that start an operation which transmitted serially on the CMD pin. A command is sent from the host either to a connected card.
- Response: A token that serially transmitted from the card to the host indicates an answer to a command. The transmission is on CMD pin.
- Data: Data is transferred via the data line from the host to the cards or vice versa.

Figure 2.5 shows the basic transaction on the SD bus which consists of command and response. A command is sent in CMD line. If there is response, a response is send from the card to the host after a command is transmitted.

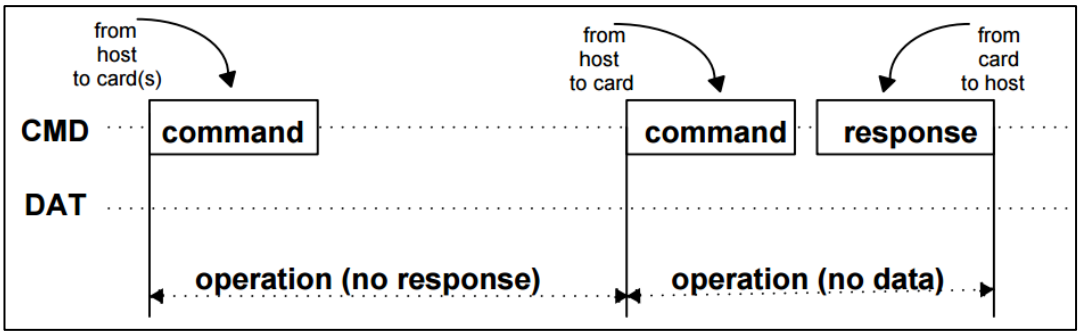

Figure 2.5 Command and Response Operations[3]

Data transfer to or from SD card are done in a blocks. Each of data blocks will always followed by a CRC bit to indicates there is no error in data transmission. There are single and multiple blocks operation. For a single block operation, every time a data being transmit, it will start by a command and response to start the data transmission and end by the command and response to stop the transmission. While for multiple block operation, after a command and response being transmit to start the transmission, all the data will be transmit after one another until the last data being transmit only then a command and response to end the transmission will be send. Figure 2.6 shows the multiple block for read operation while Figure 2.7 shows multiple block for write operation.

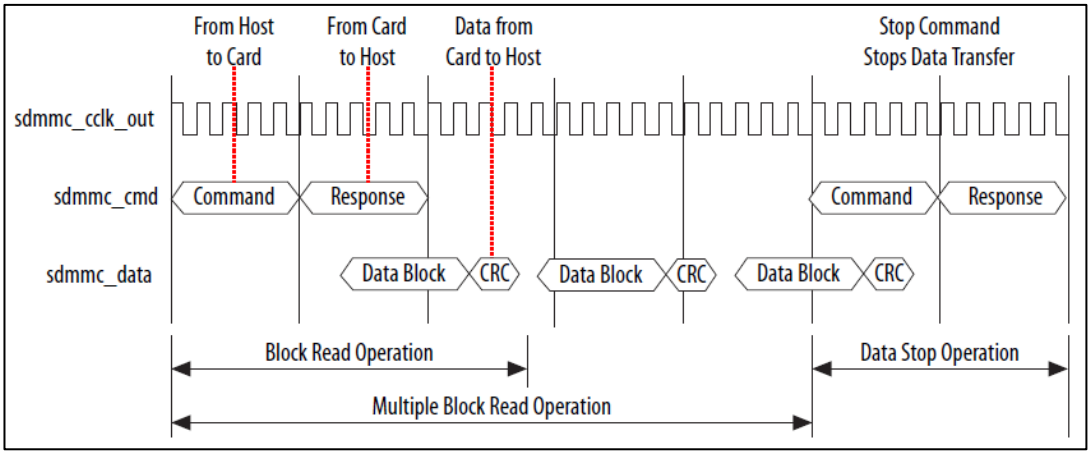

Figure 2.6 Multiple Block Read Operation[3]

