DEVELOPMENT OF SIMPLE I3C CONTROLLER BUS FUNCTIONAL MODELING FOR INTERNAL TEST CARD VERIFICATION

By

PUAN CHIA KIAN

A Dissertation submitted for partial fulfilment of the requirement for the degree of Master of Science (Microelectronic Engineering)

AUGUST 2016

ACKNOWLEDGEMENT

I would like to spend a little time to acknowledge everyone who had provided support and contribution to this research. Special thanks to my supervisor, Assoc. Prof. Dr. Asrulnizam Bin Abd Manaf for guiding me with his expertise and also to Mr. Pang Chun Hung from Intel Corporation, for giving his outmost support in my study, hence getting this research done in time. Further, I would like to thank Mr. Ban Poh Wei from Altera Corporation in giving his guidance regarding FPGA knowledge. Lastly, my appreciation goes to Universiti Sains Malaysia (USM) and Intel Corporation for allowing me to use their facilities throughout the research.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATION

ACK	Acknowledge
ADC	Analog to Digital Converter
ASMD	Algorithmic State Machine and Datapath
BFM	Bus Functional Model
CCC	Control Command Code
CU	Control Unit
DU	Datapath Unit
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
GND	Ground
GPIO	General Purposes Input Output
HDL	Hardware Description Language
HDR	High Data Rate
HDR-DDR	High Double Data Rate Mode
IBI	In-band Interrupt
I^2C	Inter-Integrated Circuit

I3C	Improved Inter-Integrated Circuit
ISS	Instruction Set Simulator
LSB	Least Significant Bit
MHz	Mega Hertz
MIPI	Mobile Industry Processor Interface
MSB	Most Significant Bit
MUX	Multiplexer
NACK	Not Acknowledge
PLL	Phase Lock Loop
RnW	Read and Write
RTL	Register Transfer Level
SCL	Serial Clock
SDA	Serial Data
SDR	Single Data Rate
SPI	Serial Peripheral Interface
SUT	System Under Test
T	Transition Bit
TSL	Ternary Symbol Legacy

TSP	Ternary Symbol for Pure Bus (no I^{2}C Devices)
UART	Universal Asynchronous Receiver/Transmitter
VCO	Voltage Controlled Oscillator

PEMBANGUNAN PENGAWAL PERMODALAN KEFUNGSIAN BUS RINGKAS I3C UNTUK PENGESAHAN KAD UJIAN DALAMAN

ABSTRAK

Di ambang dunia teknologi digital yang kian maju dan membangun, ciri sensor menjadi semakin canggih dan penggunaannya pula makin meluas dari produk seperti telefon bimbit hinggalah produk segmen baru seperti ‘Internet of Thing’ (IoT). Oleh sebab itu, terdapat keperluan dalam menghasilkan antara muka yang bukan sahaja meningkatkan kelajuan, malah mampu memindahkan data daripada sensor dengan jalur lebar yang lebih tinggi kepada sistem komputer. Oleh itu, ‘Improved Inter-Integrated Circuit’ (I3C) diperkenalkan oleh Mobile Industry Processor Interface Alliance (MIPI) sebagai standard antara muka baru yang mampu memenuhi prestasi diperlukan sensor generasi baharu selain berupaya beroperasi menggunakan kuasa rendah, satu ciri yang penting untuk sensor sebagaimana terdapat di dalam pendahulunya, I^2C. Sebagai sebahagian daripada kesediaan untuk ujian pengesahan pengawal I3C, satu kad ujian dalaman Intel akan digunakan dalam aktiviti pengesahan tersebut, menggantikan sensor atau peranti sebenar. Dengan ketiadaan pengawal I3C sebenar di dalam pasaran, satu penyelesaian sementara diperlukan supaya kad ujian tersebut boleh disahkan fungsinya, sekaligus dapat sedia digunakan apabila pengawal I3C sebenar boleh didapati dalam pasaran kelak. Oleh yang demikian, I3C BFM telah dicadangkan untuk mengantikan pengawal I3C sebenar dalam pengisuan kitaran I3C secara ringkas. Berdasarkan FPGA, I3C BFM berupaya melaksanakan aliran Enter Dynamic Address Assignment (ENTDAA). Aliran ini memerlukan I3C BFM dalam penjanaan kitaran protokol I3C Broadcast CCC dan Modal Broadcast CCC.

Setelah aliran tersebut selesai, kad ujian I3C akan mempunyai alamat dinamik, sekaligus bersedia untuk menghantar atau menerima transaksi I3C.

DEVELOPENT OF SIMPLE I3C CONTROLLER BUS FUNCTIONAL MODEL FOR INTERNAL TEST CARD VERIFICATION

ABSTRACT

Sensors usage are becoming broader as the technology gets more advance in recent years, with its features becoming more sophisticated and can be found in wide range of products from cellphone to Internet of Thing (IoT). As a result, demand for new interface with more speed and bandwidth to transfer more data from sensor to the system for processing and existing communication will soon be surplus to the requirement to be able to transfer the large data fast enough to be processed. Hence, Improved Inter-Integrated Circuit (I3C) is introduced by Mobile Industry Processor Interface Alliance (MIPI) as new interface standard that able to deliver the required performance compared to its predecessor I^2C, at the same time maintain the low power operation mode that's essential to the sensor subsystem. As part of a new interface readiness, an Intel in-house I3C Test Card will be used in validation activity against the Controller in place of actual sensors or devices. While there is not any I3C controller based system available for testing yet, there is a need to have a temporary solution to replicate it in to enable the said I3C Test Card in order to have it ready once a system with I3C controller is available for validation. Thus, an I3C BFM is proposed to enable simple I3C protocol cycle transfers between it and I3C Test Card to achieve just that. I3C BFM will be FPGA based and capable to perform Enter Dynamic Address Assignment (ENTDAA) flow. The flow will require BFM to issue I3C Broadcast CCC

and Modal Broadcast CCC protocol cycles. I3C Test Card will be assigned with a dynamic address upon completion of the flow and it is ready to send or receive I3C transaction as a result.

CHAPTER 1

INTRODUCTION

1.0 Introduction

Today's technology is getting more advanced, with the world getting "digitized" in all sorts of area. From a simple machine that able to make a simple phone call to a machine that can even replace a personal computer, the demand of multiple types of sensors in a gadget and its usage had exponentially increased. Not only that, sensor usage also getting wider with lots of essential as well as nice to have sensors built into even the simplest technology gadget to attract buyers and create a new technology waves. As a result, a higher demand of sensor communication interface in terms of bandwidth and speed become increasingly important. Among the widely used sensor communication interface are Inter-Integrated Circuit (I^2C), Serial Peripheral Interface (SPI) and Universal Asynchronous Receiver/Transmitter (UART) which mainly capable of doing transmission in relatively low power and having just enough speed to operate the sensors.

