creative commons
COMMONS DEED

저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:

저작자표시. 귀하는 원저작자를 표시하여야 합니다.

비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.

변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 이용허락규약(Legal Code)을 이해하기 쉽게 요약한 것입니다.

Disclaimer

dCollection

공학석사학위논문

A controller design for massive flash storage with an HDMI interface

2014년 8월

서울대학교 대학원

전기 컴퓨터 공학부

Nguyen Duy Thanh

A controller design for massive flash storage with an HDMI interface

지도교수 김 재 하

이 논문을 공학석사 학위논문으로 제출함

2014 년 8 월

서울대학교 대학원

전기 컴퓨터 공학부

Nguyen Duy Thanh

Nguyen Duy Thanh 의 공학석사 학위논문을 인준함

2014 년 8 월

위 원 장 : 채 수 익 (인)

부위원장 : 김 재 하 (인)

위 원 : 이 혁 재 (인)

Abstract

HDMI protocol has become more and more popular in many consumer electronics because it supports high definition, high quality video, multi-channel audio, intelligent format and control word. In this design, a high speed pipelined HDMI interface has been implemented to transfer uncompressed video data between two FPGA chips for storage purpose. Moreover, a NAND flash controller has been designed to take the advancement of new generation of NAND flash chips for massive video data storage. The controller design takes the advantages of combination of software and hardware to execute complicated functions other than basic operations like page program/page read etc. It makes the controller be flexible to control flash memory and reduces hardware cost. The controller supports up to 255 number of linked devices in a daisy-chain configuration without increasing hardware cost and diminishing data throughput. It supports concurrent multi-bank operations to improve system performance. A demo has been made to demonstrate the ability of recording/displaying video of the whole system. A multi-channel controller design can be implemented to speed up and extend storage capacity. Besides, a multi-bit error correction code hardware block need to be added to the controller in the next time to increase the reliability of NAND flash memory.

Keywords—HDMI interface, non-pipelined, pipelined, NAND flash controller, daisy-chain, multi-bank

Student number: 2012-23947

Table of contents

List of figures

1 Introduction

HDMI (High-Definition Multimedia Interface) is a compact audio/video interface for transferring uncompressed video data and compressed or uncompressed digital audio data from a HDMI-compliant source device to a compatible computer monitor, video projector, digital television or digital audio device. The purpose of using this protocol is significantly higher quality compared to video compression. But it requires an extremely large storage space. For example, the data rate for 24bit @1080p @60fps uncompressed video is 2.98Gbps or 1798 GB per/hr! As the vast growth of NAND flash technology and its outperformance and cost-effective compared to hard disk and the need of design for massive uncompressed video data storage, this thesis represents a system design for real-time uncompressed video recording applications using a new generation of massive NAND flash memory for very large and extendable massive storage system. This thesis is divided into 4 chapters. The main contributions of this thesis are in chapter 3 and 4 which are an HDMI communication between 2 FPGA boards and an NAND flash controller design.

Chapter 1: Introduction

Chapter 2: describes whole system description. System-on-chip in Atlys FPGA board and SNUPEE FPGA board are discussed in this chapter. Moreover, a short introduction about new generation of NAND flash memory is also presented.

Chapter 3: discusses about the design of HDMI interface in both Atlys and SNUPEE for transferring video data between them. There are two scheme of HDMI interface: non-pipeline scheme and pipeline scheme.

Chapter 4: describes the NAND flash controller design in detail, its operation and its features.

2 Overall system description

2.1 Overall system description

The overall system shown in Figure 2.1 has the connection between SNUPEE board and Atlys board. SNUPEE board contains a Xilinx Virtex-6 FPGA, a SDRAM memory and NAND flash memory. Atlys board is equipped with a Xilinx Spartan-6 FPGA and DDR2 DRAM memory. An HDMI interface between Atlys board and FPGA board is designed to transfer video data between them.

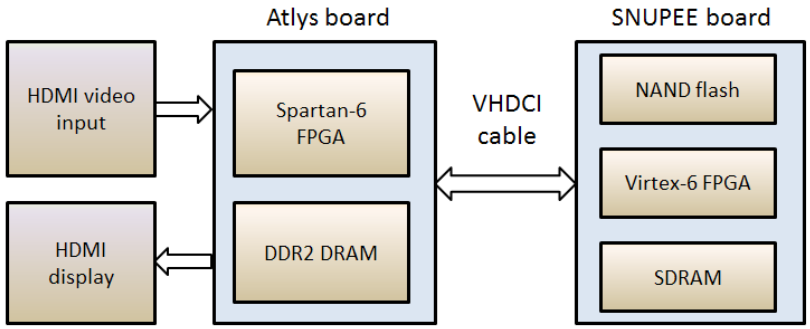

Figure 2.1 Overall system description

The flow of video data is as following. Video data from HDMI input source (for example, a laptop with HDMI port) is decoded into RGB format and written to DDR2 DRAM of Atlys board. Later, video data is sent to SDRAM in SNUPEE board through HDMI interface. Finally, it is stored in NAND flash. That is the transmitting path. In receiving path, in SNUPEE board, video is read back from NAND flash and written to SDRAM. After that, it is sent to DDR2 memory of Atlys board through HDMI interface. Finally, video data is encoded to TMDS (Transition Minimized Differential Signaling) signals to display in HDMI monitor. Section 2.2 and 2.3 below, respectively, describe the system in each FPGA board in further detail.

2.2 System on chip in Atlys FPGA board

Atlys circuit board is a development platform based on Xilinx Spartan 6 LX45 FPGA. It has 128MByte 16-bit DDR2 DRAM memory. The system on chip in Atlys FPGA board as shown in Figure 2.2 mainly has a soft processor Microblaze, HDMI input and output core that are attached directly to an MPMC (Multi-Port Memory Controller) through VFBC (Video Frame Buffer Controller) interfaces. The HDMI output core is configurable within Xilinx Platform Studio and the HDMI input core is software configurable. The system is designed based on PLB (Processor Local Bus) bus. For simplicity, some cores are not drawn in the figure. The GPIO core is connected to 5 buttons in the boards for controlling transceiver the video data. The I2C core can also be attached to control the E-DDC (Enhanced Display Data Channel) lines described in DVI 1.0 standard. When the HDMI connector is connected to output port of computer, an I2C interrupt occurs and the soft processor will send EDID (Extended Display Identification Data) packets to host PC. That EDID packet contain information on which resolutions that the device supports. Currently, in this design, the EDID packet provides fairly general resolution supports. That packet makes host PC recognize the Atlys as an HDMI display device.

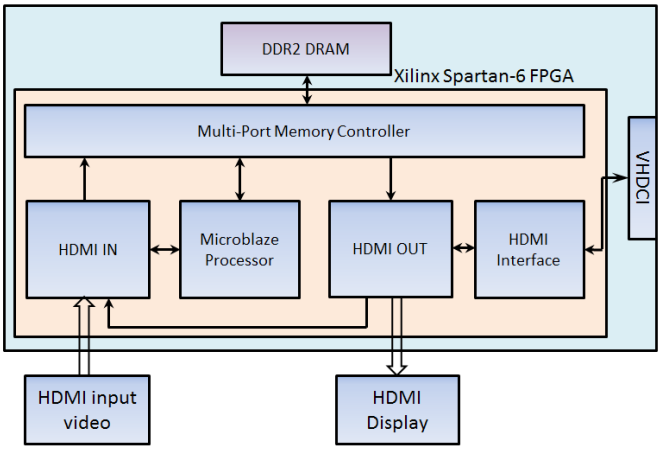

Figure 2.2 System on chip in Atlys board

In this design, the advanced I/O logic and clocking resources, namely IODELAY2, IOSERDES2 of Spartan-6 FPGA have been utilized to build DVI/HDMI transmitter and receiver with higher performance and easier implementation comparing to previous version of Spartan family. HDMI video data from input source video is decoded to RGB format and written to DDR2 memory through VFBC (Video Frame Buffer Controller) interface of Multi-Port Memory Controller (MPMC) core. Microblaze processor can be used to modify the video data stored in DDR2 memory. The HDMI output core encodes video data stored in DDR2 memory to TMDS signals output to HDMI monitor. These following contents are going to be discussed: HDMI protocol overview gives brief information about HDMI signals format, Multi-Port Memory Controller core, HDMI input core and HDMI output core. The HDMI interface between Atlys and SNUPEE board will be presented in chapter 3.

2.2.1 HDMI protocol overview

An HDMI link shown in Figure 2.3 includes 3 TMDS channels and a single TMDS Clock channel. The TMDS Clock channel constantly runs at a rate proportional to the pixel rate of the transmitted video. During every cycle of the TMDS clock channel, each of the three TMDS data channels transmits a 10-bit character. The input stream to the Source's encoding logic will contain video pixel, packet and control data. The packet data consists of audio and auxiliary data and associated error correction codes.

These data items are processed in a variety of ways and are presented to the TMDS encoder as either 2 bits of control data, 4 bits of packet data or 8 bits of video data per TMDS channel. The Source encodes one of these data types on any given clock cycle.

For simplicity, this project just considers the pixel information and control signal (HSYNC, VSYNC, CTL0, CTL1, CTL2, CTL3). Auxiliary data is discarded.

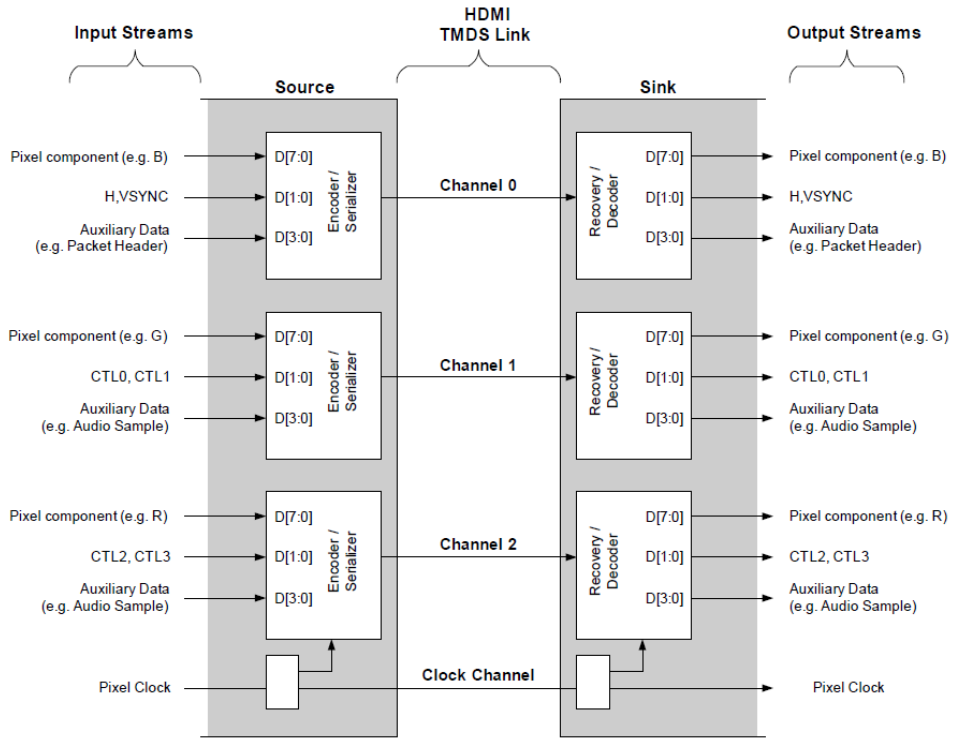

Figure 2.3 HDMI protocol overview

2.2.2 Multi-Port Memory Controller (MPMC)

MPMC is a fully parameterizable memory controller designed by Xilinx that supports SDRAM/DDR/DDR2/DDR3/LPDDR memory. MPMC provides access to memory for one to eight ports, where each port can be chosen from a set of Personality Interface Modules (PIMs) that permit connectivity into PowerPC 405 processors and MicroBlaze processor using IBM CoreConnect ToolKit Processor Local Bus (PLB) and the Xilinx CacheLink (XCL) structures, as well as a Memory Interface Block (MIB) PIM (PPC440MC) for the PowerPC 440 Processor. MPMC supports the Soft Direct Memory Access (SDMA) controller that provides full-duplex, high-bandwidth, LocalLink interfaces into memory. A Video Frame Buffer Controller (VFBC) PIM is also available. For low-level direct access to the memory controller core, a Native Port Interface (NPI) PIM is available for soft memory

controllers and the Memory Controller Block (MCB) PIM is available for the Spartan-6 hard memory controller.

Video Frame Buffer Controller (VFBC)

VBFC core is a fully functional VHDL design implemented on a Xilinx FPGA. The VFBC connects to the Multi-Port Memory Controller Native Port Interface (NPI) as a Port Interface Module (PIM) and allows user IP to read and write data in two-dimensional sets regardless of the size or the organization of external memory transactions.

2.2.3 HDMI input core

One of the major advancements in the Spartan-6 is its I/O architecture. It has new advanced I/O logic and clocking resources, namely, IODELAY2, IOSERDES2 and the phase-locked loop (PLL) and I/O clock distribution network. IODELAY2 provides calibrated delay taps for the TMDS video stream to deskew and phase align on a per-pair basis. IOSERDES2 helps with serialization and deserialization of video pixels using dedicated and hardened circuitry. The PLL synthesizes a high-speed bit rate machine clock, and the I/O distribution routes the clock to designated IODELAY2 and IOSERDES2 elements via a dedicated path. This not only gives a higher performance improvement over the Spartan-3A family but allows the actual DVI/HDMI transmitter and receiver to be built in a much easier manner.

The HDMI input core block diagram is shown in Figure 2.4. It incorporates a TMDS decoder core, Control and Status Register, main controller and VFBC controller. The decoder core recovers the bit sampling clock using the incoming pixel clock and applies the bit clock to recover the serial data stream back into 10-bit word aligned symbols. The Control and Status Register responds to read and write requests from Bus clock domain. Requests are immediately handled and acknowledged over a FIFO in that operates in opposite position. The VFBC controller generates command sequences to write HDMI frames to DDR2 memory.

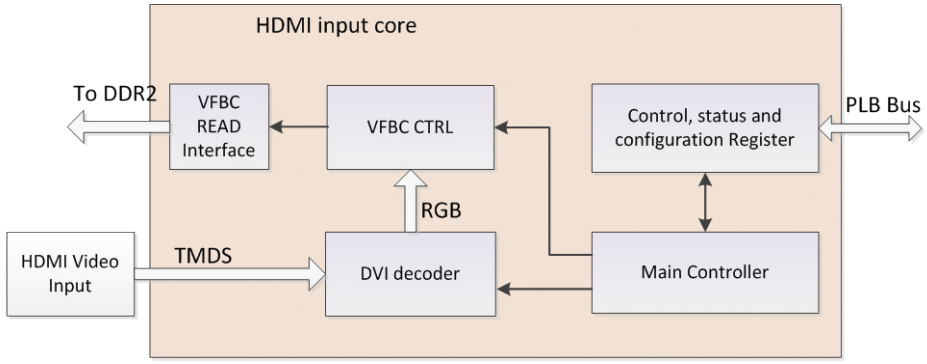

Figure 2.4 HDMI input core block diagram

2.2.3.1 TMDS Decoder

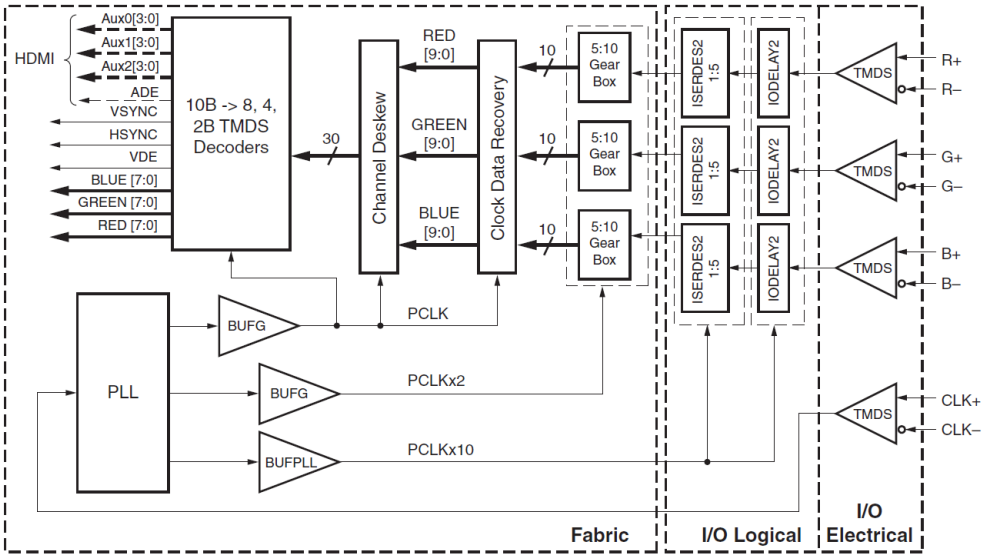

Figure 2.5 TMDS decoder design

The TMDS decoder block diagram shown in Figure 2.5 includes two mains stages: Deserialization stage and Clock and Data Recovery stage.