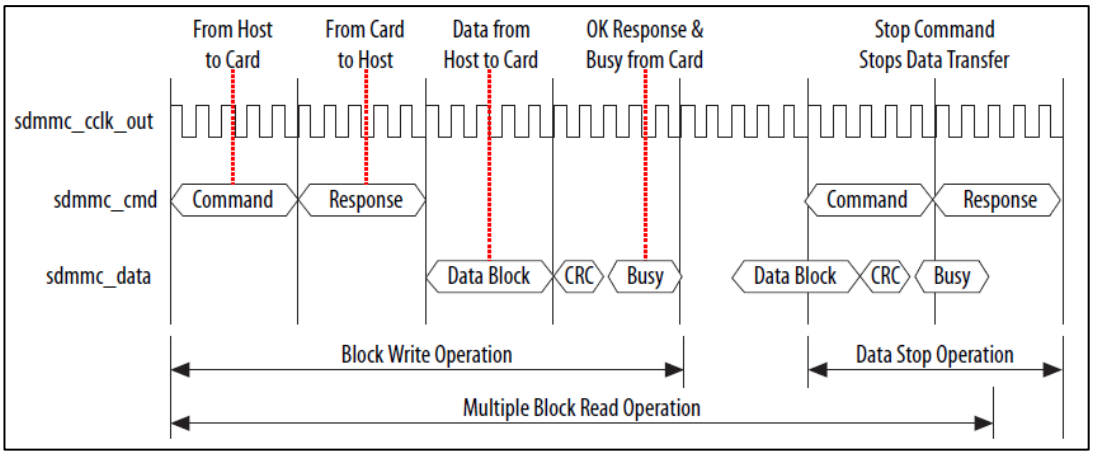

Figure 2.7 Multiple Block Write Operation[3]

Each of the command, response and data has their own format. Command token have a total length of 48bits. Each of the command is starting with a start bit (0) and end with stop bit (1). Before a stop bit it will have CRC bit to protect the transmission so that the error can be detected. CRC size is 7bits wide. The bit after start bit is showing that the command is send from host to the card (1). Figure 2.8 indicates the command format.

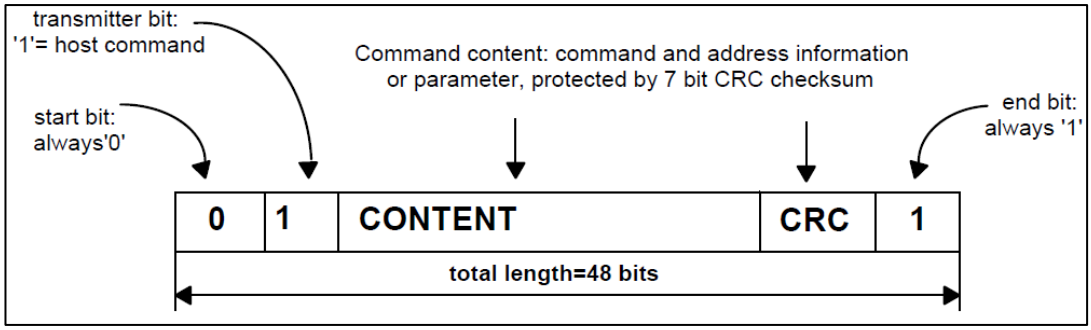

Figure 2.8 Command Format[2, 3]

A response token have two lengths, either 48bits or 136bits. The length of the response is depending on the content of the response. Like command format, response format also start with a start bit (0) and end with a stop bit (1). After start

bit, there is one bit (0) to indicate the response is transmitted from card to host as shown in Figure 2.9.

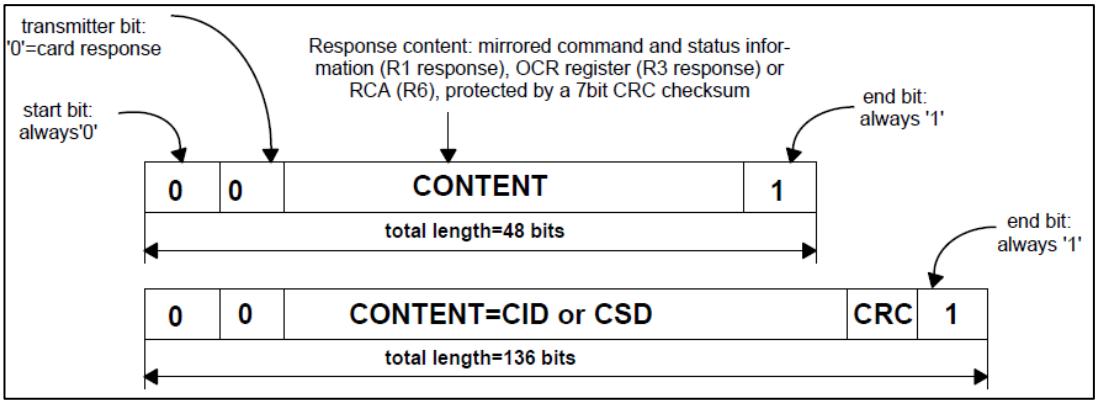

Figure 2.9 Response Format[2, 3]

Data format also similar as command and response format. It will start with the start bit (0) and end with stop bit (1) as well as each of the data will followed by CRC bit. The data transmit in a byte. There are 3 modes for SD/MMC data bus width which are 1bit mode, 4 bit mode and 8 bit mode. The 4bit mode will transmit 4 bits data at a time on data lines and 8bit mode will send 8bit data at a time. The 4 bit mode is faster that 1bit mode and 8bit mode are faster than 4bit and 1 bit mode.