In order to cope with the increased demand of sensors, a communication interface with more capabilities and higher speed but still require only little power to operate is needed to handle the heavy traffic of data. I3C is the latest bus communication standard being introduced and is intended to improve upon the features of I^2C interface [1], preserving the backward compatibility. It provides a fast, low cost, low power two-wire digital interface for sensors [2].

This thesis will focus on creating a basic I3C Bus Functional Model (BFM) that is capable of generating and accepting I3C protocol cycles. As I3C is still relatively new and actual controller will not be available anytime soon, there’s a need to have the I3C BFM to help in test card development and verification. Test card is a Field-Programmable Gate Array (FPGA) based hardware to be developed and used by internal team only, often served as replacement for actual device. In order for I3C test card to be ready for I3C controller validation, it needs to go through validation itself to ensure that the test card is behaving as defined in I3C device specification set by the industry.

Other than that, this thesis will be discussing on the I3C BFM development in terms of plan, concept and design, and tools used in terms of hardware and software. The developed BFM will be strictly for internal usage and for test card verification only. It does not intend to become a fully functional I3C controller replica in terms of power and speed.

1.1 Problem Statement

I3C, a newly improved interface protocol, is designed to be faster and loaded with more features in supporting the advancing sensors communication while still maintaining backward compatibility to its predecessor, I2C protocol. New features such as Broadcast Command, In-Band Interrupt and Peer-to-Peer slave are introduced, as such existing I2C test tool and controller will not be able to be re-used to replicate I3C features, let alone outperformed it.

As of now (when this thesis has been written), there is not any I3C product available in the open market, be it controller nor devices. That creates complication when there is a need to build an in house test card that act as a device for I3C controller testing when it is available. Since there is not any available product that able to be used in order to generate I3C cycles into the test card, a simple I3C BFM is the best solution. The BFM is able to generate basic I3C command cycles to the test card or any other I3C capable devices and received the response from the devices as well.

By having this I3C BFM, not only I3C test card will get validated, it also will be able to serve as a simple slave device or Secondary Master for further I3C validation usage. By having this BFM, the entire readiness and validation activity able to shift-left to increase confidence on the healthiness of I3C controller and test card.

1.2 Research Objectives

The objectives of this research are as follows:

I. To develop I3C BFM capable of transmitting and receiving I3C protocol cycles

II. To analyze in house FPGA hardware and custom software for BFM communication interface

III. To simulate and characterize complete I3C ENTDAA flow with I3C BFM and in-house I3C Test Card

1.3 Research Contribution

This research contributes in getting I3C devices manufacturer early development tool for their product development before actual controller is available on the market. The software used to write the BFM will be written in Register Transfer Level (RTL) language. The FPGA used will be from Altera Corporation (now Intel). The software used to program the FPGA will be Quartus Prime, also by Altera Corporation.

1.4 Research Scopes

This thesis will be focusing on using internal developed software and external sourced FPGA card to create the I3C BFM, while I3C test card are sourced in-house and it is a top secret hardware within the company. The I3C BFM only focused on transmitting and receiving I3C cycles, and will not focus on the strict timing and speed performance. Also, the BFM is not for any large data transfer and long hour operation. It is safe to assume that the BFM is representing a basic I3C controller which operate in ideal condition in terms of traffic and connectivity to the device.

1.5 Thesis Outline

This thesis was organized into five chapters as follows:

In Chapter 2, there will be comprehensive literature reviews. The reviews consists of area of research field related to this thesis such as sensors, overview of I^{2}C and I3C as well as FPGA technology. A thorough review is to be done in order to ensure there's no similar implementation being done.

In Chapter 3, methodology of the research will be spelled out. It is split into four development stages and each of the stages will be discussed in this chapter. Two available I3C BFM implementation choices will be discussed in the first stage. Each of the options will be studied in details including its advantages and disadvantages, ultimately contributed on picking up the best option among the two. In second stage, the BFM development planning will be outlined in details, together with the timelines. This will be depending on the preferred solution option outcome in the first stage. In third stage, I3C BFM High Level Architecture Design and I3C BFM Block Diagram will be presented to give the overall view on the hardware connectivity as well as outlook of I3C BFM System Design. Finally, in the fifth stage, I3C BFM Function Outlines and ASMD Chart are discussed here. The lists of all I3C BFM functions implementation and description will be shared, with ASMD Chart to show the code flow.

More details on FPGA hardware and software will be discussed in Chapter 3 as well. The FPGA card of choice and its software, Quartus Prime are going to be introduced there. The overall features available on the FPGA cards and the relevant features going to be used for I3C BFM will be pointed out. The steps on using Quartus Prime to compile I3C BFM design RTL and program the FPGA card will be presented as well. Also, there will be discussion on the simulation tools named Modelsim that’s been used to do verify I3C BFM design RTL in simulation level.

In Chapter 4, the I3C BFM RTL codes module will be presented here, corresponding to the planned function outlines. The result of overall I3C BFM supported features will also be presented in simulation waveform. The features and simulations will be break into few parts in a cycle transaction as the entire simulation waveform will be too long to be put together. An in-depth analysis will be done to ensure the BFM generating correct I3C cycles as defined in MIPI I3C Specification documentation.

Finally, the Conclusion and Future Improvements will be in Chapter 5 which will wrap up the entire research progress and results. This chapter will also spell out future improvements that can be implemented into the I3C BFM.

CHAPTER 2

LITERATURE REVIEW

2.5 Introduction

Prior to design, there are few major elements that needs to be studied in details in order to create the I3C BFM with correct behavior and compliant to specification. Those major elements are sensor, I^2C, I3C, HDL and FPGA. This chapter will describe in details the properties of each of the elements.

2.1 Overview of Sensors

A sensor is often defined as a “device that receives and responds to a signal or stimulus.”[3]. In order to narrow down the scope to be more focused rather than a broad definition, it can be also be defined as a device that breaks into two main functions, that is, a) to detect certain conditions or scenarios and b) to output in form of electrical signals when the conditions or scenarios make occurrences. That been said, a sensor is actually act as translator of a generally nonelectrical value into an electrical signal value in which can be amplified, modified, and channeled by electronic devices. The sensor’s output signal can be represented in the form of current, charge or voltage. These may be further described in terms of digital code (binary 1 or 0), polarity (positive or negative), frequency, phase or amplitude. This set of characteristics is called the output signal format. Therefore, a sensor has input properties (of any kind) and electrical output properties.