Deserialization:

It is based on hardened circuitry working in Single Data Rate (SDR) mode. Each input pin in the Spartan-6 FPGA comes with a built-in ISERDES2 block that

converts 1-bit serial stream to a maximum 4-bit parallel bus. By cascading with another ISERDES2 block within its adjacent input pin, the hardened circuitry is able to perform serialization with a maximum 1:8 ratio. The ISERDES2 blocks are fully configurable to work in either cascading or non-cascading mode with any ratio from 1 to 8. To achieve 1:10 deserialization, two stages of conversion must be undertaken: 1:5 ISERDES2 cascading and 1:2 logic gear box.

Clock and Data Recovery:

The TMDS Clock channel carries a character rate frequency reference from which the decoder reproduces a bit rate sample clock for the incoming serial streams. The reproduced bit rate clock, however, does not have a guaranteed phase relationship associated with any of the three data lanes. In addition, both DVI and HDMI specification allow certain skew between any two data lanes. As a result, the clock phase must be adjusted individually for each data lane to correctly sample the incoming serial bit. This involves aligning the clock rising edge to the middle sampling window of each data lane. Like deserialization stage, the Clock and Data Recovery stage is mostly done through hardened circuitry. The bit sampling clock is reproduced through a PLL using the incoming TMDS pixel clock as a reference. The TMDS clock bit multiplied by 10 to match the bit rate and then fed to a BUFPLL that routes the 10x clock to designated ISERDES2 and IODELAY2 blocks. This clock is named IOCLK. Figure 2.5 shows the clock connections.

Other than deserialization of serial bits into a parallel bus, the ISERDES2 block in the Spartan-6 FPGA possesses a unique “phase detector” function when operating in cascading mode. By sampling and comparing the incoming serial data samples using the two ISERDES2 block residing in two adjacent differential input pins, the phase detector is able to determine the phase relationship between the current IOCLK rising edge and the serial data transition edge. Subsequent validation and control signals are sent to the FPGA logic for a soft controller state machine to adjust the IODELAY2 accordingly.

The IODELAY2, on the other hand, provides a dynamically adjustable delay line to the incoming serial data bit. Upon receiving the control signals from the controller state machine, the IODELAY2 is able to align the rising edge of IOCLK to the middle of the data sampling window.

Data lane deskew:

To remove the allowed data lane skew (up to 0.6 pixel time), FIFO-based deskew logic is built. The channel deskew circuit shown in Figure 2.6 uses a handshake negotiation scheme between all three channels to remove any skew beyond the specification limit. Each channel receives a signal from the phase alignment units to indicate if the incoming 10-bit symbols are valid. If all channels are valid (have achieved phase alignment), a FIFO inside each deskew module start passing data through (continuously writing in and reading out). When a control token is detected in any of the FIFO outputs, the read out flow is suspended and a ready signal is generated to indicate the arrival of a particular marker in the video stream. The read

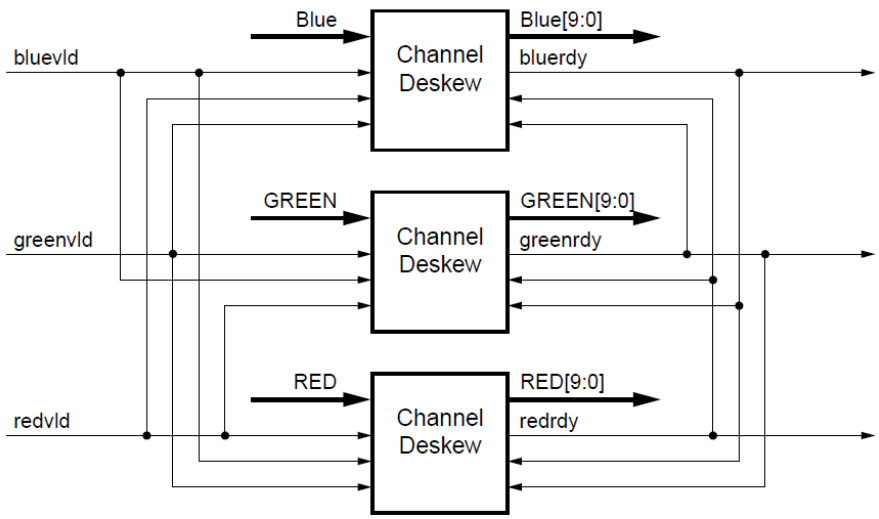

Figure 2.6 Channel-to-channel deskew

out flow resumes only when this marker has arrived on all three channels. Thus, the relevant skew has been removed. The FIFO uses the distributed RAM resources and

provides a depth of 16 words. Theoretically, the circuit can remove up to 16 pixels of channel to channel skew.

2.2.3.2 Main controller

This main controller is responsible to frame detection and writing frame or line data to DDR2 memory. The block diagram of state machine is drawn in Figure 2.7. The state description is given below:

FRAME_RST: starts the state machine and reset internal registers.

FRAME_FIND_POL: waits for frame video available.

FRAME_WAIT_VSYNC: if VSYNC signals goes “High”, it means that new frame has come. So “new_frame” signal is asserted.

FRAME_LOCKED: in this state, no new frame is written to DDR2. If there is a read frame request (read frame from SNUPEE and send to Atlys board or BT2 operation), the state machine jumps to FRAME_LOCKED_R for writing line by line data from SNUPEE to DDR2. Otherwise, if “write_en” signal goes high, it allows new frame from HDMI input port to be written to DDR2 memory. “write_en” signal is controlled by pushing button C in Atlys board.

FRAME_WR_CMD: controller passes the write command to VFBC COMMAND interface which includes: frame base address, frame height, frame width, and line stride.

FRAME_WR_DATA: controller writes the whole frame from HDMI input port to DDR2 through VFBC DATA interface.

FRAME_LOCKED_R: waits for line data available in input buffer by tracking BT2_DATA_DDR_COPY_DONE_WAIT_i signal from HDMI output core.

FRAME_WR_CMD_R and FRAME_WR_DATA_R are similar to FRAME_WR_CMD and FRAME_WR_DATA. The only difference is now the frame has only one line data.

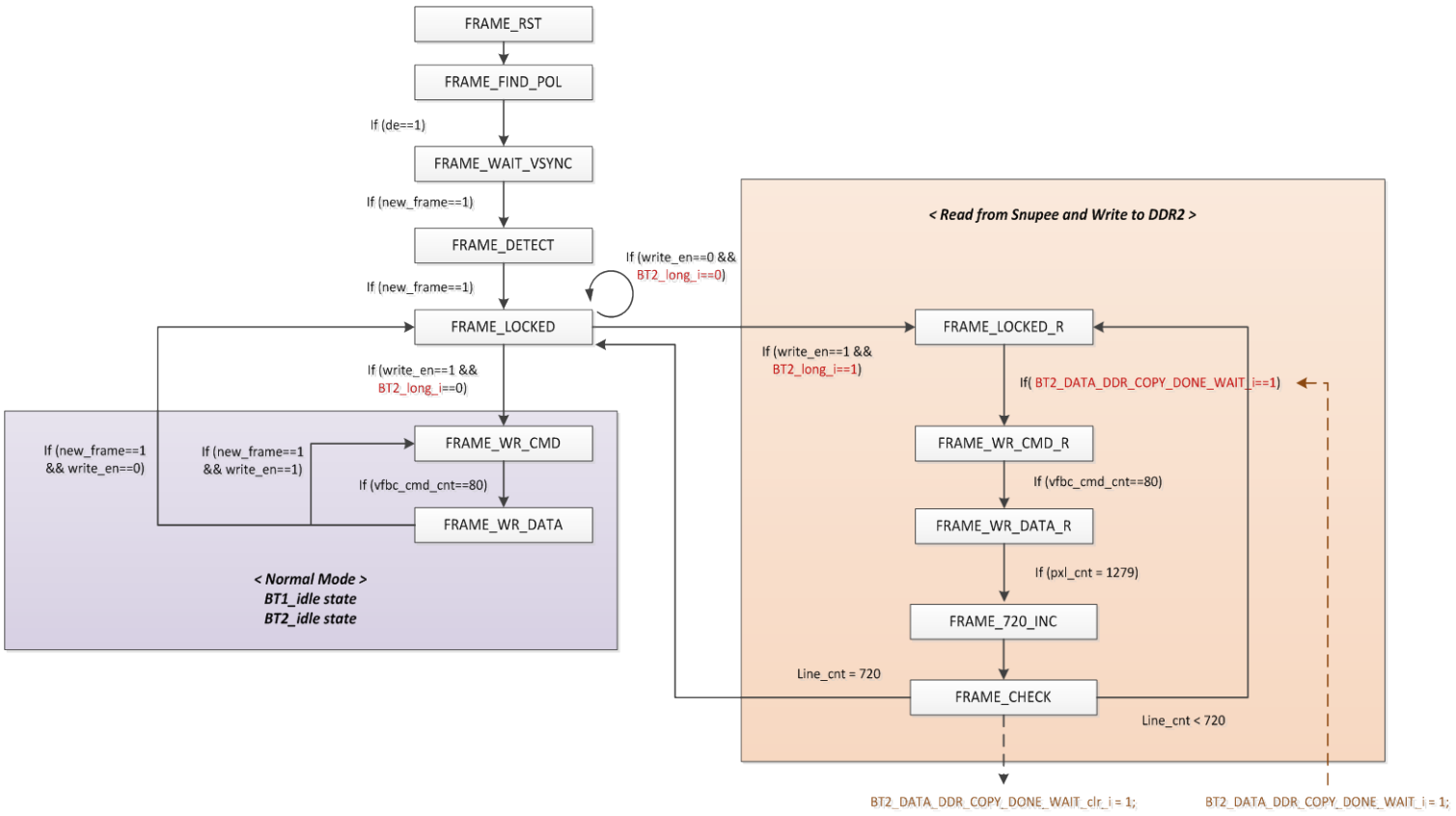

Figure 2.7 Block diagram of main controller state machine in HDMI input core

FRAME_720_INC: increases the line counter.

FRAME_CHECK: sends BT2_DATA_DDR_COPY_DONE_WAIT_clr acknowledge signal to HDMI output core to notify that one line write to DDR2 has completed. If the whole frame is received successfully from SNUPEE board, the state machine jumps to FRAME_LOCKED to wait for new request.

2.2.3.3 Control, Status and Configuration Registers

The register description is given along with memory offsets from the HDMI_IN base address. Bit numbers are assigned such that Bit(0) = MSB and Bit(31) = LSB. All registers are only accessible when a video source is attached and providing a valid clock signal. Registers are reset whenever a video source is disconnected.

Control Register: 0x00 : Read/Write

Bits(0:30) – Reserved

Bit(31) – Write Enable – Enables the core to begin writing video data to the frame buffer. Default is '0'.

Status Register: 0x04 : R

Bits (0 : 29) – Reserved

Bit(30) – Frame Locked – Reading value of '1' means that the frame dimensions of the incoming video signal have been determined.

Bit(31) – PLL Locked – Reading a value of '1' means that a valid clock signal is detected on the TMDS line.

Frame Width Register: 0x08 : R

Bits (0 : 15) – Reserved

Bits (16 : 31) – Unsigned value that represents the width of the input frame in pixels.

Frame Height Register: 0x0C : R

Bits (0 : 15) – Reserved

Bits (16 : 31) – Unsigned value that represents the height of the input frame in lines, minus 1.

Line Stride Register: 0x10 : R/W

Bits (0 : 15) – Reserved

Bits (16 : 31) – Unsigned value that defines the line stride of the frame in pixels. This value must have bits 25 to 31 equal to zero in order to be 128 byte aligned. The default value is 0 and must be set before enabling the core.

Frame Based Address Register: 0x14 : R/W

Bits (0 : 31) – Unsigned value that defines the physical address of the frame buffer. This address must fall somewhere within the DDR2 memory space and have enough trailing memory to fit the entire frame. This value must have bit 25 to 31 equal to zero in order to be 128 byte aligned. Default value is 0 and must be set before enabling the core.

2.2.4 HDMI output core

The HDMI output core shown in Figure 2.8 mainly has following components: VFBC_CTRL controller, BT1_CTRL controller, BT2_CTRL controller and DVI output module.

VFBC_CTRL: when there is no read/write request (no BT1 or BT2 pressed), this controller continuously reads the whole frame from DDR2 and sends to DVI encoder. DVI encoder encodes RGB pixel into TMDS signals and sends to HDMI TV for displaying.

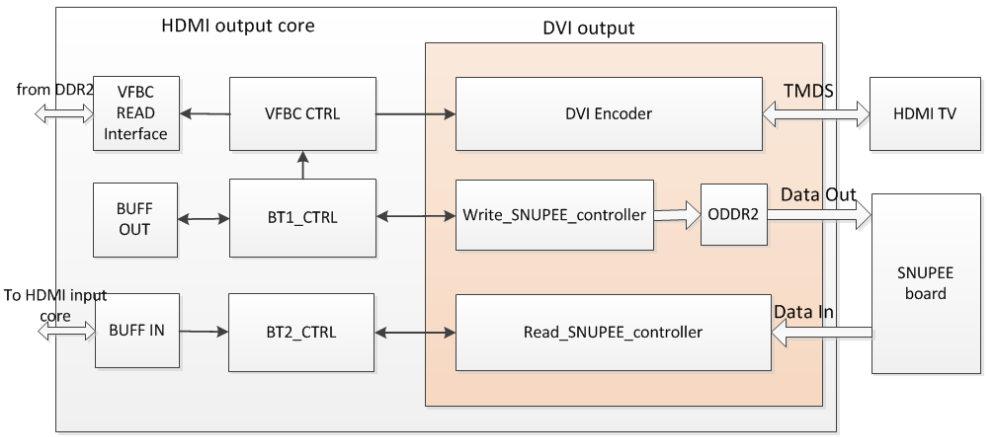

Figure 2.8 HDMI output core block diagram

BT1_CTRL controller: when there is request sending frame to SNUPEE board (BT1 operation), this state machine controls VFBC interface to read line by line video frame from DDR2 memory and send to SNUPEE board.

BT2_CTRL controller: when there is request reading frame from SNUPEE board (BT2 operation), this state machine controls reading video frame from SNUPEE board and sending video data to HDMI input core.

Write_SNUPEE_controller state machine works together with BT1_CTRL for sending HDMI frames to SNUPEE board. Read_SNUPEE_controller state machine works together with BT2_CLR to read frames from SNUPEE board.

BUFF OUT and BUFF IN are line buffers used for temporarily store line data before writing to DDR2 or sending to SNUPEE FPGA board.

The detail of these state machines is discussed in chapter 3.

2.2.4.1 DVI Encoder

The DVI Encoder is logically divided into two parts: TMDS encoding and 10-bit parallel to serial conversion. The block diagram of DVI Encoder is Figure 2.9.