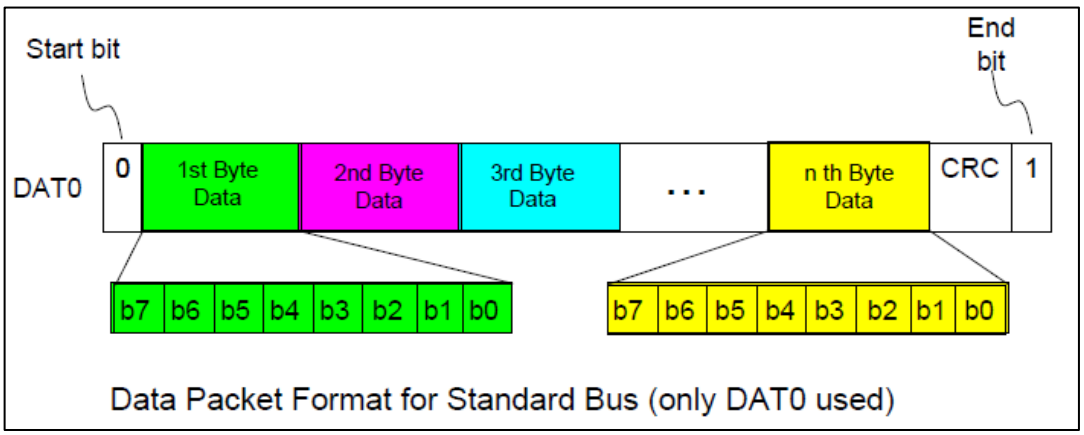

Figure 2.10 1 bit mode bus width format[3]

Figure 2.10 shows 1 bit mode bus width format. 1 bit mode will only use one data line which is data zero line (DAT0). The first byte to transmit is the least significant byte data and the last byte to transmit is the most significant byte data. Each byte will have 8 bits. The most significant bit is bit 7 and the least significant bit is bit 0. On each of the byte, the most significant bit will transmit first.

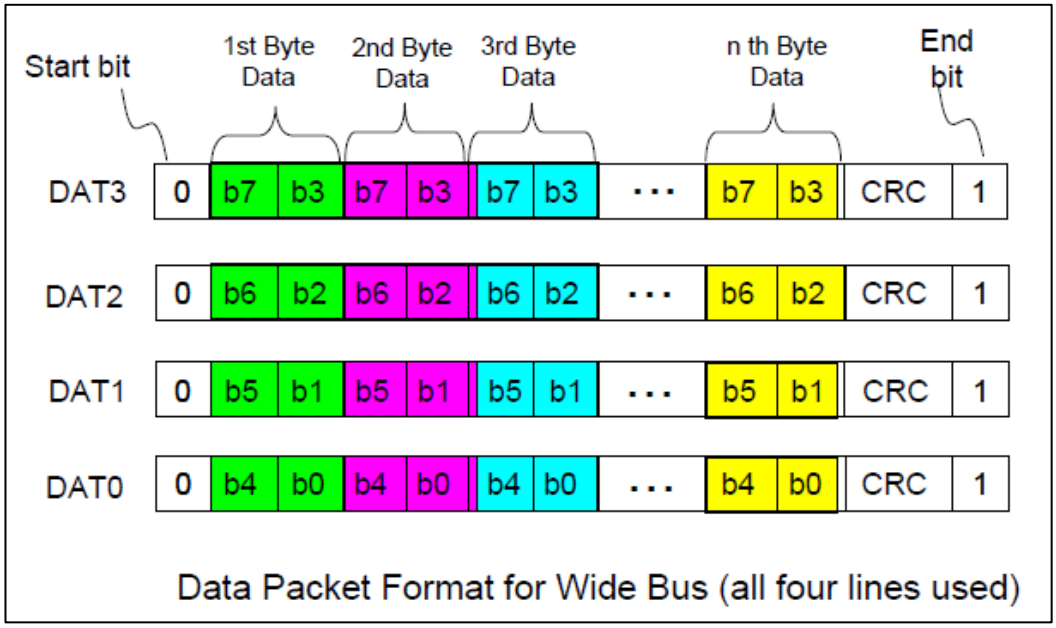

Figure 2.11 4 bit mode bus width format[3]

Figure 2.11 explains on 4 bit mode bus width formats. In 4 bit mode, four data line is use for data transfer. Same as 1bit mode, the least significant byte will transmit first. In 4 bit data width mode, a byte of data is divided by two to lower 4 bit and upper 4 bit. Each of byte will transmit the upper 4bit first followed by the lower 4 bit.