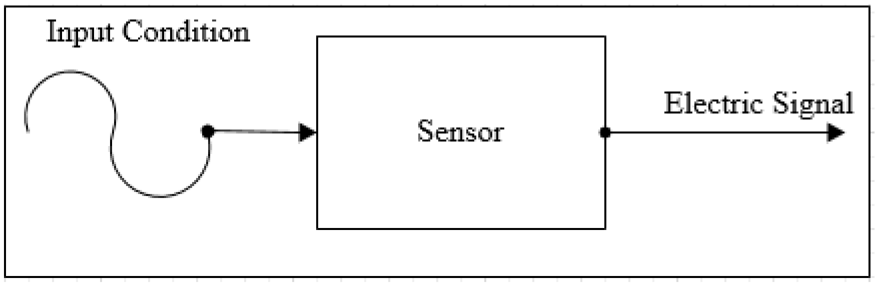

Figure 2.1 Overview of Sensor

The term sensor should be distinguished from transducer. The latter is a converter of any one type of energy into another, whereas the former converts any type of energy into electrical energy. Transducers may be used as actuators in various systems. An actuator may be described as an opposite to a sensor; it converts electrical signal into generally nonelectrical energy. For example, an electric motor is an actuator; it converts electric energy into mechanical action. Transducers may be parts of complex sensors. For example, a chemical sensor may have a part, which converts the energy of a chemical reaction into heat (transducer) and another part, a thermopile, which converts heat into an electrical signal. The combination of the two makes a chemical sensor, a device which produces electrical signal in response to a chemical reagent. Note that in the above example a chemical sensor is a complex sensor; it is comprised of a nonelectrical transducer and a simple (direct) sensor converting heat to electricity.

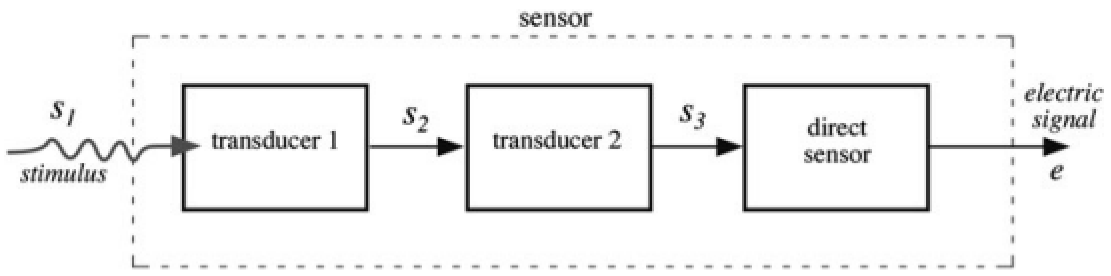

Figure 2.2 Combination of Sensor and Transducer to Produce Electric Signal [4]

Generally there are two types of sensors; direct and complex. A direct sensor converts a stimulus into an electrical signal or modifies an electrical signal by using an appropriate physical effect, whereas a complex sensor in addition needs one or more transducers of energy before a direct sensor can be employed to generate an electrical output. A sensor does not function by itself; it is always a part of a larger system that may incorporate many other detectors, signal conditioners, signal processors, memory devices, data recorders, and actuators. The sensor's place in a device is either intrinsic or extrinsic. It may be positioned at the input of a device to perceive the outside effects and to signal the system about variations in the outside stimuli. Also, it may be an internal part of a device that monitors the devices' own state to cause the appropriate performance. A sensor is always a part of some kind of a data acquisition system. Often, such a system may be a part of a larger control system that includes various feedback mechanisms.

To illustrate the place of sensors in a larger system, Figure 2.3 shows a block diagram of a data acquisition and control device. An object can be anything: a car, space ship, animal or human, liquid, or gas. Any material object may become a subject of some kind of a measurement. Data are collected from an object by a number of sensors. Some of them (2, 3, and 4) are positioned directly on or inside the object. Sensor 1 perceives the object without a physical contact and, therefore, is called a noncontact sensor. Examples of such a sensor are a radiation detector and a TV camera. Even it contains the word “noncontact,” energy transfer always occurs between any sensor and an object. Sensor 5 serves a different purpose. It monitors internal conditions of a data acquisition system itself. Some sensors (1 and 3) cannot be directly connected to standard electronic circuits because of inappropriate output signal formats. The sensors require the use of interface devices (signal conditioners). Sensors 1, 2, 3, and 5 are passive. The sensors generate electric signals without energy consumption from the electronic circuits. Sensor 4 is active. It requires an operating signal, which is provided by an excitation circuit. This signal is modified by the sensor in accordance with the converted information.

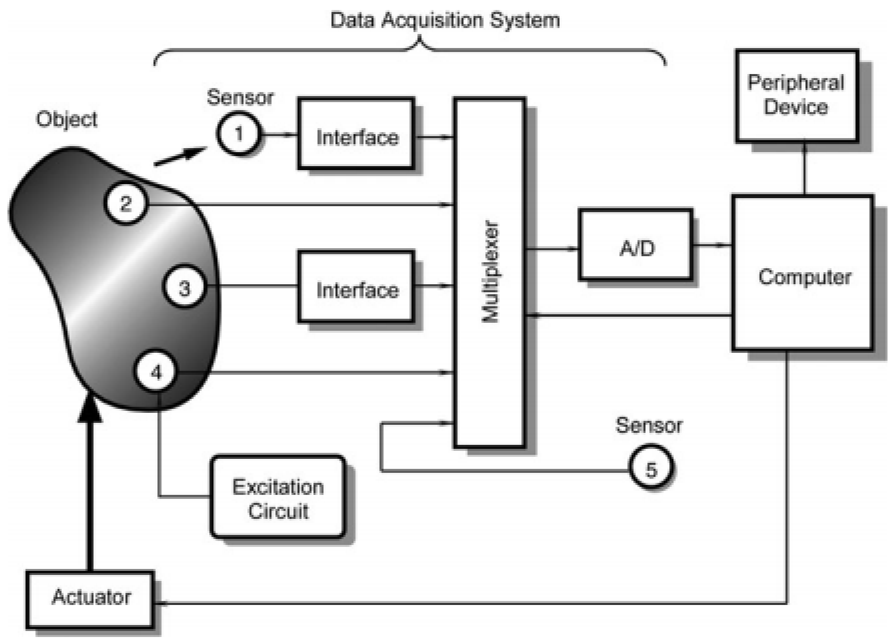

Figure 2.3 Overview of Sensor Subsystem [3]

An example of an active sensor is a thermistor, which is a temperature-sensitive resistor. It needs a constant current source, which is an excitation circuit. Depending on the complexity of the system, the total number of sensors may vary from as little as one (a home thermostat) to many thousands (a space shuttle). Electrical signals from the sensors are fed into a multiplexer (MUX), which is a switch or a gate. Its function is to connect sensors one at a time to an analog-to-digital converter (A/D or ADC) if a sensor produces an analog signal, or directly to a computer if a sensor produces signals in a digital format. The computer controls a multiplexer and an A/D converter for the appropriate timing. Also, it may send control signals to the actuator, which acts on the object. Examples of the actuators are an electric motor, a solenoid, a relay, and a pneumatic valve. The system contains some peripheral devices (for instance, a data recorder, a display, an alarm, etc.) and a number of components that are not shown in the block diagram.