Serializer:

Similar to DVI decoder, serializer design in Spartan-6 FPGA is compeletely based hardened circuitry. Each input pin in the Spartan-6 FPGA comes with a built-in OSERDES2 block that converts a maximal 4-bit parallel bus to a 1-bit serial stream. By cascading with another OSERDES2 block within its adjacent output pin, the design is able to perform serialization with a maximal 8:1 ratio. The OSERDES2 blocks are fully configurable to work in either cascading or non-cascading mode with any ratio ranging from 1 to 8 according to *Spartan-6 FPGA SelectIO Resources User Guide*. To perform the required 10:1 serialization for both DVI and HDMI, two stages of conversion must be adopted: a 2:1 soft gear box and 5:1 OSERDES2 cascading.

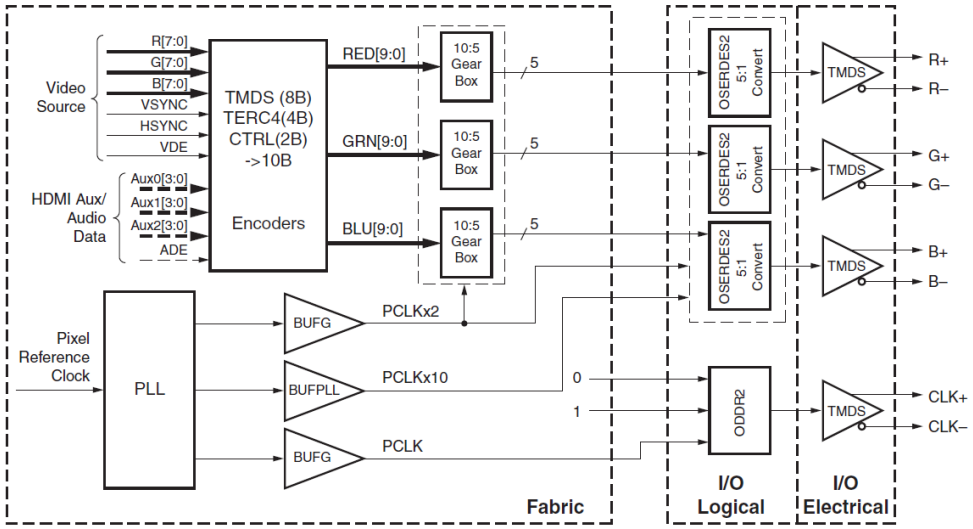

Figure 2.9 DVI Encoder block diagram

Clocking Scheme:

Based on the high-performance PLL and dedicated I/O clock network, both the soft gear box 2:1 and the hard 5:1 OSERDES2 block work in single data rate (SDR) mode with higher performance than that of Spartan-3A FPGA. The PLL takes the oscillator clock inside of Atlys board as input reference clock and synthesizes three clocks: a pixel clock, a 10x pixel clock and a 2x pixel clock. The 10x pixel clock is used to match the serial data bit rate. It is routed to OSERDES2 clock through a dedicated BUFPLL driver. The OSERDES2 block also takes the 2x pixel clock as its 5-bit parallel data input reference.

The 2:1 soft gear box takes both the pixel clock and the 2x pixel clock. It converts the 10-bit TMDS encoded data into 5-bit data stream.

2.3 System on chip in SNUPEE board

The system architecture in SNUPEE board as shown in Figure 2.10 is designed based on AHB (Advanced High-performance Bus) bus system. It has a RISC processor for controlling whole system, a SDRAM controller providing interface

from host to Synchronous DRAM, a hardware NAND flash controller for accessing new generation of NAND flash memory. An HDMI (CM_HDMI module) interface has been designed to transfer video data between SNUPEE board and Atlys board in real-time through a VHDCI (Very-High Density Cable Interconnect) port.

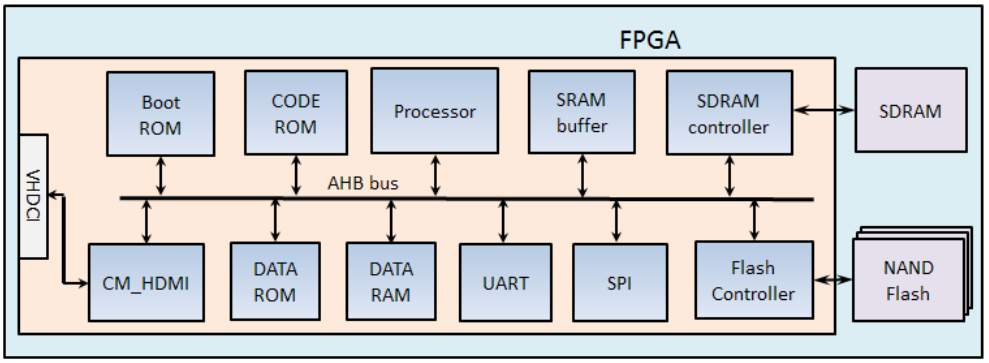

Figure 2.10 System on chip in SNUPEE board

UART module is used for debugging in host computer. SPI master module provides processor the access to serial on-board flash to support running system on external firmware. It will save a lot of time since the firmware of the system can be updated without re-synthesizing the whole system. The flash controller runs on both software and hardware to make flash controller be more flexible and hardware cost effective.

2.4 NAND flash memory

The NAND flash device of MOSAID company is a new generation of flash memory delivering the advanced capabilities with high performance or a broad range of flash applications [1]. This flash memory based on NAND flash cell technology provides the most cost-effective solution for mass storage applications. NAND is flash device packaged in an MCP (Multi-Chip-Package), composed of a stack of 9 dies, including eight monolithic independent 32Gb MLC (multi-level-cell) Toggle Mode NAND flash chips, evenly distributed in four banks, and one high speed ASIC interface chip. This device supports an Error Correction Code (EDC) feature to

eliminate bit errors in “Command Packets” to ensure reliability and error-free communication of commands and register data.

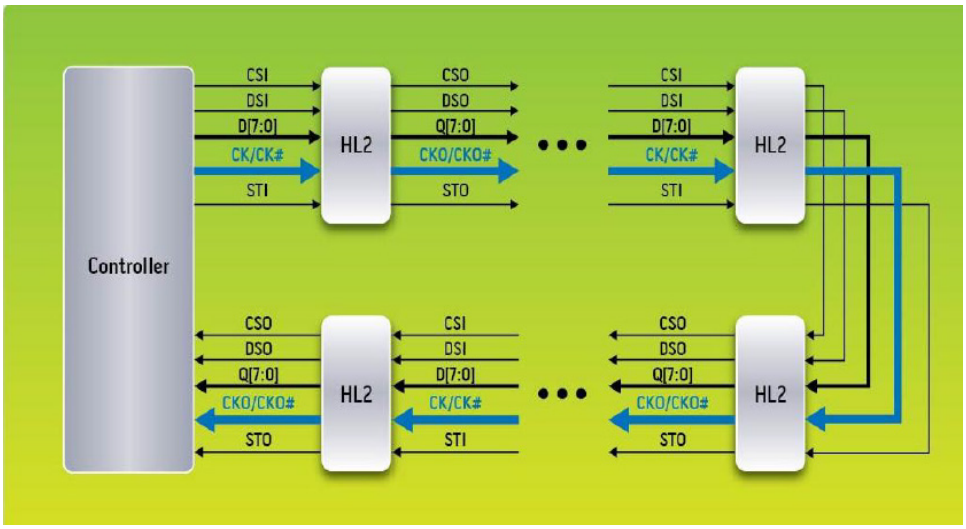

Figure 2.11 Daisy-chain topology for extendable storage system

It provides user configurable virtual page size for read with the various page depth choices. The unified synchronous interface and protocol of NAND flash greatly improves data throughput while supporting feature-rich operation. NAND’s serial interface with high clock rate, Double Data Rate technology achieve a peak bandwidth of 800MB/s and maximum clock frequency of 400MHz. NAND flash devices are connected to the host controller, and each other, in a simple daisy-chained ring topology that accommodates a virtually unlimited number of devices without suffering from throughput reducing loading effects, thus facilitating system integration with greater expandability and flexibility. Especially, a serial daisy-chain ring of flash device, shown in Figure 2.11, provides a unidirectional flow of data and commands from the controller through each memory device and back to the controller. The pin configuration is identical to one of the single device. This reduces cost and power consumption and provides more PCB space for more NAND devices.

3 Pipelined scheme for high speed HDMI Interface

3.1 Board interface

HDMI interface is designed to transfer video data between Atlys and SNUPEE board. Two FPGA boards are connected by a high speed VHDCI (Very-High Density Cable Interconnect) cable. The pin names are shown in Figure 3.1. The transmitter clock (CM_SP_CLK) is Atlys's system clock which is 75 MHz. With 16-bit data width of the HDMI interface, it provides maximum bandwidth of transmitter is 150 MB/s. The receiver clock (CLK_SNUPEE) is SNUPEE's system clock which is 50 MHz. Hence, the maximum receiver bandwidth is 100 MB/s.

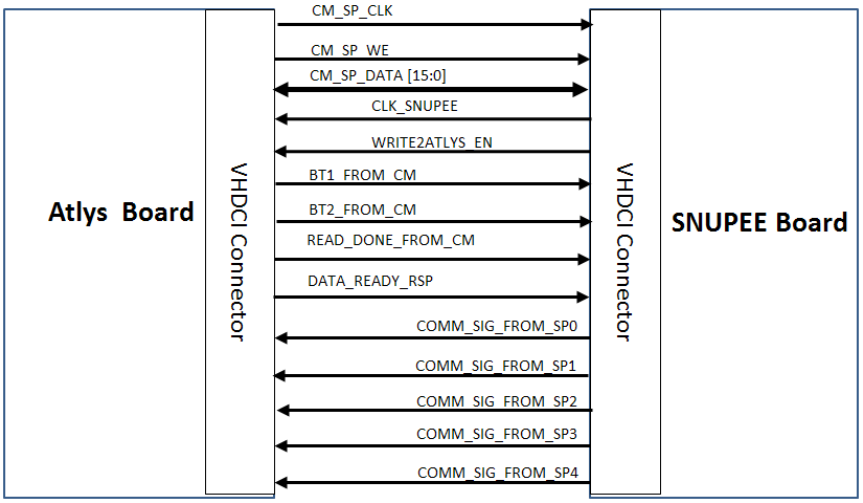

Figure 3.1 VHDCI pin connection

There are two schemes of HDMI interface: non-pipelined scheme and pipelined-scheme which are presented in the following sections. The detail meaning of each signal is also described in section 3.2 below.

3.2 Non-pipelined scheme

The BT1 operation is the transmit path and BT2 operation is the receive path. As shown in Figure 3.2, the transmit path includes 3 stages: reads line video from

DDR2 memory and writes to output buffer OBUFA in Atlys board (Tx1), reads OBUFA and writes to buffer in IBUFS of SNUPEE (Tx2), reads IBUFS and writes to SDRAM (Tx3). The next stage can start only after the current stage finish. Similarly, the receive path includes 3 stages: reads line data from SDRAM and writes to output buffer OBUFS (Rx1), reads line data from OBUFS and writes to input buffer IBUFA (Rx2), reads line data from IBUF and writes to DDR2 memory (Rx3).

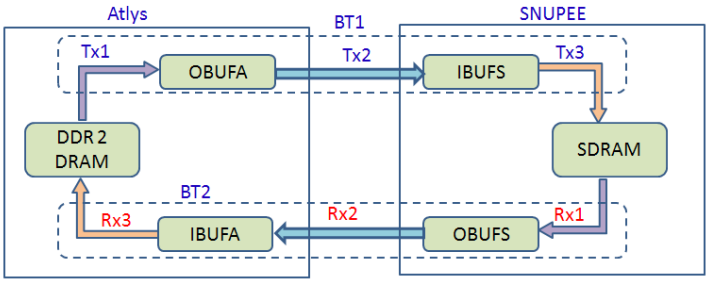

Figure 3.2 Non-pipelined HDMI interface

The send data packet timing is shown in Figure 3.3. When Atlys requests a frame write (BT1_FROM_CM goes “High”) to SNUPEE board, SNUPEE sends an acknowledge signal (COMM_SIG_FROM_SP0) to Atlys. Then Atlys starts to read one line data from DDR2 memory and write to output buffer (stage Tx1). Next, it asserts CM_SP_WE# to “Low” to write line data to input buffer of SNUPEE board (stage Tx2). The signal named COMM_SIG_FROM_SP2 goes “High” means that stage Tx2 has completed. Finally, SNUPEE writes line data from input buffer to SDRAM (stage Tx3). The completion of this stage is detected by the signal COMM_SIG_FROM_SP4. This is end of one line transfer. These operations are repeated until a full frame is written to SDRAM of SNUPEE board.

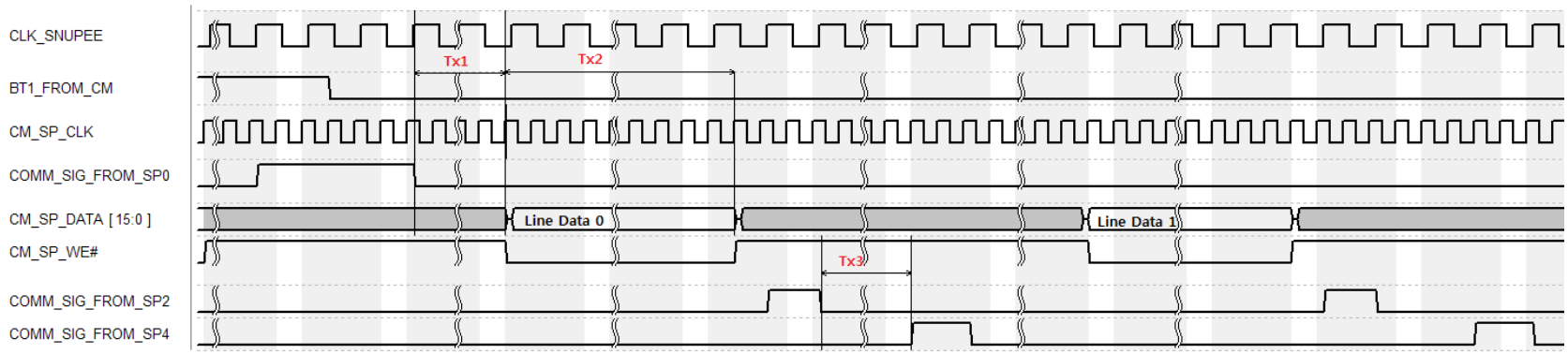

Figure 3.3 Send data packet timing

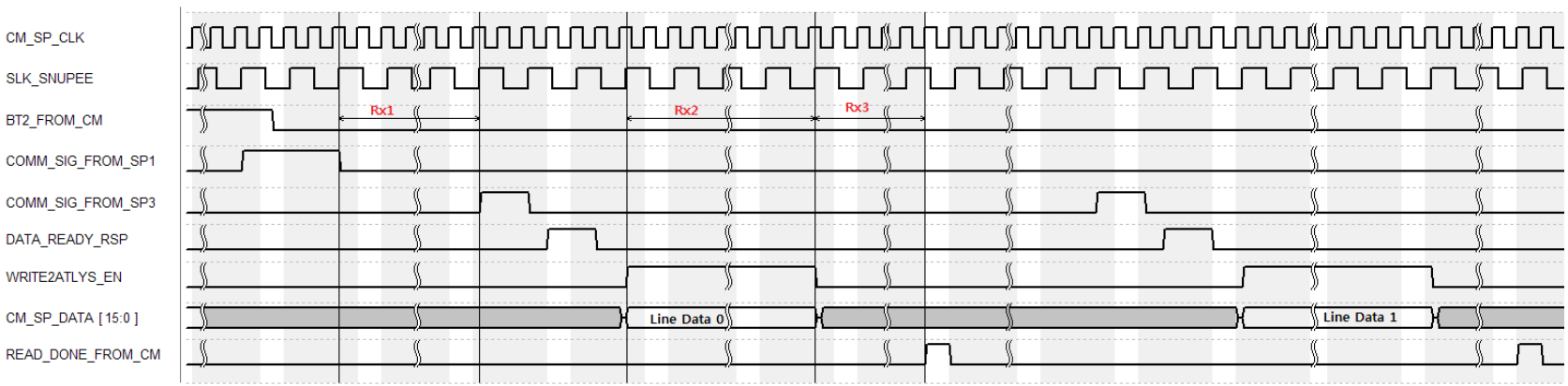

Figure 3.4 Receive data packet timing

The receive data packet timing is demonstrated in Figure 3.4. When Atlys requests a frame read (BT2_FROM_CM goes “High”), SNUPEE sends an acknowledge signal (COMM_SIG_FROM_SP1) back to Atlys and starts to read one line data from SDRAM (stage Rx1). Atlys tracks the completion of this stage by monitoring COMM_SIG_FROM_SP3 signal. Once Atlys receives this signal, it sends a acknowledge signal DATA_READY_RSP to SNUPEE to indicate that it is ready to receive data. After that, SNUPEE activates signal WRITE2ATLYS_EN and sends one line data to input buffer of Atlys (stage Rx2). After one line data is written to input buffer, Atlys reads it and writes to DDR2 memory (stage Rx3). Once, this stage completes, it sends READ_DONE_FROM_CM signal to SNUPEE to end one line receive process. The operation is repeated until a full frame is written to DDR2 memory of Atlys.