2.3 SD/MMC Card Overview

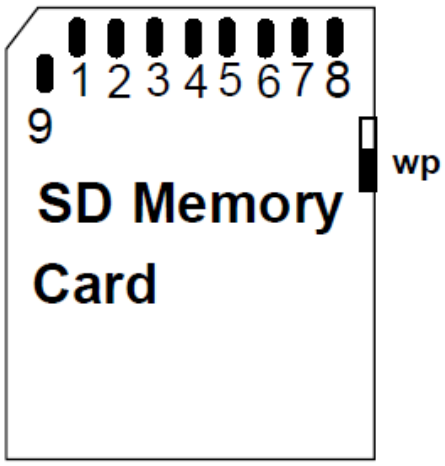

Figure 2.12 SD Card Top View[4]

Figure 2.12 shows the standard size of SD card and its interface. For a standard size card there are 9 pins for interface (clock, command, 4xData lines and 3xPower lines). The SD card interface was designed to operate in a low voltage range from 2.7V to 3.6 volt. But normal voltage used for SD card is 3.3 volt. Table 2.2 indicates the SD card pin description. All the pins are connected to the SD controller pin. The sdmmc_cclk_out from controller is the clock output connected to the card CLK pin number 5. Once the card and controller is power up, the clock signal is supplied from controller to the card.

Since the command, response and data format needs start bit with '0', ideally the SD/MMC bus must be in always high '1'. To have such connection, install a pull up resistor between every CMD and DAT pins. Figure 2.13 shows the SD bus connection diagram. R_{DAT} and R_{CMD} is the pull up resistor to maintain the bus in high state.

Table 2.2 SD card pin assignment

Pin Number	Name	Type	Description
1	CD/DAT3	I/O	Card Detect/ Data Line [Bit 3]
2	CMD	I/O	Command/ Response
3	VSS1	S	Supply voltage ground
4	VDD	S	Supply voltage
5	CLK	I	Clock
6	VSS2	S	Supply voltage ground
7	DAT0	I/O	Data Line [Bit 0]
8	DAT1	I/O	Data Line [Bit 1]
9	DAT2	I/O	Data Line [Bit 2]

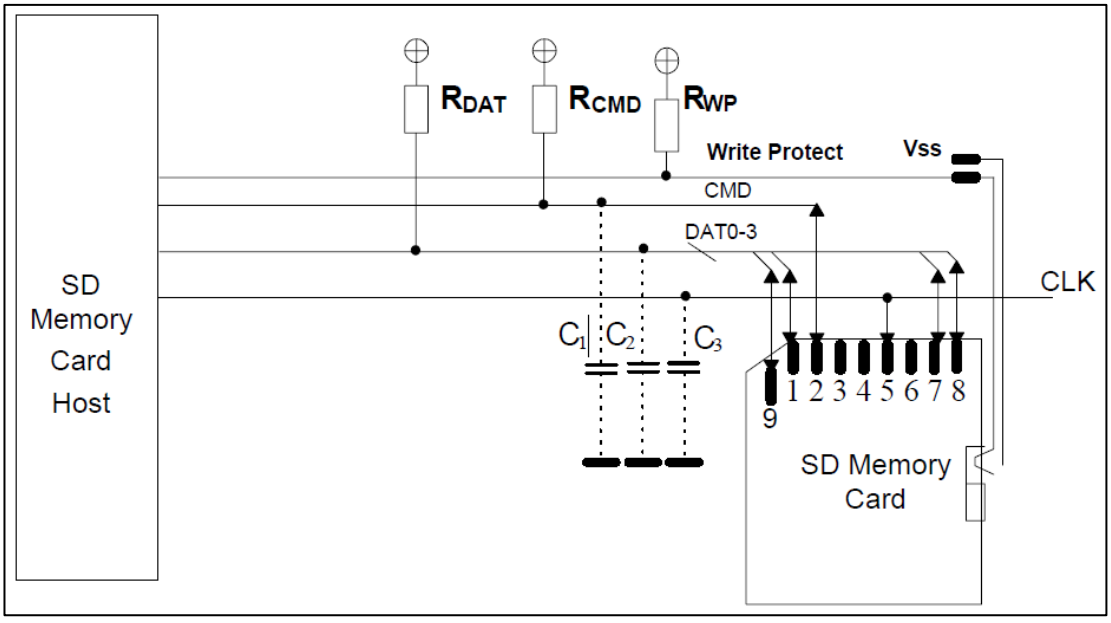

Figure 2.13 SD bus connection diagram[4]