2.2 Overview of I^{2}C

From the diagram, it shows that sensors are connected to interfaces, with I^{2}C among the commonly used ones. I^{2}C (also formerly known as IIC) bus was originated from 1982 and its original purpose was to provide a convenient way to connect a Central Processing Unit (CPU) to peripheral chips in a television set. Philips labs in Eindhoven, Netherlands invented the I^{2}C protocol that only requires two wires for connecting all the peripherals to a microcontroller in order to save additional “glue logic” and pins that are required to do CPU to peripheral decoding. The original specification defined a bus speed of 100 kb/s. The specification was reviewed several times, notably introducing the 400 kb/s speed in 1995 and, since 1998, 3.4 Mb/s for even faster peripherals. As of October 1, 2006, no licensing fees are required to implement the I^{2}C protocol, making official a type of “fair open policy” that Philips Semiconductor has always demonstrated with I^{2}C [4].

I^{2}C is a multi-master protocol that uses two signal lines. The two I^{2}C signals are called serial data (SDA) and serial clock (SCL). There is no need of chip select (slave select) or arbitration logic. Virtually any number of slaves and any number of masters can be connected onto these two signal lines and communicate between each other using a protocol that defines few of essentials data structure to be sent through SDA such as 7-bit slave addressing, 8-bit data and a few of control bit to control START, STOP, Write/Read as well as ACK generation and acknowledgement.

As shown in Figure 2.4 below, I^{2}C has a total of 3 physical pin; SCL, SDA and a common groung GND. Both SCL and SDA are bi-directional and connective to a pull-up resistors. At the physical layer, both SCL and SDA lines are open-drain I/Os with pullup resistors. Pulling such a line to ground is decoded as a logic ZERO, whereas releasing the line and letting it float is a logic ONE. Actually, a device on an I^{2}C bus “only drives zeros.”

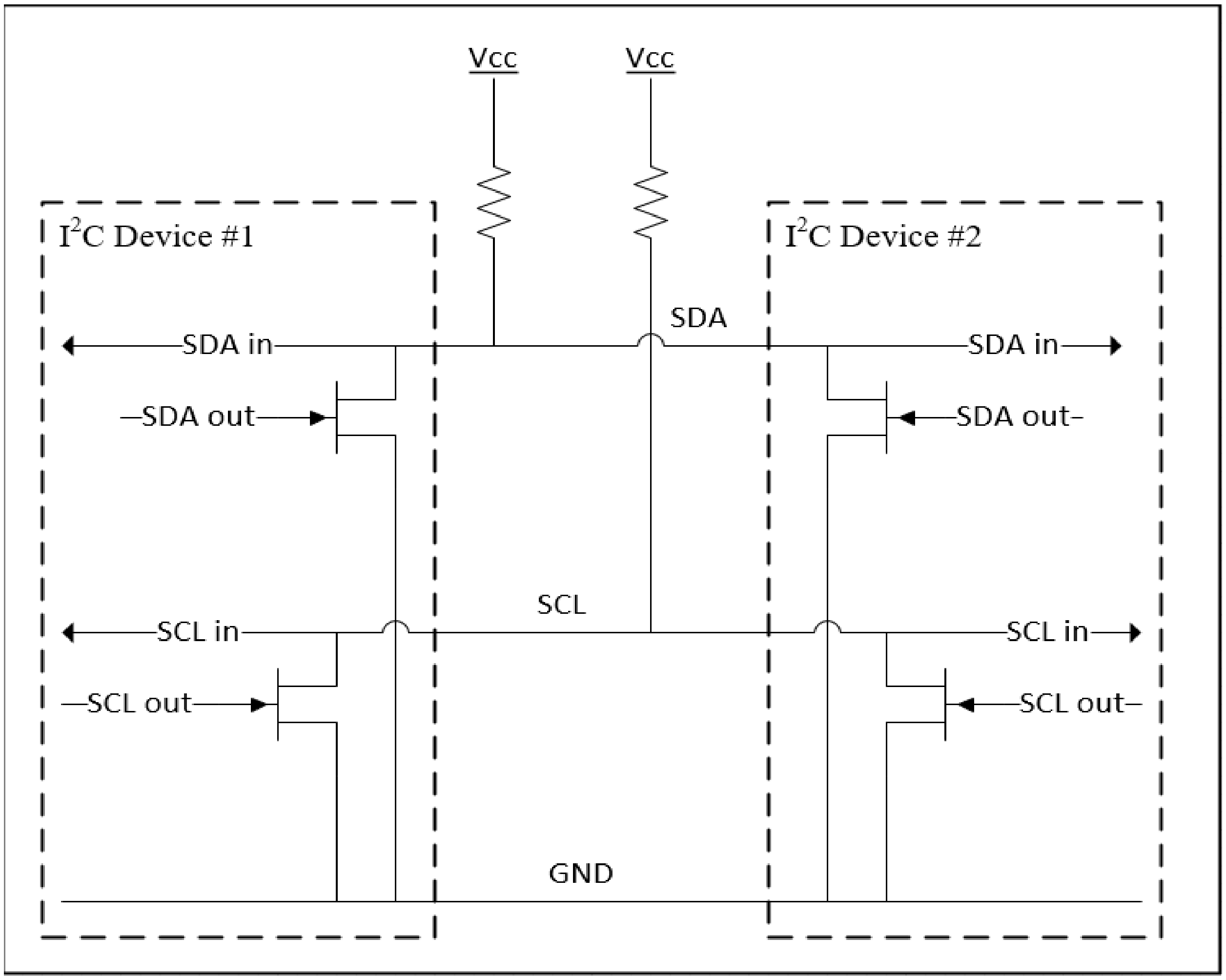

Figure 2.4 High Level I^{2}C Bus Diagram

Here come to where I^{2}C is truly elegant. Associating the physical layer and the protocol described above enables flawless communication between any numbers of devices on just two physical wires. IF two devices are trying to put information into SDA and SCL simultaneously, there is actually no conflict at electrical level because if, for example, one of the drivers tries to write a logic ZERO and the other a logic ONE, then the open-drain and pull-up structure ensures that there will be no short-circuit and the bus will actually see a logic ZERO transiting on the bus. In other words, in any conflict, logic ZERO always "wins."