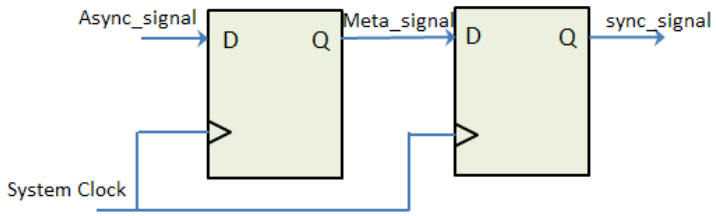

Figure 3.5 2DFF synchronizer

Since systems in Atlys and SNUPEE are asynchronous, synchronizer must be used for each input signals coming from one FPGA chip to the other chip. If not, the state machine in each system would work in a very strange manner due to the timing violation of asynchronous signals fed to state machines. For fabric flip flop, the metastability is quite low. So two flip flops are sufficient for almost FPGA applications. The principle is that the first flip-flop might go into a metastable state if it samples an invalid signal. But on the next clock edge, it will almost certainly have recovered from that state so that the second flip-flop almost always samples valid levels. Sometimes (very rarely) second flip-flop samples invalid level because all synchronizers have a finite mean time between failure (MTBF). An easy way to increase the MTBF is to add more flip-flops in cascaded manner.

The 2-flip flop synchronizer shown in Figure 3.5 is easy to implement in Verilog/VHDL but on Xilinx FPGAs, one thing to consider is that if 2DFF synchronizer is described a standard shift-register, the Xilinx Synthesis Tool (XST) is very likely to merge all the flip-flops of the cascade into a shift-register LUT (SRL) primitive. Unfortunately, SRLs are not a good choice for implementing synchronizers because they don't contain full-fledged flip-flops. Instead, they are more like chained SRAM cell with bad metastability characteristics. The solution to prevent XST from inferring SRLs is to use SHREG_EXTRACT synthesis constraint.

Detail of state machines of HDMI interface in each FPGA board is presented in section 3.2.1 and 3.2.2 below.

3.2.1 HDMI interface in Atlys board

3.2.1.1 Transmitter

The transmitter operation (BT1 operation) is described as in the Figure 3.6.

BT1_CTRL state machine:

When BT1 is pressed, BT1_CTRL state machine starts to operate.

BT1_DATA_READ_FROM_DDR2: controller sends the VFBC line read request to VFBC CTRL. Line data is read from DDR2 and written to CM_BUFFER_OUT. When one line data reading is done, state machine jumps to BT1_DATA_READY_TO_CM_SP state.

BT1_DATA_READY_TO_CM_SP: sends ready signal to Write_SNUPEE state machine and waits for the acknowledged signal from Write_SNUPEE state machine.

BT1_CM_SP_COPY_DONE_WAIT: waits for one line completely transferred to SNUPEE board.

BT1_720_INC: increases the line counter and BT1_720_CHECK to check whether whole frame was completely transferred to SNUPEE board. If done, jumps to BT1_IDLE.

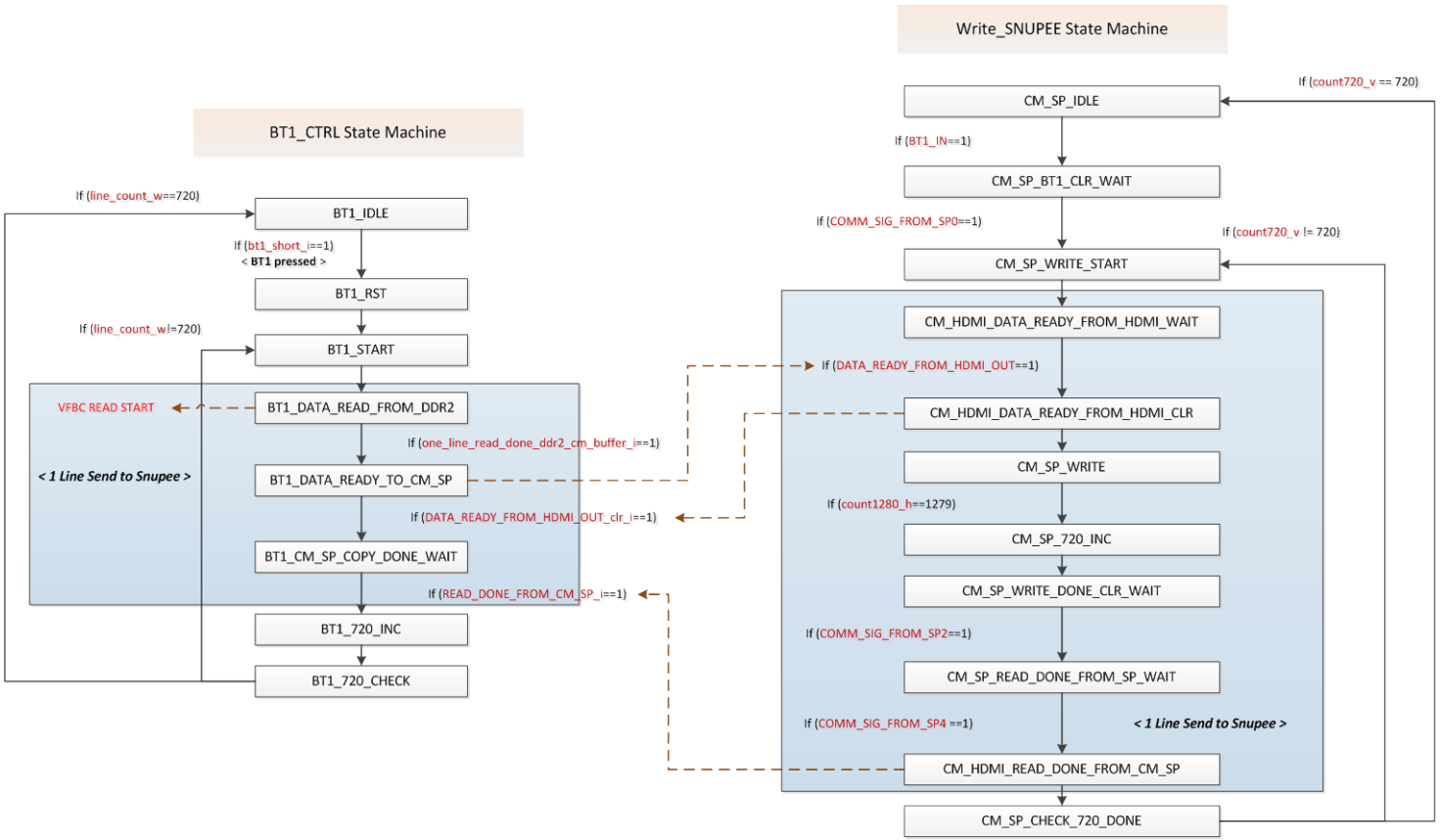

Figure 3.6 Transmitter state machine in Atlys board

The state machine can easily be extended to operate in multi-frame mode.

Write_SNUPEE state machine:

CM_SP_BT1_CLR_WAIT: When BT1 is pressed, state machine sends BT1_FROM_CM to SNUPEE and waits for the response from SNUPEE. If COMM_SIG_FROM_SP0 goes "High", BT1 state machine in SNUPEE is ready to receive frame data and Write_SNUPEE state machine can jump to next state.

CM_HDMI_DATA_READY_FROM_HDMI_WAIT: if state machine receives the signal indicating that line data is ready in CM_BUFFER_IN, it jumps to CM_HDMI_DATA_READY_FROM_HDMI_CLR to send the acknowledged signal to BT1_CTRL state machine.

CM_SP_WRITE: reads line data from CM_BUFFER_OUT and writes to input buffer of SNUPEE board.

CM_SP_720_INC: increases the line counter by one.

CM_SP_WRITE_DONE_CLR_WAIT: waits for write line data to input buffer of SNUPEE is done by tracking signal COMM_SIG_FROM_SP2 from SNUPEE.

CM_HDMI_READ_DONE_FROM_SP_WAIT: waits for write line data to SDRAM of SNUPEE is done by tracking signal COMM_SIG_FROM_SP4 from SNUPEE.

CM_HDMI_READ_DONE_FROM_CM_SP: sends "done" signal to BT1_CTRL state machine.

CM_SP_CHECK_720_DONE: checks whether full frame is sent to SNUPEE or not.

3.2.1.2 Receiver

Receiver state machine (BT2 operation) block diagram is shown in Figure 3.7.

BT2_CTRL state machine:

If BT2 is pressed, the state machine starts to read a frame from SNUPEE to write to DDR2.

BT2_DATA_READY_FROM_CM_SP_WAIT: waits for line data is ready in input buffer by tracking signal DATA_READY_FROM_CM_SP_i from Read_SNUPEE state machine.

BT2_DATA_DDR_COPY_DONE_WAIT: sends signal to HDMI input core to start writing line data to DDR2 memory. When write is done, it receives responding signal BT2_DATA_COPY_DONE_WAIT_clr_i from HDMI input core and jumps to BT2_DATA_READ_DONE_TO_CM_SP.

BT2_DATA_READ_DONE_TO_CM_SP: READ_DONE_FROM_HDMI_OUT signal is sent to Read_SNUPEE state machine to notify that line data has been written to DDR2 successfully.

BT2_720_INC: increases line counter for next line data transfer.

BT2_720_CHECK: if one frame is written to DDR2, state machine goes to BT2_IDLE, otherwise, starts next line transfer.

Read_SNUPEE state machine:

When BT2 is pressed, the state machine starts to operate.

CM_SP_BT2_CLR_WAIT: Sends start signal (BT2_FROM_CM) to BT2 state machine in SNUPEE and waits for acknowledgement of SNUPEE by tracking COMM_SIG_FROM_SP1 signal.

CM_SP_DATA_READY_WAIT: waits for line data being read from SDRAM to output buffer of SNUPEE by tracking COMM_SIG_FROM_SP3 signal.

CM_SP_DATA_READY_RSP: notify SNUPEE that Atlys is ready to receive line data.

CM_SP_READ: line data is read from output buffer of SNUPEE and written to input buffer of Atlys.

CM_SP_720_R_INC: increase line counter by one.

CM_HDMI_DATA_READY_FROM_CM_SP: DATA_READY_FROM_CM_SP signal is sent to BT2_CTRL state machine to inform that line data is ready in input buffer.

CM_HDMI_READ_DONE_FROM_HDMI_WAIT: waits for line data being written to DDR2 by HDMI_IN module. CM_SP_CHECK_720_R_DONE: if one frame is written to DDR2 then state machine goes to idle state. Otherwise, it jumps to CM_SP_READ_START for next line data transfer.

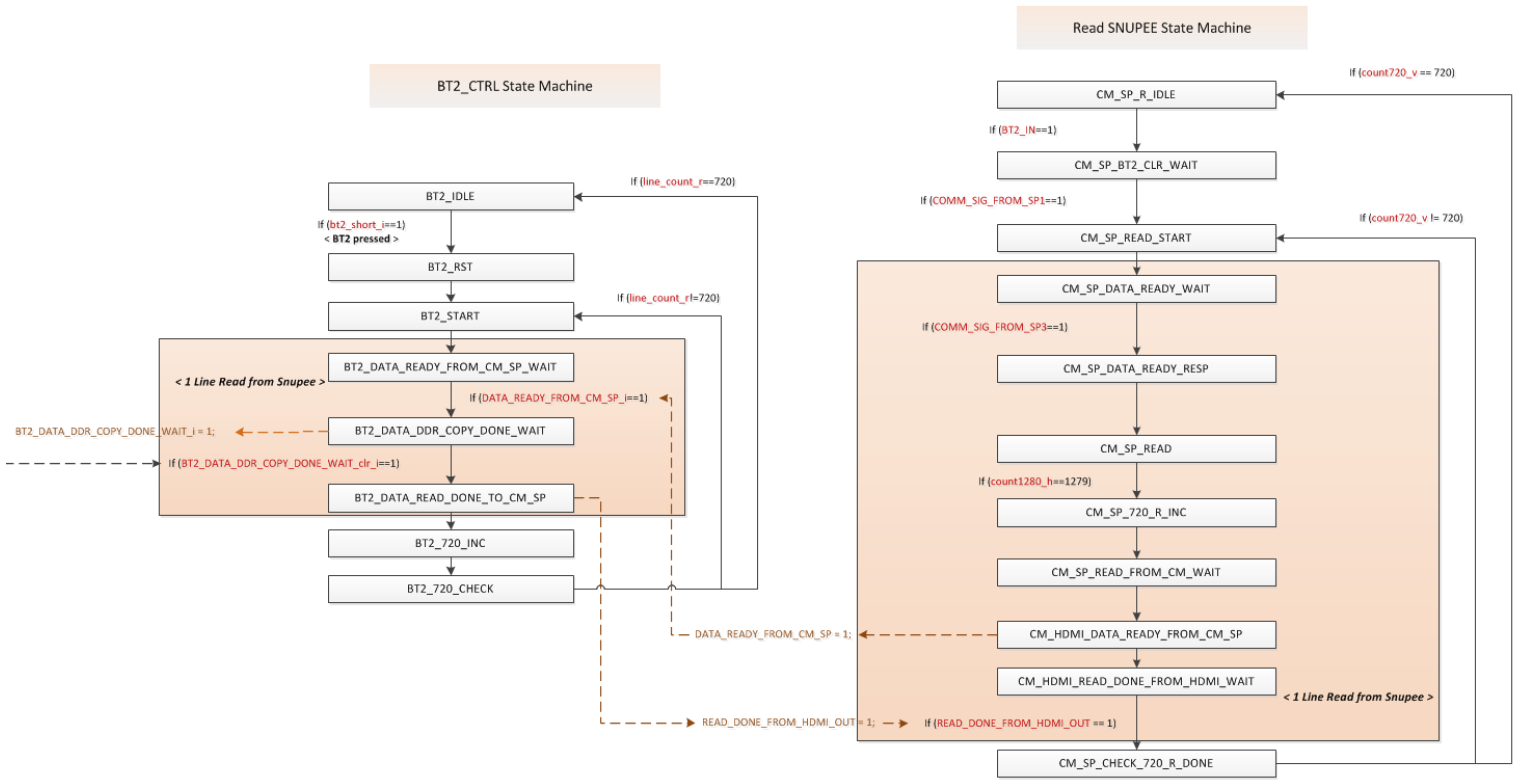

Figure 3.7 Receiver state machine in Atlys board

Scheme for clock forwarding and data output to pin

In proposed design, the transmitter clock and receiver clock are different. Transmit data and receive data are aligned with transmitter clock and receiver clock, respectively. Since the VHDCI cable is long (1 meter or above), the clock and data signals will have big delay and is potentially unstable along the transmitter or receiver path, this design has adopted the dedicated technique using Double Data Rate registers for clock forwarding and data transferring to reduce skew and maximize stability. The clock forwarding technique recommended by Xilinx applied for Xilinx Spartan-6 FPGA as in Figure 3.8:

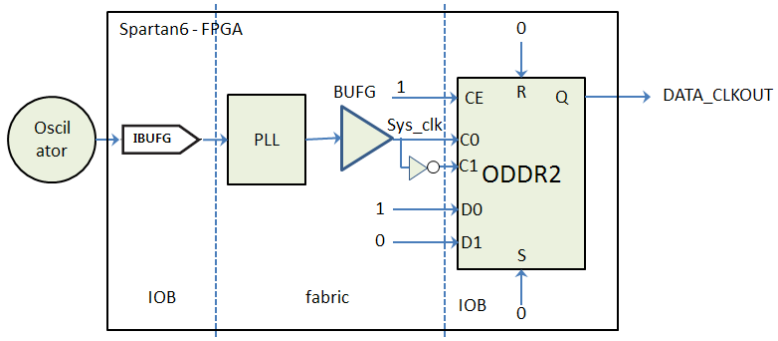

Figure 3.8 Clock forwarding technique

Xilinx Spartan-6 FPGA supports Double Data Rate Register ODDR2 for clock forwarding. It is useful for propagating a clock and DDR data with identical delay, and for multiple clock generation, where every clock load have unique clock drivers. This is accomplished by put "1" to input D0 and "0" value to input D1 of ODDR2. Input clocks to ODDR2 are two clocks which are 180 degree phase different. This clock is output of global clock buffer (BUFG). This guarantees the generated forwarding clock has a minimum amount of accumulated jitter by minimizing the logic components in the path to the clock output.