2.4 Post-Silicon Validation

In post-silicon validation, the chips are tested in actual system environments with a characterization board and software to detect and fix design bugs. Designs bug can be classified in two categories which are[5]:

1) Logic bugs: This is caused by design errors and includes incorrect hardware implementation or incorrect interactions between the hardware and system software.
2) Electrical bugs: This is caused by interaction between a design and its electrical state. As an example signal integrity, thermal effects and process variations. Electrical bugs normally are under specific operating conditions such as voltage, temperature and frequency[6].

Post-silicon is needed because pre-silicon verification alone is insufficient. Traditional pre-silicon is too slow, and do not have capability in detecting the electrical bugs that appear after the chips are manufactured[6]. The costs of post-silicon validation are rising from time to time since SoCs is complex and contains variety of components that includes processors, memories and peripherals. Therefore the complexity aggravated the post-validation challenge[7-9]. Post-silicon validation involves three activities as below[10].

1) Bug detection by applying a proper stimuli.
2) Identified the root cause of the bugs.
3) Fix the bugs using software, circuit editing or silicon respin.

In post-silicon validation process, the effort is to identify the root cause of the bugs from the observation on the failure dominates the cost of the process. Furthermore, post-silicon validation is challenging and difficult because most of the existing techniques to identify the root cause of the bugs rely on:

1) System-level simulation to obtain the expected system response.

2) System-level failure reproduction, which involves returning the system to an error-free state and rerunning the system with the exact input stimuli. Example, validation test instructions, validation test inputs, operating conditions such as voltage, temperature, frequency and interrupt to reproduce the failure.

However, the problem with this system-level failure reproduction, the simulation is in order of magnitude slower than actual silicon and failure reproduction is difficult due to Heisenbug effects, asynchronous I/O and multiple clock domains. In post-silicon validation, there are two types of validation which are functional validation and timing validation. This is discussed in the next subtopic.

2.5 Functional Validation Overview

Validation on every design is important to make sure the design can be sell and used. Functional validation can be done in pre-silicon and post-silicon. In pre-silicon validation, the design that is sure to be working can be used as a golden reference for implementation in a real device. To make sure the specified design is correct and working as expected, the specification validation is extremely important. The specification validation is important to ensure the design is following all the requirements and the specifications and ensure the design is correct.

There are two key distributers to the SoC silicon failures (re-spin) which are specification errors and implementation errors. As in [11] it is expected that 8 percent of the designs with re-spin as the results from functional flaws had implementation errors. Moreover, 47 percent of the designs with re-spin as the results from functional flaws also had incorrect or incomplete specifications. Therefore, this is the reason why it is necessary to validate specifications before validating the implementation (post-silicon validation).

2.5.1 Functional Validation Methods

The most widely used method for SoC validation is simulation. Compared to random testing methods which use billions of random and pseudo-random tests in the traditional design flow, directed tests are very promising in reducing the overall validation effort since a significantly smaller number of directed tests can achieve the same coverage goal[12]. However, a biggest problem in directed test generation approach is that the approach mostly performed by human intervention. For example, hand-written tests entail laborious and the design under verification that needs a deep knowledge from verification engineers. Due to the manual development, it is infeasible to generate all directed tests to achieve a comprehensive coverage goal in a short time. Automatic directed test generation based on a comprehensive functional coverage metric is an alternative to address this problem.

One of the general used formal methods for automated test generation to validate software or hardware designs is model checking[13, 14]. In the context of test generation, a design specification is described using a formal model. The required functional scenarios are described in the form of temporal logic formulas. When checking a false property using a model checker, one counter example is reported to falsify the property. Because this counter example is a sequence of variable assignments, it can be used as a directed test to validate the functional scenario of the specification. However, model checking based techniques do not scale well for large designs due to the “state space explosion”.

Even though the simulation based methods are fast but it cannot guarantee the processes of functional coverage. In addition, model checking methods can automatically generate directed tests but cannot deal with large designs. As for now, most of the SoC validation ways use a hybrid method which is incorporates both techniques. The hybrid method will first perform the random simulation to get as much functional coverage as possible. Then the uncovered functional scenarios and corner cases are activated using the directed tests.