A device or controller that initiates data transfer first is regarded as Master. At the same time, all the over devices that present on the bus will become slaves devices. The master will issue a START condition with the pattern as shown in Figure 2.5. This acts as an "Attention" signal to all of the connected devices. All ICs on the bus will listen to the bus for incoming data.

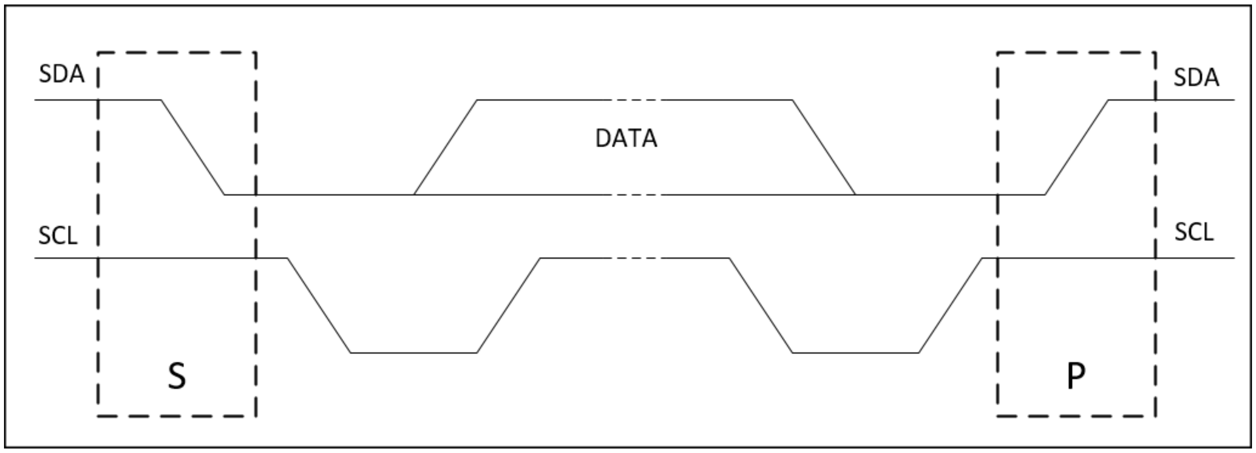

Figure 2.5 START and STOP Protocol in I^{2}C

Then, the master sends the ADDRESS of the device it wants to access, along with an indication of whether the access is a Read or Write operation (Write in our example). Having received the address, all Ics will compare it with their own address. If it does not match, they simply wait until the bus is released by the stop condition (see below). If the address matches, however, the chip will produce a response called the ACKNOWLEDGE signal. Once the master receives the ACKNOWLEDGE, it can start transmitting or receiving DATA. In our case, the master will transmit data. When all is done, the master will issue the STOP condition. This is a signal that states that the bus has been released and that the connected Ics may expect another transmission to start at any moment. When a master wants to receive data from a slave, it proceeds the same way, but sets the RD/nWR bit at a logic ONE. Once the slave has acknowledged the address, it starts sending the requested data, byte by byte. After each data byte, it is up to the master to acknowledge the received data.

Table 2.1 Overview of I^2C Bus Protocol for Data Transfer

START	Slave Address	R/W	ACK	DATA	ACK	DATA	ACK	STOP
1 bit	7 bits	1 bit	1 bit	8 bits	1 bit	8 bits	1 bit	1 bit

In addition, the I^2C protocol also helps in dealing with communication problems. Any device present on the I^2C listens to it permanently. Potential masters on the I^2C detecting a START condition will wait until a STOP is detected to attempt a new bus access. Slaves on the I^2C bus will decode the device address that follows the START condition and check if it matches theirs. All the slaves that are not addressed will wait until a STOP condition is issued before listening again to the bus. Similarly, because the I^2C protocol foresees active-low acknowledge bit after each byte, the master/slave couple is able to detect their counterpart’s presence. Ultimately, if anything else goes wrong, this would mean that the device “talking on the bus” (master or slave) would know it by simply comparing what it sends with what is seen on the bus. If a difference is detected, a STOP condition must be issued, which releases the bus I^2C also has some advanced features, like extended bus addressing, clock stretching, and the very specific 3.4 Mb/s high speed mode.

Table 2.2 below shows reserved addressing codes in I^{2}C protocol. In the table, it's defined that address with "11111xxx" (where x can be 0 or 1) is reserved for future purposes. This is where I3C protocol takes advantage to slit in I3C specific protocol to further expand the features while maintaining backward compatibility towards I^{2}C. Details will be discussed in I3C overview session below.

Table 2.2 Range of I^{2}C addresses and its purposes

Address (X – don't care)	**Purposes**
00000000	General call address. Supported by all devices
00000001	Start byte
0000001X	CBUS Addresses
0000010X	Reserved for different bus formats
0000011X	Reserved for future purpose
00001XXX	High-speed master code
11110XXX	10-bit slave addressing
11111XXX	Reserved for future purposes

As the I^{2}C has been introduced for more than 30 years, it has reach a performance bottleneck where it can barely kept up with ever growing demand of speed and bandwidth required by advanced sensors. Thus, an improvement to support such sensors is needed to achieve the performance desired while maintaining backward compatibility and low operating power.

2.3 I3C Overview

Improved Inter Integrated Circuit, also known as I3C is more like an expansion to the old but ever reliable I^2C protocol. It took advantages of reserved addressing that's available in I^2C as shown in Table 2.2 by using the addressing to call up I3C specific protocol command. In I3C, address 0x7E (corresponding to address of 7'b 1111110) is been used as address that's been sent right after START bit gets generated. If there are slaves or devices that supports I3C, an ACK will be generated, and at the same time I^2C devices will just ignore it, thus retaining backward compatibility towards I^2C. The I3C interface provides major efficiencies in Bus power while providing greater than 10x speed improvements over I^2C [2].