For guaranteeing the clock and data to Atlys board has the identical delay, ODDR2 registers must be used for data and data enable signal as shown in Figure 3.9. The data CM_SP_DATA_O is routed to both D0 and D1 of ODDR2 register.

The VHDCI port used for HDMI interface has 40 usable pins. Since it has not enough pin for separating 16bit-width data in and data out, the data pins are used as bi-directional. So IOBUF must be used. Since the data out (CM_SP_DATA_O) uses ODDR2, the tri-state control signal also has to use ODDR2. Otherwise, it would get errors at mapping phase when synthesizing. And one bit data out need one bit tri-state control signal as shown in Figure 3.10. So it needs 32 ODDR2 registers for 16-bit bi-directional data output.

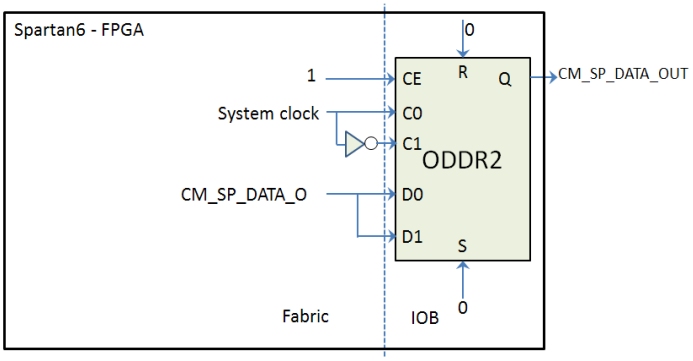

Figure 3.9 Data output to pin using ODDR2

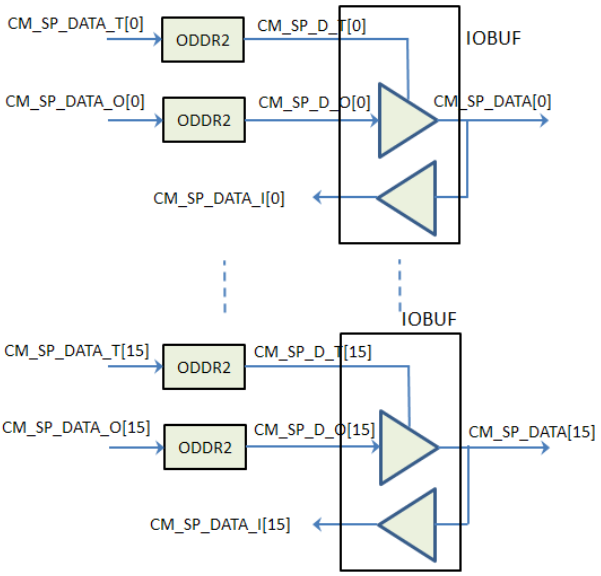

Figure 3.10 Output bi-directional data to pin using ODDR2

3.2.2 HDMI interface in SNUPEE board

Figure 3.11 shows the block diagram of CM_HDMI module in SNUPEE board.

When BT1 is pressed, line data first is written to line buffer (CM_BUFFER_IN). BT1 state machine will write line by line an HDMI frame to SDRAM.

When BT2 is pressed, BT2 state machine reads line data from SDRAM and writes to CM_BUFFER_OUT and then later transmits to Atlys board.

AHB master transactor module sends read/write requests to SDRAM through AHB Master port.

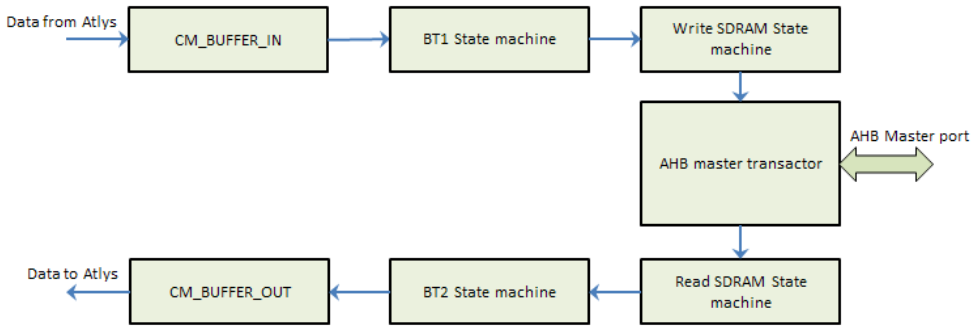

Figure 3.11 Block diagram of CM_HDMI module in SNUPEE

Explanation of BT1 state machine shown in Figure 3.12 is as follow:

BT1_IDLE: waits for write request (BT1_FROM_CM) from Atlys.

BT1_DONE_CLR: sends COMM_SIG_FROM_SP0 signal to Atlys board to indicate that BT1 operation is ready to start.

BT1_START: check whether full frame is received or not. If full frame is transferred, go to BT1_IDLE state and send “frame_wr_done” signal to processor.

BT1_WRITE: when CM_SP_WE is active low, CM_SP_DATA_I is available for writing to buffer in. When writing one line is done, state machine jump to BT1_WRITE_DONE.

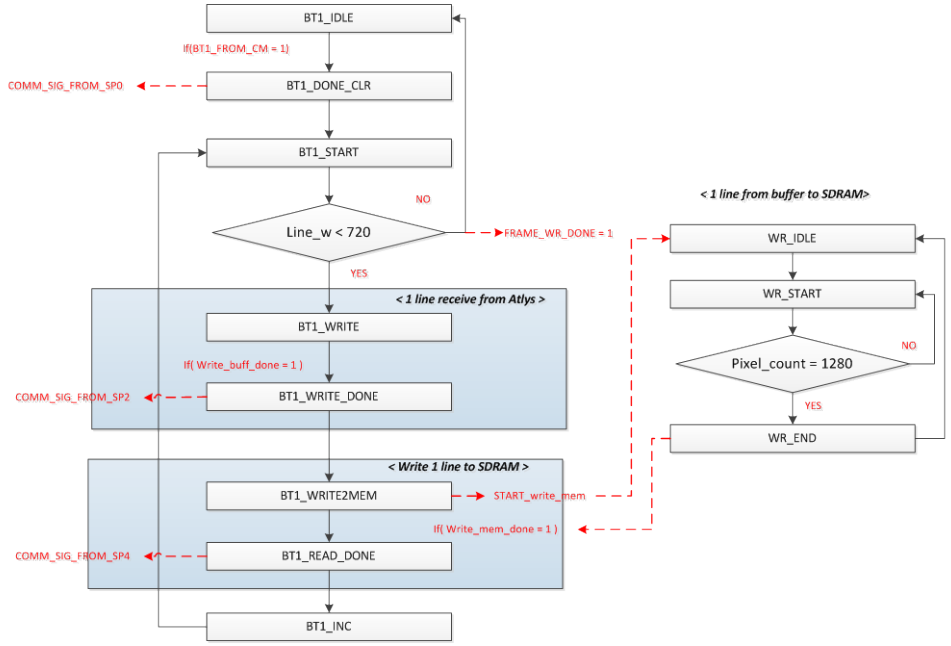

Figure 3.12 Receive state machine block diagram in CM_HDMI

BT1_WRITE_DONE: sends COMM_SIG_FROM_SP2 signal to Atlys board to notify that one line is written to SNUPEE input buffer.

BT1_WRITE2MEM: writes one line from input buffer to SDRAM.

BT1_READ_DONE: sends COMM_SIG_FROM_SP4 to Atlys to notify that one line is written to SDRAM of SNUPEE successfully.

BT1_INC: increases the line counter by one and jumps to BT1_START.

Explanation of BT2 state machine shown in Figure 3.13 is as follow:

BT2_DONE_CLR: send COMM_SIG_FROM_SP1 signal to notify Atlys that BT2 operation is ready to start.

BT2_START: Check if full frame is transferred or not, sends “frame_rd_done” signal to CM_CTRL module.

BT2_READ_MEM: read one line from SDRAM and store to CM_BUFFER_OUT.

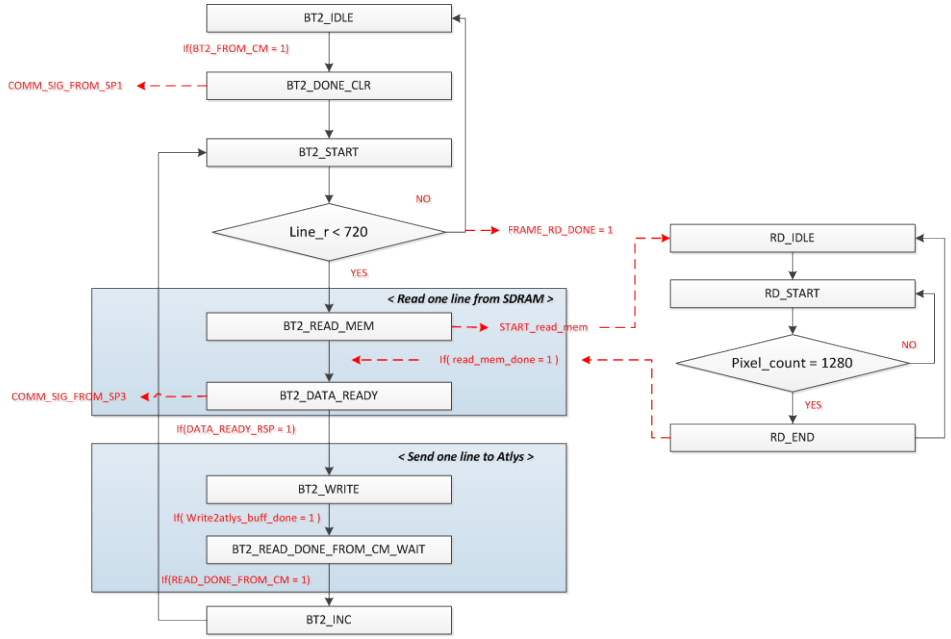

Figure 3.13 Transmit state machine block diagram in CM_HDMI

BT2_DATA_READY: send COMM_SIG_FROM_SP3 signal to Atlys that the line data is ready. If DATA_READY_RSP signal from Atlys goes “High”, it means that Atlys is ready to receive line data. Then, state machine jumps to BT2_WRITE.

BT2_WRITE: controller reads line data from CM_BUFFER_OUT and transmits to Atlys. The clock for transmitting data (DATA_CLKOUT) is same as system clock of SNUPEE. If writing to Atlys’s buffer is done, state machine jumps to next state.

BT2_READ_DONE_FROM_CM_WAIT: waits for Atlys’s writing one line to DDR2 memory by tracking READ_DONE_FROM_CM signal. Then state machine jumps to BT2_INC to increase the line counter.

3.3 Pipelined HDMI interface

Time for accessing data to SDRAM in single-beat AHB burst mode is quite slow (Tx3 or Rx1). So for effectively using pipelined technique, it is needed to reduce

time to access SDRAM. In proposed design, the CM_HDMI module supports full 16-beat AHB burst mode to significantly speed up external memory access time. To enable the parallelism, the proposed design uses double input buffer and double output buffer for both Atlys and SNUPEE board and schedules for transmit path as in Figure 3.14 and Figure 3.15.

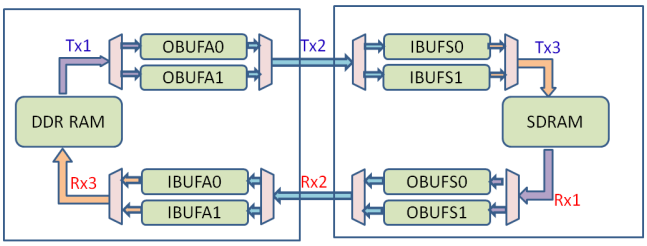

Figure 3.14 Pipelined scheme of HDMI interface

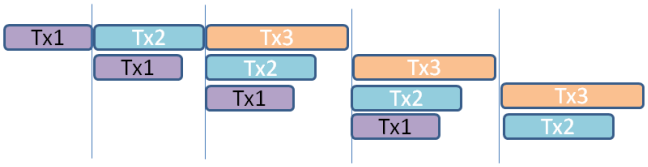

Figure 3.15 Scheduling for pipelined HDMI interface

Timing for transmitting one line video is now the maximum value of the three values: Tx1, Tx2 and Tx3. The scheduling for receiver path is similar.

Based on the above idea, the pipeline scheme for HDMI interface in Atlys board and SNUPEE board are described in section 3.3.1 and 3.3.2, respectively. Two new pins are added to board interface in pipeline scheme which are: COMM_SIG_FROM_SP4_CLR and COMM_SIG_FROM_SP5. These pins are used to guarantee that state machines in both Atlys and SNUPEE board complete for current line data to confirm the correctness of pipeline scheme.

3.3.1 Pipelined HDMI interface in Atlys board

As discussed above, the input line buffer and output line buffer of Atlys are doubled to enable parallelism.

The pipelined scheme for sending path is depicted in Figure 3.16. The sending path mainly includes two stages: Read one line data from DDR2 memory and writes to output buffer (FSM 1), send one line data from output buffer to SNUPEE (FSM2). The pipelined scheme utilizes double buffers so that two stages can process in parallel.

It can be seen in the figure that the state CM_SP_WRITE_START of FSM1 sends "start" signal to FSM2 at the start of line transfer. However, FSM2 starts at one line data later than FSM1. Both state machines can start new line data transfer only if they receive the COMM_SIG_FROM_SP4 from SNUPEE indicating that the current line data has been written to SDRAM of SNUPEE. Then Atlys sends responding signal COMM_SIG_FROM_SP4_CLR to SNUPEE so that SNUPEE can start process for new line data.

Similarly, Figure 3.17 presents the pipelined schedule for receiving path (BT2 operation). The receiving path in non-pipelined scheme has two stages: receives one line data from SNUPEE and writes to input buffer (state machine FSM2), reads one line data from input buffer and writes to DDR2 memory (state machines FSM0 and FSM1). The input buffer also is also doubled to enable parallelism. Now, two stages can process in parallel because they alternatively access different input buffer. The state machine FSM1 starts and ends one line data later than FSM2. The start of FSM1 is triggered by BT2_START signal from state CM_SP_READ_START of FSM2 .The condition "count720_v > 0" makes the FSM1 starts one line data later than FSM2. Both FSM1 and FSM2 wait for READ_DONE_FROM_HDMI_OUT signal from FSM0 state machine indicating that the current line data has been written to DDR2 memory successfully. Then FSM2 sends READ_DONE_FROM_CM signal to SNUPEE indicating that Atlys has done and waits for COMM_SIG_FROM_SP5 from SNUPEE meaning that SNUPEE have also completed. After that, both state machines in Atlys and SNUPEE can start process for new line.