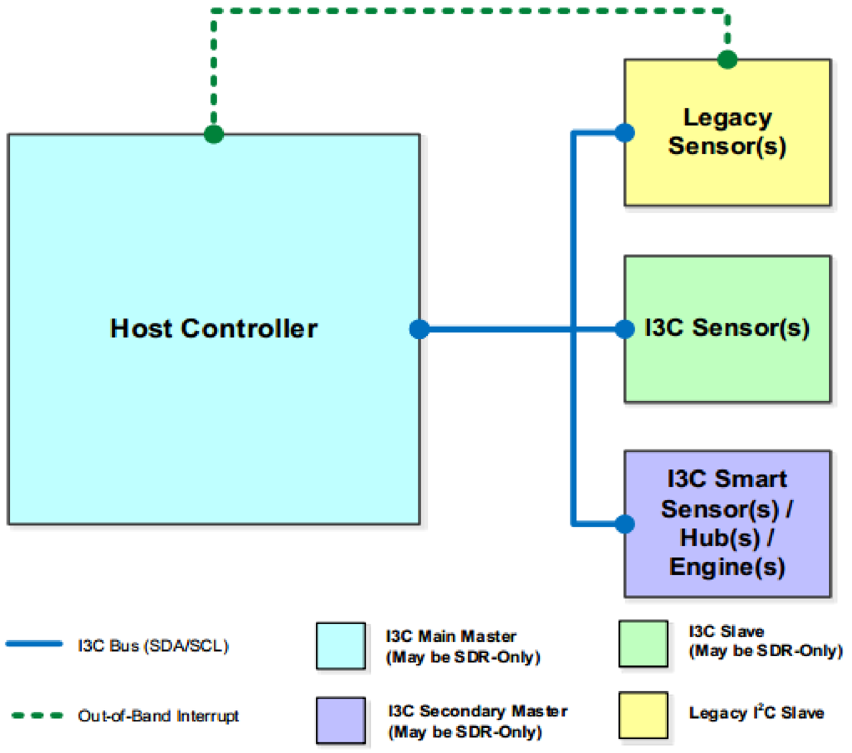
Legacy Sensor(s)
Host Controller
I3C Sensor(s)
I3C Smart Sensor(s) / Hub(s) / Engine(s)
I3C Bus (SDA/SCL)
Out-of-Band Interrupt
I3C Main Master (May be SDR-Only)
I3C Secondary Master (May be SDR-Only)
I3C Slave (May be SDR-Only)
Legacy I²C Slave

Figure 2.6 I3C System Diagram [2]

I3C supports a handful of communication format, all of which sharing a two-wire interface, SCL (Serial CLK) and SDA (SDA), just like I^{2}C. In certain HDR mode in I3C, SCL also will be able to carries Data. An I3C bus supports various Message types:

i. I^{2}C-like Single Data Rate (SDR) Message, with SCL speeds up to 12.5Mhz

ii. Broadcast and Direct Common Command Code (CCC) Messages that allow communicate to one or multiple Slaves in an instance

iii. HDR Mode Messages, which achieve higher data rates with equivalent clock cycle

iv. Legacy I^{2}C Messages to I^{2}C Slaves

v. Slave-initiated request to Master such as In-Band Interrupt or request to be Secondary Master

vi. Peer-to-Peer Messages between I3C Devices

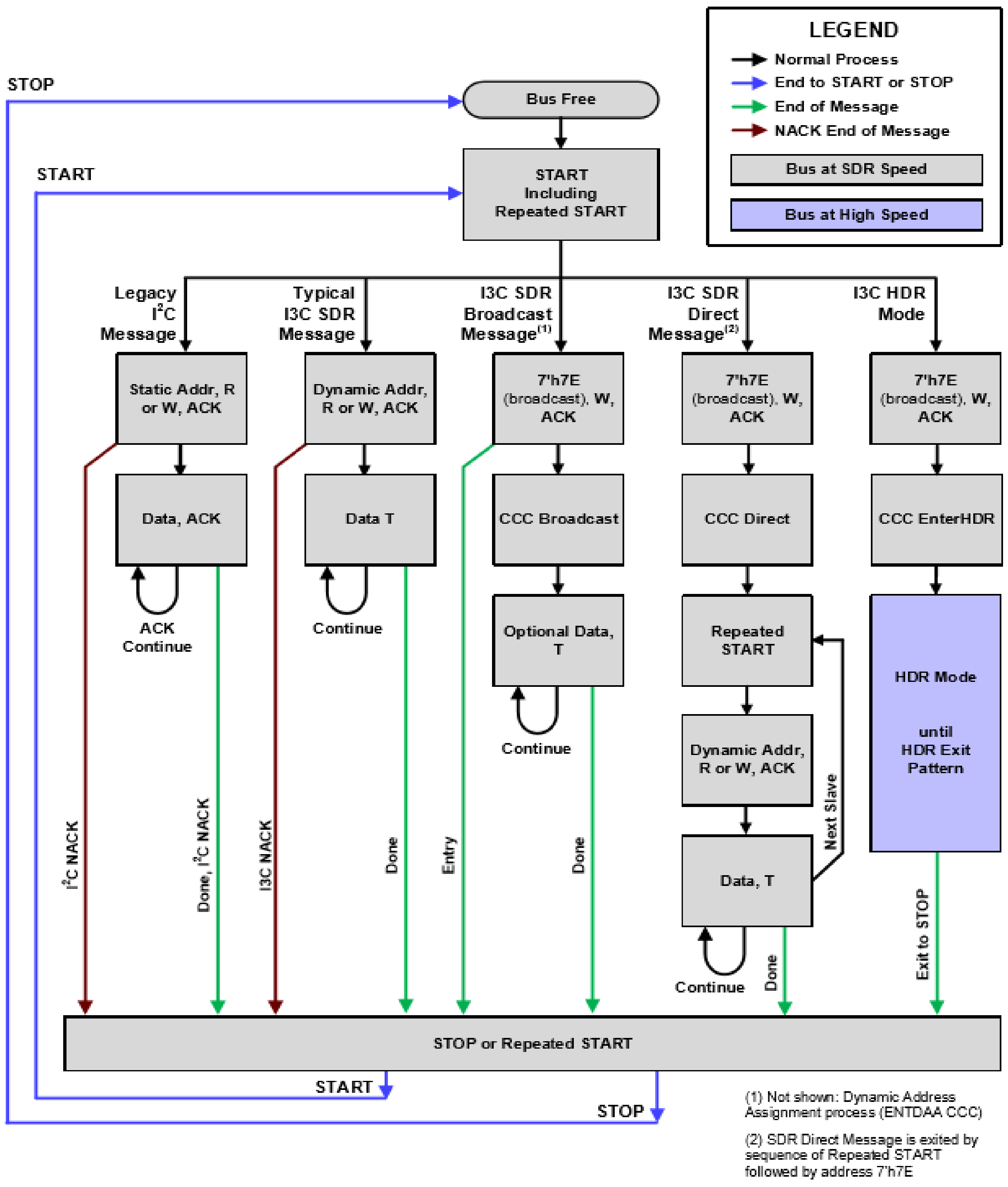

Figure 2.7 I3C Communication Flow [2]

I3C is based on Frame encapsulation approach, whereby a frame includes START, Header, Data Payload and STOP. The Header following a START allows for bus arbitration and the Master use the Header to address Slave Device(s). I3C only allows one Master to have control of the I3C bus at a time. Figure 2.8 below shows I3C bus connecting a Main Master, two Secondary Masters, and both I3C and I^{2}C Slaves at a time.

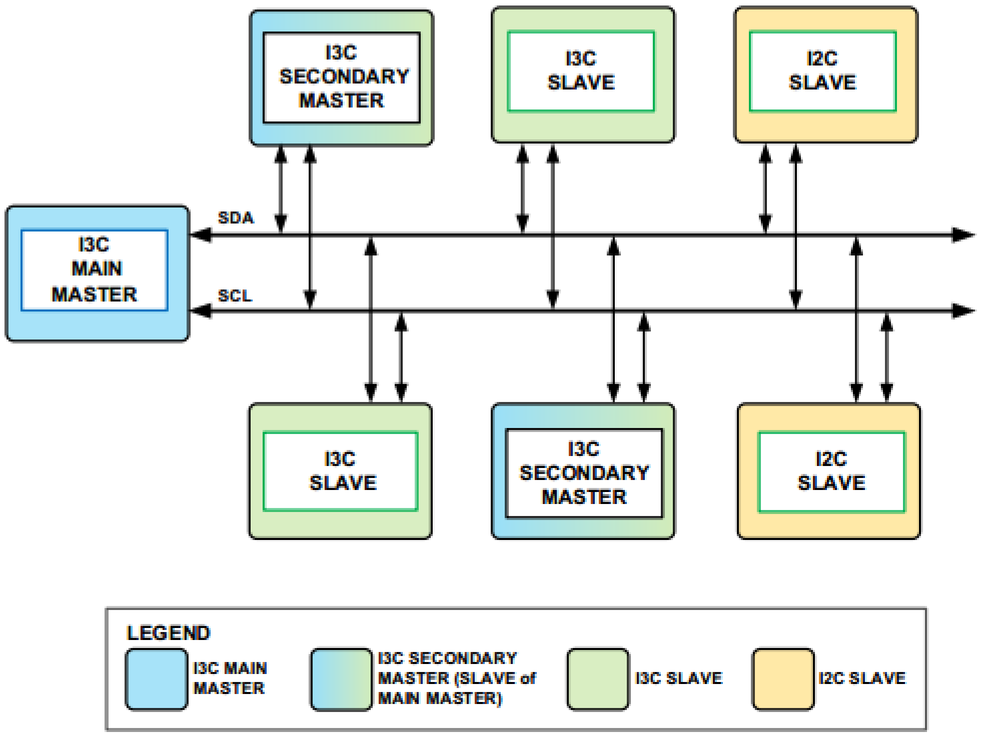

Figure 2.8 I3C Bus with I^{2}C Devices and I3C Devices [2]

Figure 2.9 below illustrates example communication using I3C Single Data Rate (SDR) Mode. It shows the Master reading a byte of data from the Slave at Address 0x2B in SDR Mode. From the Bus-Free condition, the Master issues a START by driving the SDA Line low while keeping the SCL Line high. It then issues the Broadcast Address (0x7E) followed by RnW (0 for Write). Then the Master turns on a pull-up resistor and goes to open drain. All Slaves ACK by pulling the SDA Line low (in the Figure, yellow fill means the Slave is in control of the SDA Line at this time).

The Master then issues a STARTr (repeated start), then the Address of the Slave (0x2B) it wants to read followed by RnW (1 for Read). The Master then turns on a pull-up resistor and goes to open drain, allowing the Slave to acknowledge by pulling the SDA Line low. At this point, the Master continues to toggle the SCL Line and release the SDA Line, allowing the Slave to drive SDA to send one byte of data (0x4A) followed by 'T'. T=1 informs the Master that there is additional data, whereas T=0 signals the end. Here there is additional data, so the Slave drives SDA high until SCL goes high, at which time it releases SDA. The Master has the option of holding SDA high with a weak pull-up, which signals to the Slave that the Master allows another byte to be transmitted, or to pull SDA low (while SCL is high – hence a repeated START), which would signal to the Slave that the Master has terminated the Read and is taking over. SDR Mode is backwards compatible with legacy I^{2}C devices, because the high time of an SCL pulse is always less than 50ns and therefore SCL will always appear to be low because of the I^{2}C 50ns glitch filter.