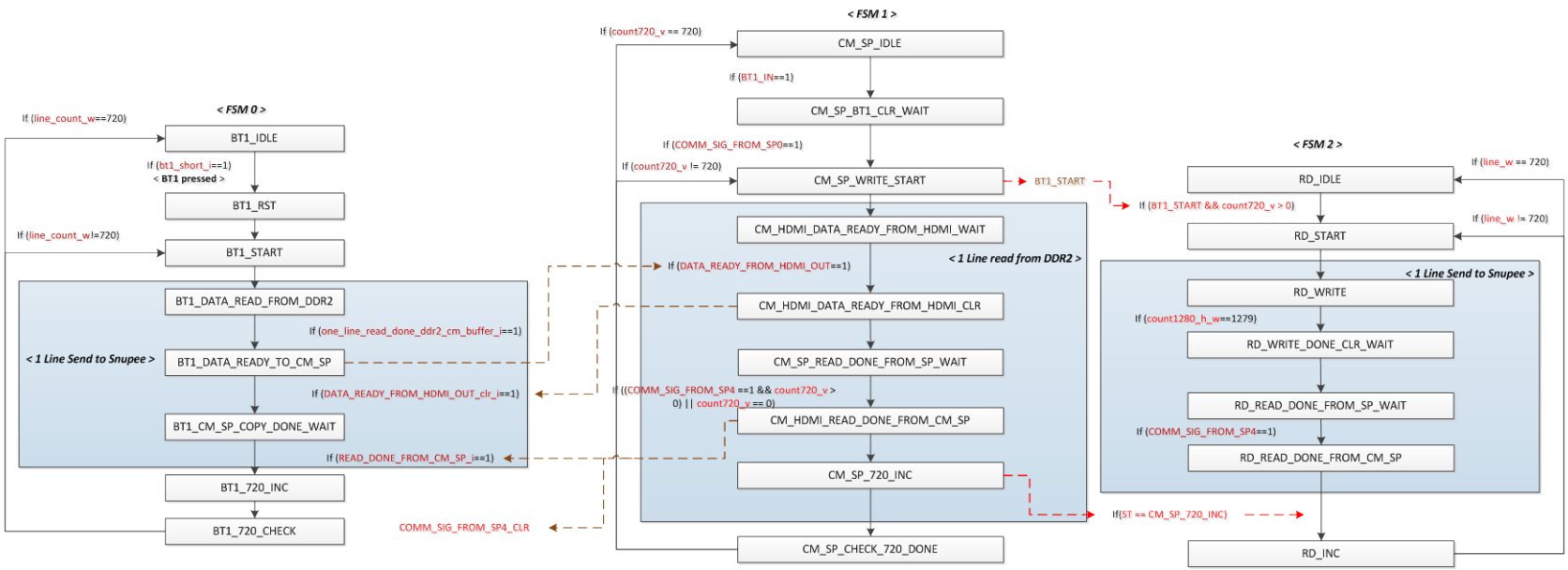

Figure 3.16 Pipelined transmitter in Atlys board

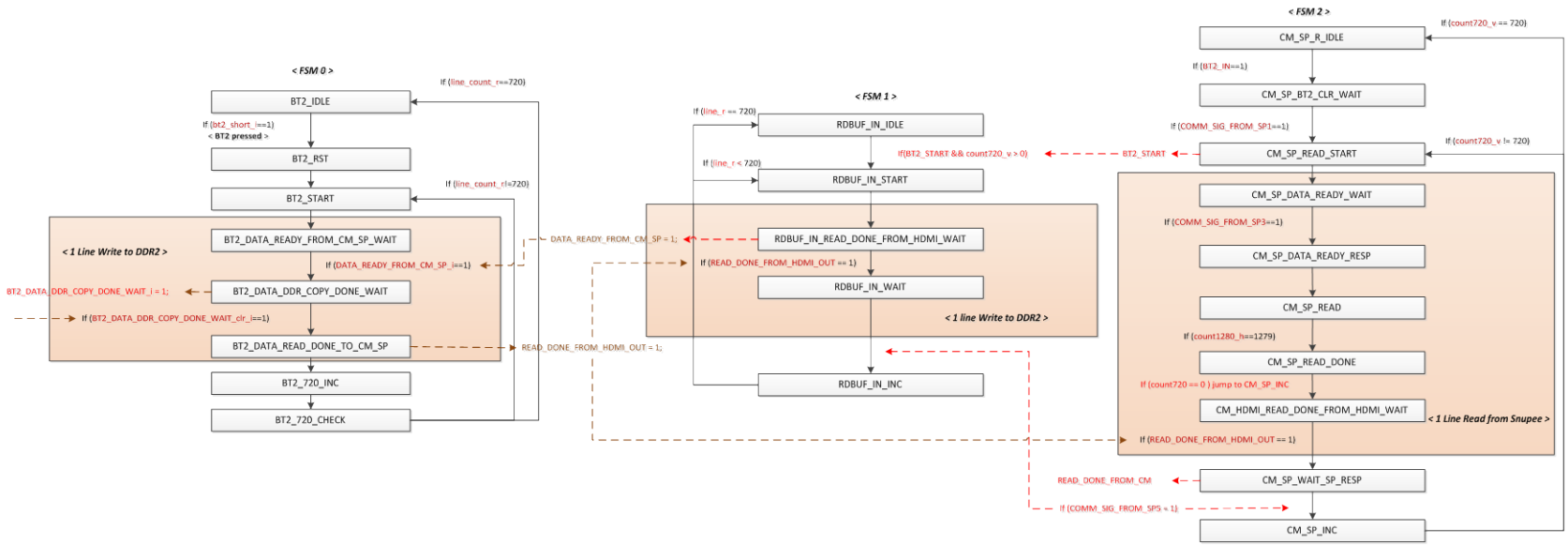

Figure 3.17 Pipelined receiver in Atlys board

3.3.2 Pipelined HDMI interface in SNUPEE board

In non-pipelined scheme, the single input buffer or single output buffer are used. So, the current stage cannot start until the previous stage completes. To improve the speed of the interface, as discussed above for Atlys, input buffer and output buffer are also doubled so that two stages which previously are in order can process in parallel because they access the different buffers.

Figure 3.18 is the pipelined scheme for transmit path (BT1 operation). The state machine (in Figure 3.12) in non-pipelined scheme now is broken into 2 parallel state machines. It can be seen that two stages: "receive 1 line from Atlys" (FSM 0) and "Write 1 line to SDRAM" (FSM 1) are now processing in parallel. FSM0 receives line data from Atlys and writes to the first input buffer. Meanwhile, the FSM1 reads line data from the second buffer and writes to SDRAM. FSM1 starts one line later than FSM0 and ends one line later than FSM0. Two state machines can only start a new line data if both of them complete processing for the current line data and receive COMM_SIG_FROM_SP4_CLR signal from Atlys meaning that Atlys's state machines have also completed for current line data.

Similarly, the pipelined state machine for receive path (BT2 operation) is drawn in Figure 3.19. The non-pipelined scheme has two stages: Read line data from SDRAM and send line data to Atlys board. Now, the state machine in Figure 3.13 is broken down into 2 parallel state machines: FSM0 and FSM1. FSM0 reads one line data from SDRAM and writes to an output buffer. Meanwhile, the FSM1 reads one line data from another output buffer and sends to Atlys board. At the start-up stage, FSM0 starts first and then FSM1 processes one line later. If both of them complete processing for the current line data, state RD_READ_DONE_FROM_CM_CLR of FSM1 sends COMM_SIG_FROM_SP5 signal to Atlys to notify that SNUPEE has completed for current line. After that, both state machines in Atlys and SNUPEE can start process for new line.

Table 3-1 shows the performance of non-pipelined scheme and pipelined scheme HDMI interface with:

Transmit bandwidth: 150MB/s

Receive bandwidth: 100MB/s

Frame size: 1280x720x16bit (1.8 MB)

Table 3-1 Performance of HDMI interface

Scheme	**Tx (frames/s)**	**Rx (frames/s)**	**Tx Throughput MB/s**	**Rx Throughput MB/s**
Non-pipelined scheme	25.3	22.0	45.54	39.60
Pipelined scheme	67.1	53.5	120.78	96.30

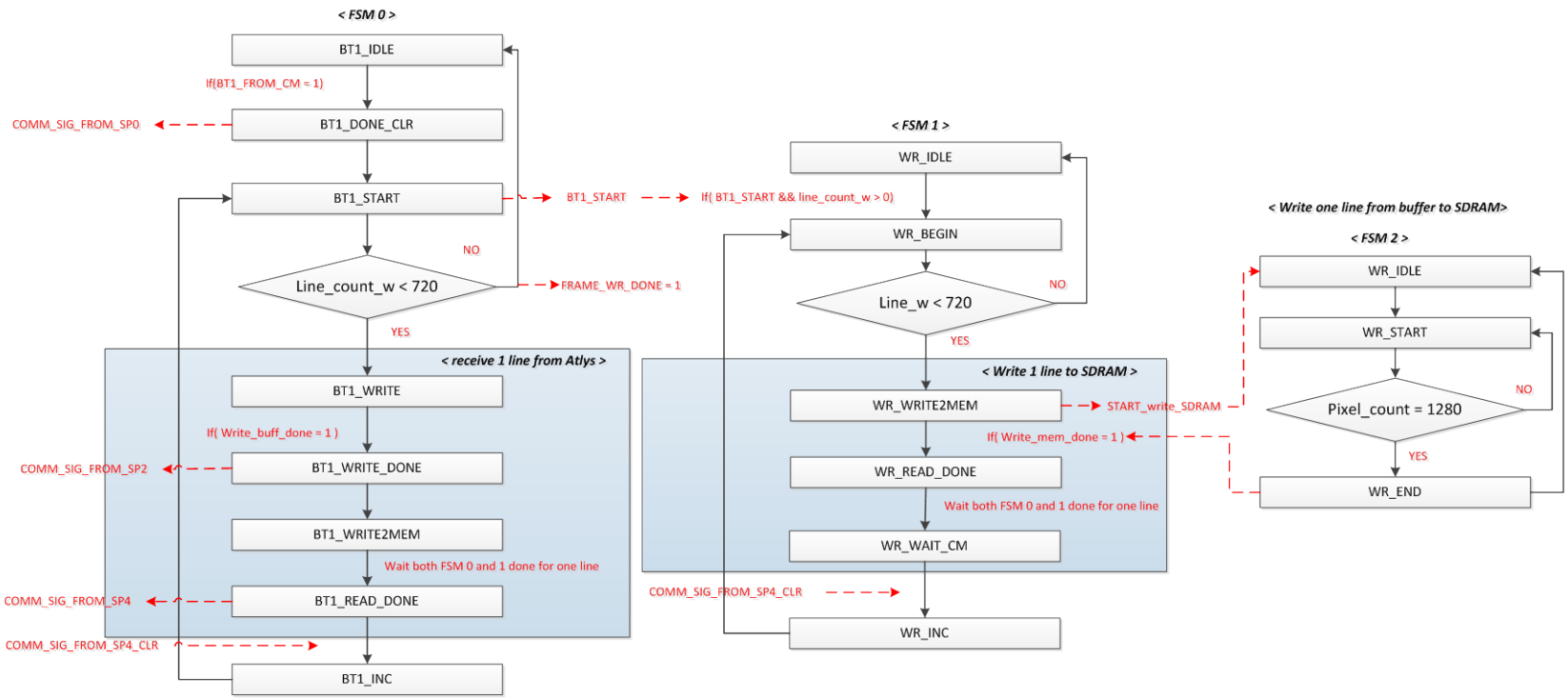

Figure 3.18 Pipelined scheme for BT1 state machine in SNUPEE

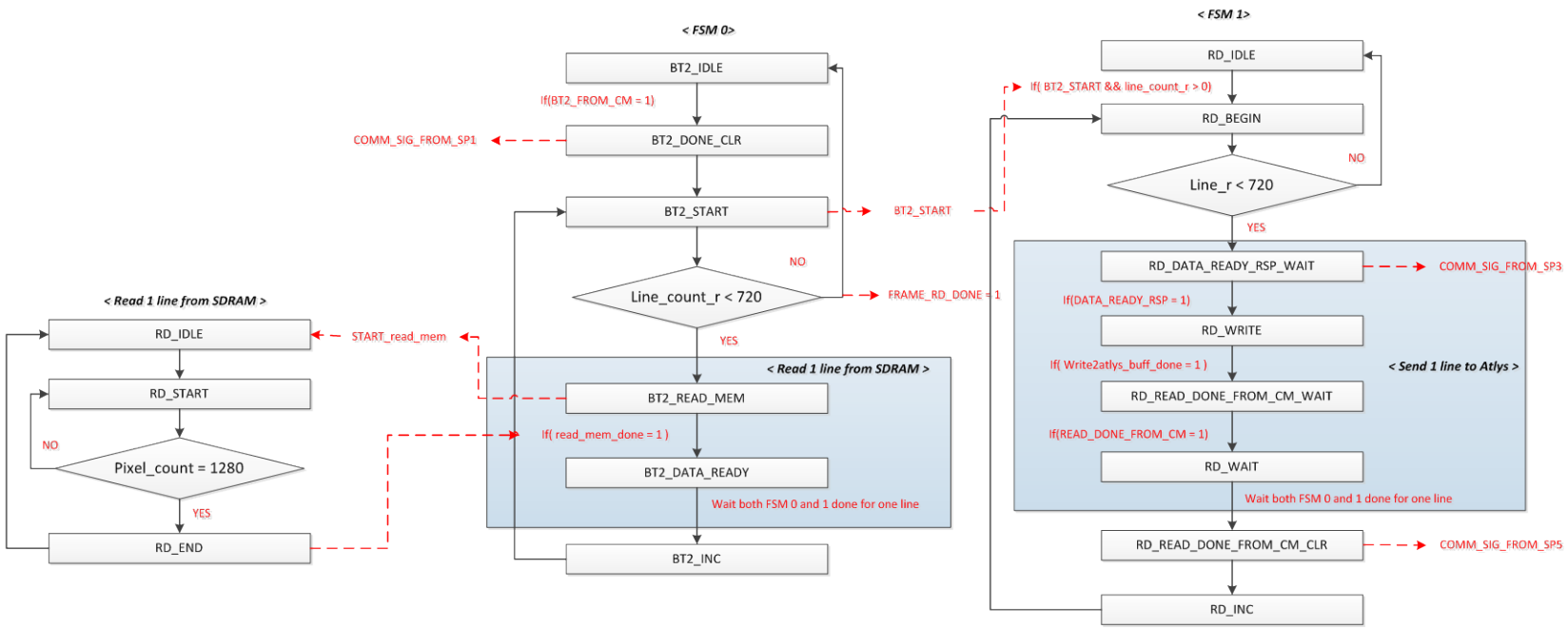

Figure 3.19 Pipelined scheme for BT2 state machine in SNUPEE

4 Controller design for NAND flash memory

4.1 Architecture of the NAND flash controller

The architecture of hardware controller for massive NAND flash is shown in Figure 4.1. The flash controller includes 3 main blocks: Direct Memory Access (DMA), Control and status, and Main controller.