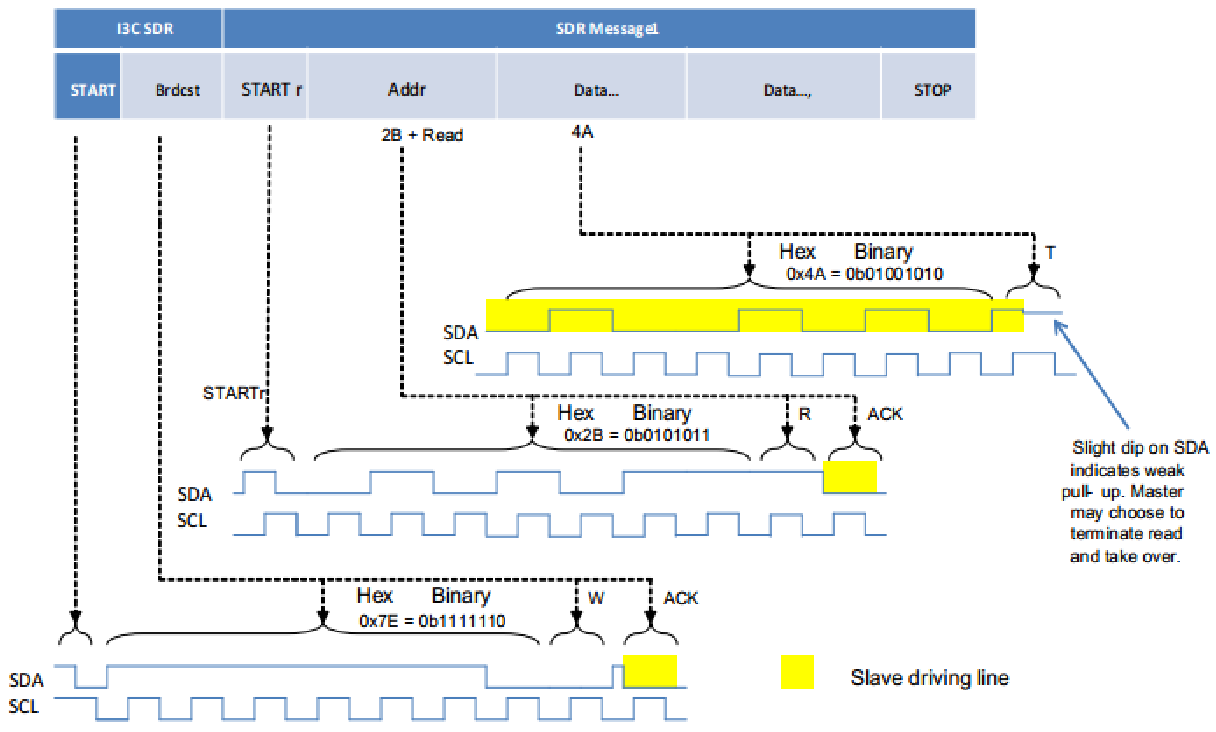

Figure 2.9 I3C SDR Mode Communication Protocol [2]

One of the major improvement introduced in I3C compared to its predecessor is the speed. While I^2C High Speed mode supports up to 3.4 MHz, I3C able to go up to 12.5MHz. I3C also supports High Data Rate (HDR) Mode which increased bandwidth via HDR-DDR (Double Data Rate) as the data is sampled in both rising and falling edge of SCL. This is corresponding to the highly increasing bandwidth demand from the more complex sensor usage soon to be made available in the future market.

Figure 2.10 illustrates use of the HDR-DDR Mode. It shows how the Master can change the Mode from SDR Mode to HDR-DDR Mode, and shows example of DDR data format. From the Bus-Free condition, the Master issues a START by driving the SDA Line low while keeping the SCL Line high. The Master then issues the Broadcast Address (0x7E) followed by RnW (0 for Write). Then the Master turns on a pull-up resistor and goes to open drain. All Slaves ‘ACK’ by pulling SDA low (in the Figure, yellow fill means the Slaves are in control of SDA at this time).

The Master then issues the Broadcast Common Command Code for ENTHDR0 (0x20), followed by parity bit “T” (odd parity = 0 for 0x20). At this point, the Bus is in HDR-DDR Mode. In the HDR-DDR protocol, the SDA Line is sampled on every SCL edge (both low-to-high and high-to-low transitions of SCL). The HDR-DDR word consists of a 2 bit preamble, followed by 4 bytes of data, followed by 2 parity bits. The waveform for the 5 bit CRC and following traffic is not shown in the Figure. HDR-DDR Mode is backwards compatible with legacy I^2C devices, because the high time of an SCL pulse is always less than 50ns and as a result SCL will always appear to be low because of the I^2C 50ns glitch filter.

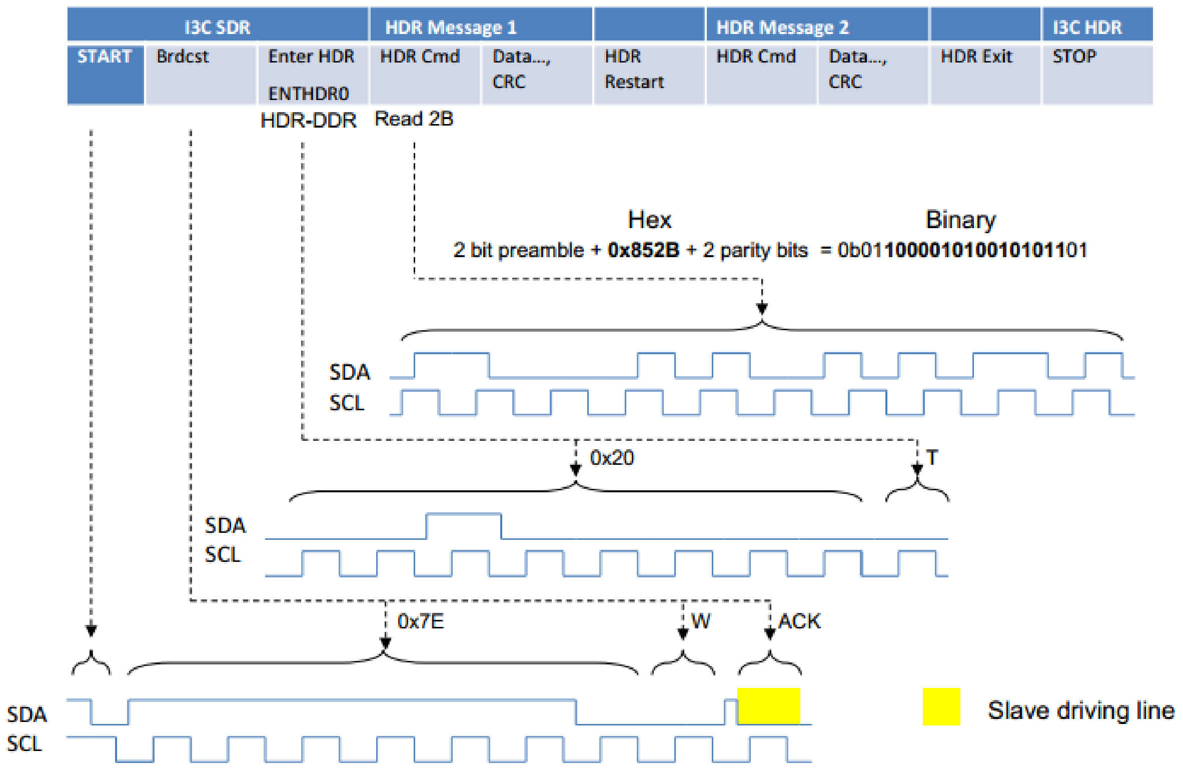

Figure 2.10 I3C HDR-DDR Mode Communication Protocol [2]

Other than increased speed, I3C also offered a few new features that's not available in previous generation, such as hot-join mechanism, peer-to-peer slave cycle and In-band Interrupt. The I3C protocol supports a Hot-Join mechanism, to allow Slaves to join the I3C Bus after it is already started. Hot-Join may be used for:

i. I3C Devices mounted on the same board, but de-powered until needed. Such Devices shall not violate electrical limits for Slaves when depowered (or while transitioning) including capacitive load.

ii. I3C Devices mounted on a module/board that is physically inserted after the I3C Bus has already been configured.