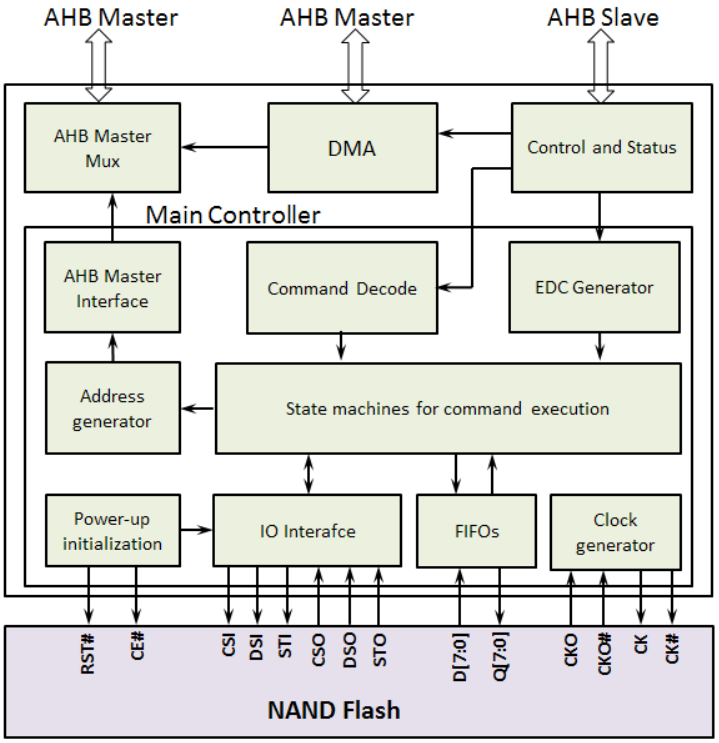

Figure 4.1 Design of NAND Flash controller

Direct Memory Access block: directly accesses to external memory and internal memory to reduce processor's work load and increase transfer speed.

Control and status block: The configuration and operation of the NAND flash controller is controlled by the host processor through the control and status registers inside of this block. It translates the request command from host processor to

control signals for state machines in main controller and DMA to operate. The status registers also provide DMA status and operating status of the flash interface such as Busy and Ready signals.

Main controller block: It is responsible for device initialization, generation of the NAND flash device access sequence. It supports all the basic commands of NAND device so that the driver in host processor can execute whatever multi-command requests like two plane operation, simultaneous read/write operation, etc.

The main controller block has the following blocks:

1. Power-up Initialization: initializes the device in a pre-defined manner to make it operate properly.
2. Command Decode: translates the request command from host processor to execute the corresponding state machine.
3. EDC generator: generates Hamming Error Correction Code the for command packet.
4. State machines for command execution: each command has its own state machine. Since there are some group of command that have the same command sequence, PAGE READ (DA & 0Xh) and BLOCK ADDRESS INPUT (DA & 8Xh) for example, to save hardware resource, the commands in same group uses the same state machine.
5. Clock generator: generates the differential clocks to flash and assures that these clocks are aligned with data, control signal to flash when a system reset occurs.
6. IO interface: outputs control signals to flash and synchronizes input signals from flash to FPGA board.
7. FIFO: provides interface for transferring data between flash and FPGA.
8. Address generator: generates addresses to read or write buffer through an AHB master port.

When the host processor requests a write command, DMA copies data from SDRAM and writes to buffer (internal memory). Then the main controller starts to read data from internal memory and program to NAND flash. Similarly, when the

host processor requests a read command, the main controller reads data from NAND flash and writes to internal memory. After that, DMA core copies data from internal memory and writes to SDRAM. In the next section, the operation of NAND flash controller is explained in more detailed.

4.2 Operation of NAND flash controller

4.2.1 Power-up Initialization process for NAND flash

In order to properly initialize a device, the following power-up initialization procedure shown in Figure 4.2 is required. The description of state machine is as following:

PLL IDLE: On power-up, the system controller held RST# low to keep all the NAND flash devices in reset mode while the power stabilizes and the devices prepare themselves for operation. RST# will be held low by controller for a minimum of t_{RSTL} period of time (minimum reset time) after the input voltage stabilizes. While RST# is being held low, all finite state machines in the NAND Flash memory devices are initialized, and any configuration and status registers are reset to their default state.

PLL IDLE1: waits for the hold time (t_{CRH}) of chip enable signal (CE#) after the falling edge of RST# is satisfied.

PLL LOCK: asserts CE# to low to start the PLL locking procedure and waits for t_{LOCK} period of time. This period depends on the number of flash device in the link.

SET DEV ADDR0 to SET DEV ADDR7: after PLL of all devices in the link have been locked successfully, the controller sends down the link a special one byte Read Data Packet, with initial value “00h” on D0. Each NAND flash memory device, being aware that this first packet is special, takes the value it receives as its unique device address. It also increments the value with a serial adder as the packet flows through the device to form the Device Address of the next downstream device.

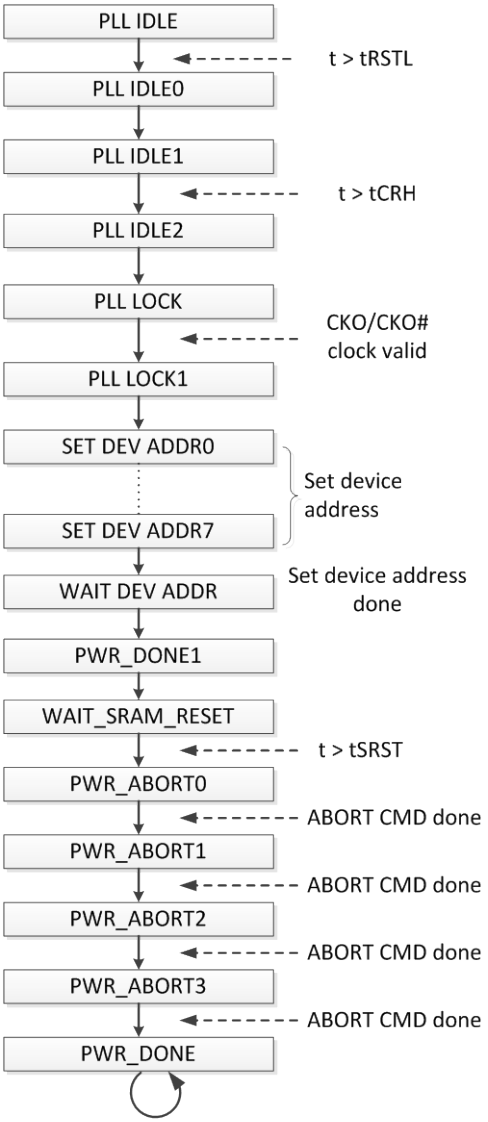

Figure 4.2 Power-up initialization state machine

WAIT DEV ADDR: waits setting device address for all devices is done. The value in the special Device Address Read Data packet when it returns back to the controller is the number of devices in the link.

WAIT_SRAM_RESET: NAND flash needs t_{SRST} pediod of time starting from the locking of the PLL to reset all the SRAMs inside if it.

WAIT_ABORT0, WAIT_ABORT1, WAIT_ABORT2, WAIT_ABORT3: after SRAMs are reset, the controller should immediately broadcast an OPERATION ABORT (FFh & CXh) command to all the device on the link. As each ABORT command is bank specific, total of 4 broadcast ABORT commands are required in order to reset all 4 banks in each device. And then, each device will be busy for a maximum of 5ms after ABORT command. During this time period, the acceptable command is READ STATUS REGISTER only.

If RST# goes "Low" (system hard reset) during the normal device operation, the system controller would performs the "Power-up Initialization" procedure again. NAND Flash memory must be powered up and initialized in a predefined manner. Operational procedures other than those specified may result in undefined operation.

4.2.2 Operation Abort

The OPERATION ABORT (DA &CXh) is used to put the memory bank into a known condition and to abort a bank operation already in progress. Figure 4.3 describes the sequence of ABORT (DA & CXh) command.

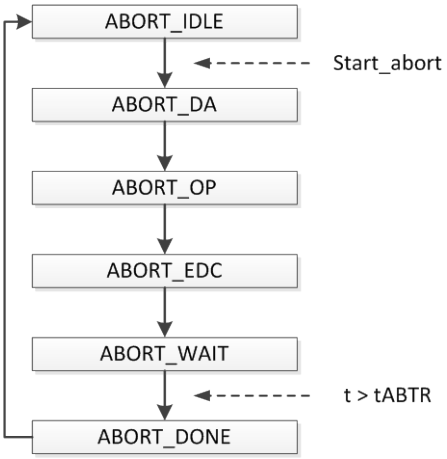

Figure 4.3 Abort command state machine

ABORT_IDLE: waits for request from host processor or power-up initialization.

ABORT_DA: passes the device address to flash. In case of broadcast command, device address is FFh.

ABORT_OP: the operand of ABORT command. Since it is bank specific, the operand is CXh while X is the bank number.

ABORT_EDC: if EDC feature is enabled, it passes the Error Detection Code for this command, otherwise, the default value 00h is needed.

ABORT_WAIT: waits for t_{ABTR} to reset registers inside of device. After this time, controller can issue READ STATUS REGITER (DA & F0h) to check the status of the bank or monitor the output of STO (Status OUT) pin to check whether the bank is ready or not.

The maximum or minimum Operation Abort time depends on the status of the device: being read, programmed or erased.

4.2.3 Read Status Register (DA & F0h) and Read Plane Status Register (DA & F1h)

NAND has a 3-byte status register that system controller can read the current status of the device: “Read / Busy” and “Pass / Fail” status for each LUN of each bank. Two commands use the same command sequence shown in Figure 4.4.

RDST_IDLE: waits for read status request from host processor.

RDST_DA: device address of device. If use broadcast command, the special device address is FFh.

RDST_OP: Operand of command: F0h for Read Status Register command and F1h for Read Plane Status.

RDST_EDC: Error Correction Code for the command.

RDST_CDS: Separation time for Command Strobe Input and Data Strobe Input.

RDST_DATA: Activate the Data Strobe Input signal to flash device.

RDST_DATA1: Data Strobe Output active, and status information is available on data out port of flash. This data is stored in internal memory for reading by processor.

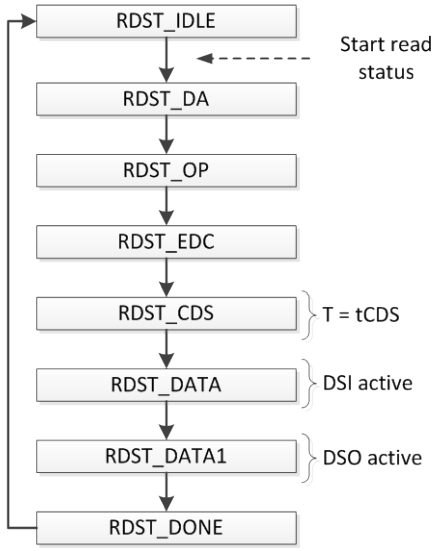

Figure 4.4 Read Status and Read Plane Status Register State machine

4.2.4 Read Registers (DA & F2h)

All registers inside of flash device are read using the READ REGISTER (DA & F2h) command as shown in Figure 4.5.

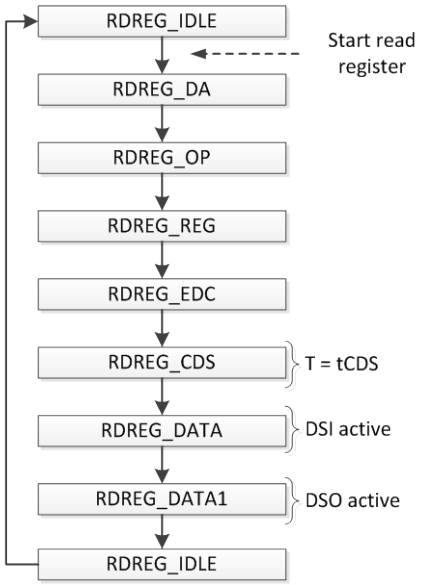

Figure 4.5 Read Register State machine

It begins with a device address followed by the READ REGISTER command word (F2h), register ID, and concluded with the Error Detection Code for a total of four bytes. After the Command Strobe Input is active, controller needs to wait for t_{CDS} (Command Strobe Input and Data Strobe Input Separation) before activating DSI signal. One clock cycle later, the Data Strobe Output from flash device is active indicating that the register values are available on the output port of flash device. These register values are stored in internal memory for reading by host processor.

4.2.5 Write Register (DA & F3h)

The state machine of Write Register command shown in Figure 4.6 is described as below:

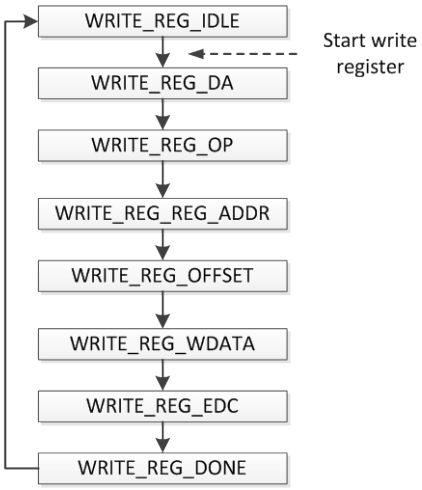

Figure 4.6 Write Register State Machine (DA & F3h)

WRITE_REG_DA: the device address of flash device.

WRITE_REG_OP: command word of Write Register (F3h)

WRITE_REG_REG_ADDR: the register ID. Registers in flash device include Device Information Register (ID: 02h), Link configuration Register (ID: 01h) and Device Configuration Register (ID: 03h).

WRITE_REG_OFFSET: Each register above contains multi-byte of data. For each specific write, the byte offset is needed.

WRITE_REG_WDATA: The data is being written.

WRITE_REG_EDC: Error Correction Code of the command packet.

The user controller can issue the WRITE DEVICE CONFIGURATION REGISTER command to set the virtual page size for read in each bank for maximum system performance and set the internal frequency divide ratio (FDR) to change the clock frequency to internal NAND dies.

4.2.6 Page Read Operation

4.2.6.1 Page Read command (DA & 0Xh)

The state machine for Page Read command shown in Figure 4.7 is described as below:

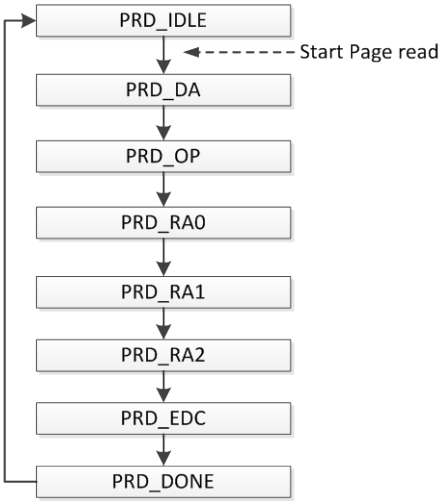

Figure 4.7 Page Read State Machine

PRD_IDLE: waits for page read request from host processor.

PRD_DA: device address of flash device.

PRD_OP: command word of Page Read (0Xh) where X is the bank number.

PRD_RA0: the LSB byte of row address.

PRD_RA1: the second byte of row address.

PRD_RA3: the MSB byte of row address and the virtual page address if necessary.

PRD_EDC: the Error Detection Code for command packet.

PRD_DONE: passes command to flash device done.

Once the internal Page Read operation starts, the full page of data from page location in NAND memory bank are read to physical page buffer and the selected data is transferred to virtual page buffer. After a virtual page is transferred by an initial Page Read command, a Burst Data Read (DA & 2Xh) command may access the same virtual page within the same physical page range to get desired data. The command sequence of Burst Data Read command is presented in the next section.

4.2.6.2 Burst Data Read (DA & 2Xh)

Figure 4.8 is the state machine block diagram of BURST DATA READ (DA & 2Xh) command.

BDRD_DA and BDRD_OP are device address and command word (2Xh) to flash device where X is the bank number.

BDRD_CA0, BDRD_CA1, BDRD_CA2: columns address and a virtual page address, if necessary, so that the data in the virtual page buffer is read starting from the column address provided with in the selected virtual page.

BDRD_EDC: Error Detection Code for the command packet.

BDRD_CDS: controller needs to wait for t_{CDS} (CSI and DSI separation time) before activating Data Strobe Input pin.

BDRD_DATA: Data Strobe Input is active. The length of DSI is equal to the number of byte data the controller want to read.

BDRD_DATA1: Data Strobe Output from flash device is active means that data is available on output port of flash device. The length of DSO signal is equal to the length of DSI signal. The controller writes this data to the 4-byte-depth FIFO as soon as it is available. Each time the FIFO is full, the word data is read from FIFO and written to internal memory. This operation is continuous until all byte data is written to internal memory.

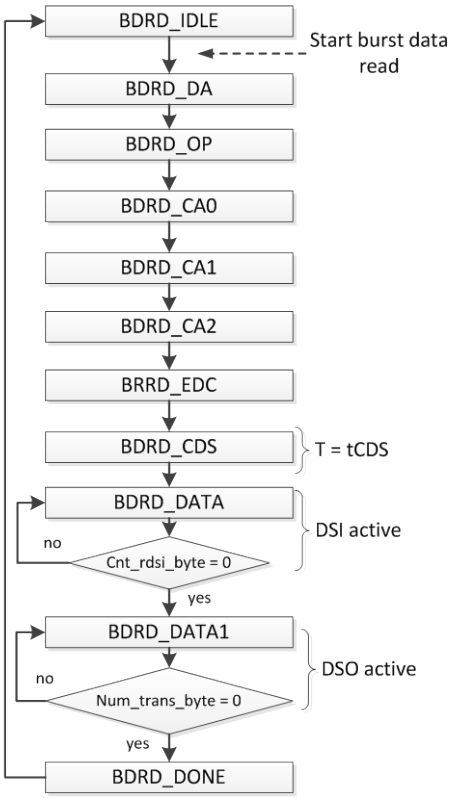

Figure 4.8 Burst Data Read State Machine

4.2.7 Program Operation

The program operation is the sequence of commands as follows:

1. One BURST DATA LOAD START (DA & 4Xh) command with data
2. Zero or more BURST DATA LOAD (DA & 5Xh) commands with data to other column address within the same page.
3. One PAGE PROGRAM (DA & 6Xh) command to transfer data in virtual page buffer to physical page buffer, on the flash chip, and programs into the bank's selected page.

NAND flash provides only one virtual page size choice for programming data which is full page program. It's not possible to program data on a basis of virtual page sizes smaller than the full page size.

4.2.7.1 Burst Data Load Start (DA & 4Xh)

The BURST DATA LOAD START (DA & 4Xh) command is used to begin loading data into a given bank's fresh virtual page buffer. This data load command must be the first data load command issued to a bank's fresh virtual page buffer. The state machine of BURST DATA LOAD START is shown in Figure 4.9. The state description is as below:

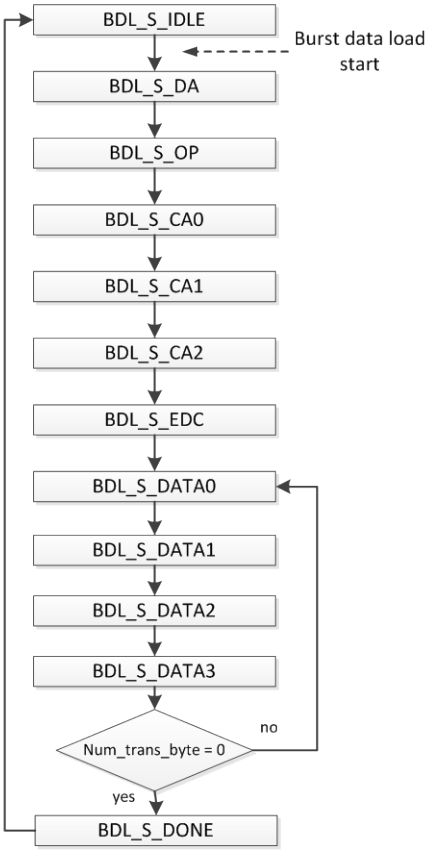

Figure 4.9 Burst Data Load Start command State Machine

BDL_S_IDLE: waits for Burst Data Load Start request from host processor

BDL_S_DA: the target device address

BDL_S_OP: command word: 4Xh where X is the bank number

BDL_S_CA0, BDL_S_CA1, BDL_S_CA2: the column address in virtual page buffer that the data is written starting from.
BDL_S_DATA0, BDL_S_DATA1, BDL_S_DATA2, BDL_S_DATA3: 4 bytes of a word data read from internal memory and transfer to flash device. At BDL_S_DATA3 state: if all the byte data is transferred to flash device, state machine jumps to BDL_S_DONE, otherwise, it jumps to BDL_S_DATA0 to continue data transfer.

4.2.7.2 Burst Data Load (DA & 5Xh)

After initial data is input with the BURST DATA LOAD START (DA & 4Xh) command, additional data may be written to a new column address within the same page using BURST DATA LOAD (DA & 5Xh) command. The BURST DATA LOAD command can be issued any number of times to the same page address prior to issuing the PAGE PROGRAM (DA & 6Xh) command. The command sequence of BURST DATA LOAD (DA & 5Xh) is identically same as BURST DATA LOAD START (DA & 4Xh).

4.2.7.3 Page Program (DA & 6Xh)

The command sequence of PAGE PROGRAM (DA & 6Xh) is the same as that of PAGE READ (DA & 0Xh) as shown in Figure 4.7 which are, in order, device address, command word (6Xh), 3-byte row address and a EDC byte. To check the status of PAGE PROGRAM command, the host controller monitors the STO output signal from flash device. Once the host controller detects the STO pulse two times meaning that the PAGE PROGRAM command has completed, it may issue the READ STATUS (DA & F0h) to check the detailed status of command.

4.2.8 Page Copy Operation

The PAGE COPY operation is used to quickly and efficiently transfer data stored in one page of a bank to another page in the same bank without reloading data when

there is no bit error in the stored data. In the case of a page failure, the whole block must be considered as invalid so the Page Copy operation is particularly useful as all the valid data stored in the other pages of the same block must be copied into the newly assigned block. The Page Copy operation is also very useful for Garbage Collection, where it is necessary to defragment the memory array to optimize the allocation of the storage resources. To start Page Copy operation, the host processor may issue the following command sequences.

1. The PAGE READ FOR COPY (DA & 1Xh) is issued first. The state machine of this command is the same as PAGE READ (DA & 0Xh) command.
2. The BURST DATA READ (DA & 2Xh) command can be issued in order to check bit error of page data.
3. The BURST DATA LOAD (DA & 5Xh) command is issued to modify the copied data if bit error is detected.
4. The PAGE PROGRAM (DA & 6Xh) command is issued to program modified data (error corrected) to the new destination page. After issuing this command, once host processor detects 2 consecutive STO pulse, it may issue the READ STATUS REGISTER (DA & F0h) command or BROADCAST READ STATUS REGISTER (FF & F0h) to check the detailed status.

4.2.9 Erase Operation

The erase operation is the sequence of two commands:

1. BLOCK ADDRESS INPUT (DA & 8Xh)
2. ERASE (DA & AXh)

4.2.9.1 Block Address Input (DA & 8Xh)

The state machine of BLOCK ADDRESS INPUT (DA & 8Xh) command is the same as PAGE READ (DA & 0Xh) command shown in Figure 4.7. Its requires

three bytes of row address for selection of block to be erased (Page address RA[6:0] is ignored). When all address information for the block to be erased is loaded, the ERASE (DA & AXh) command must be issued to start the internal erase operation.

4.2.9.2 Erase (DA & AXh)

The state machine of ERASE (DA & AXh) command is described in Figure 4.10. ERASE_DA, ERASE_OP and ERASE_EDC are target device address, command word (AXh) and EDC byte, respectively. If the host controller detects the falling edge of STO signals two times, it may issue the READ STATUS REGISTER (DA & F0h) or the BROADCAST READ STATUS REGISTER (FF & F0h) to check detailed status of command. If an error is found, host controller maps out the block as bad and replaces it another good block.

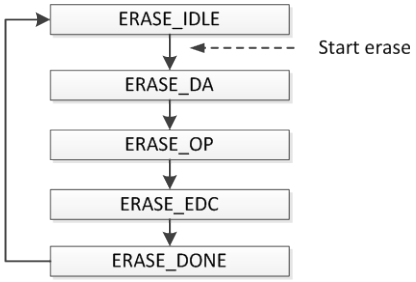

Figure 4.10 Erase Command State Machine

4.3 Feature of NAND flash controller

- Simultaneous Read & Write operation in case of daisy-chain ring topology
- Support Error Detection Code (EDC) for Command Packets and Register Read Packets
- Highly Flexible Modular Command Structure
- Supports concurrent Multi-Bank Operations: Simultaneous Read & Write in different bank, page program while block erase, page copy while block erase, block erase while page program.

- Supports two-LUN per Bank Operations
- Supports two-Plane per LUN Operations
- Broadcasting commands
- Supports bad block management
- Daisy-chain cascade up to 255 number of linked devices to increase memory capacity with undiminished data throughput and the pin configuration at the top level is identical to one of single device (as in Figure 4.11). Since single device capacity is 256Gb/512Gb, the daisy-chain topology will give us nearly unlimited storage for every application. Daisy-chain topology also enhances parallelism by issuing concurrent multi-bank, multi-chip operations to increase system throughput.
- Currently, does not support Error Correction Code function.
- Runs at maximum frequency of 219.46 MHz.

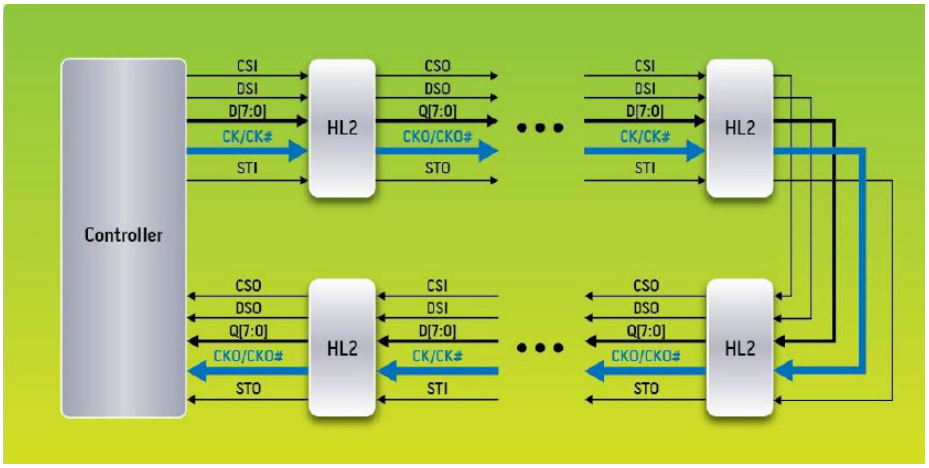

Figure 4.11 Daisy-chain configuration

As shown in Figure 4.11, the daisy-chain configuration requires echo pins for control signals:

- CSO: Echo pin output of CSI
- DSO: Echo pin output of DSI

- STO: Echo pin output of STI

All signals including input data and control stream down from the first device to the last device and end up at the controller. Each device has unique device address which is automatically set during power-up initialization. The address of first device is “00h”, second device is “01h”, and device address for the rest of devices can be set with same manner. Special device address “FFh” is used for broadcasting command executing command for all devices in the link. When controller wants to access a specific device, it has to pass the device address at the start of command sequence.

Table 4.1 shows the hardware cost of NAND flash controller. It uses 128 Kilobyte of memory buffer to support concurrent 4-bank operations in double chip (concurrent 8-bank operations) since the full virtual page size for write and read is 8832 bytes. When processor issues 4-bank PAGE PROGRAM command, for example, four data pages being written to flash are copied from DRAM to memory buffer by using DMA module. After that, controller executes four consecutive PAGE PROGRAM commands to concurrently write four pages to desired location in four different banks. The process to copy four pages of data from buffer to virtual page is sequential but four PAGE PROGRAM commands are simultaneous. Concurrent 4-bank PAGE READ command is similar. Controller executes four PAGE READ commands to read data from flash to memory buffer. Later, processor will sequentially copy four pages from buffer to DRAM.

Table 4-1 Synthesis result of flash controller

Logic Utilization	**Used**	**Available**	**Utilization**
Slice Registers	994	948480	0%
Slice LUTs	1183	474240	0%
Fully used LUT-FF pairs	642	1535	41%
Block RAM/FIFOs	32	720	4%

The controller is synthesized and tested using Xilinx Virtex-6 FPGA (XC6VLX760-1FF1760) in SNUPEE board.

The controller is verified in FPGA with single NAND flash chip and double chip in a daisy-chain configuration. A demo video of the whole system has been made supporting concurrent 4-bank operation in double chip. The recording speed of the whole system is 6.6 HD 720p frames/s and displaying speed is 8.7 720p HD frames/s. The throughput of whole system could be increased significantly by increasing the clock speed to flash memory. Currently, the SNUPEE clock speed is only 50MHz, so the frequency of clock to NAND flash is only 25MHz while the maximum clock frequency to flash can be 400MHz.

References

[1] MOSAID, “DDR800 Arrives”, *MOSAID presents at Flash Memory Summit 2011*, August 2011

[2] Peter Gillingham, Jin-Ki Kim, .et al, “256Gb NAND Flash Memory Stack with 300MB/s HLNAND Interface Chip for Point-To-Point Ring Topology”, *IEEE IMW*, May 2011

[3] MOSAID, “Enabling TB Class and GB/s Performance with HLNAND”, *MOSAID presents at NVRAMOS11*, Korea, November 2011

[4] MOSAID, “A 2/4 TB SATA3 SSD Employing a Single Controller”, *MOSAID presents at Flash Memory Summit 2012*, August 2012

[5] Roland Schuetz, HakJune Oh, et al,“HyperLink NAND Flash Architecture for Mass Storage Application”, *IEEE NVMSW*, August 2007.

[6] Hitachi, Toshiba, .etc, “High-Definition Multimedia Interface Specification”, Ver 1.3, June 2006.

[7] Xilinx, “Implementing a TMDS Video Interface in the Spartan-6 FPGA”, December 2010.

[8] Xilinx, “Video Connectivity Using TMDS I/O in Spartan-3A FPGAs, June 2011.

[9] Digilent, “Atlys board reference manual”, February 2011.

[10] Xilinx, “Virtex-6 FPGA SelectIO Resource User Guide”, June 2013.

[11] Xilinx, “Virtex-6 FPGA Clocking Resource User Guide”, November 2013.

[12] Xilinx, “Virtex-6 FPGA Datasheet”, May 2013.

[13] Xilinx, “Spartan-6 FPGA SelectIO Resource User Guide”, February 2013.

[14] Xilinx, “Spartan-6 FPGA Clocking Resources User Guide”, June 2013.

[15] Jim Cooke, “The Inconvenient Truths of NAND Flash Memory”, *presents at Flash Memory Summit 2007*, August 2007.

Abstract

A controller design for massive flash storage with an HDMI interface

Nguyen Duy Thanh

Electrical and Computer Engineering

The Graduate School

Seoul National University

HDMI 프로토콜은 고해상도, 고화질, 다채널 오디오, 지능적인 포맷과 제어어의 지원으로 인해서 점점 더 많은 전자제품 분야에서 사용되고 있다. 본 논문에서는 2 개의 FPGA 간 무압축 영상 전송을 위한 고속 파이프라인 HDMI 인터페이스 설계 디자인에 대한 내용을 다루었으며, 대용량 영상 저장을 위한 NAND 플래시 제어기 또한 다루었다. 구현한 제어기는 소프트웨어와 하드웨어의 조합의 장점을 이용하여 Page Program/Page Read 와 같은 기본적인 동작이 아닌 복잡한 기능을 수행한다. 이를 통해서 제어기의 유연한 플래시 메모리 제어가 가능하며, 동시에 제어기의 하드웨어 비용을 줄일 수 있다. 또한 제안한 제어기는 별도의 하드웨어 비용 증가와 데이터 처리량의 감소 없이 데이지 체인으로 연결된 최대 255 개의 장치를 지원하며 멀티 뱅크 연산의 동시 수행을 지원하여 시스템의 성능을

향상시켰다. 전체 시스템의 검증을 위해서 HDMI 인터페이스와 NAND 플래시 제어기가 구현된 시스템에서 영상 녹화와 출력을 시연하였다. 추가적인 속도 향상과 저장 용량 확장을 위해서 멀티 채널 제어기 설계가 구현될 수 있으며 NAND 플래시 메모리의 안정성을 높이기 위해서 multi-bit 오류 정정 부호 하드웨어 블록이 추가될 필요가 있다.

Keywords—HDMI interface, non-pipelined, pipelined, NAND flash controller, daisy-chain, multi-bank

Student number: 2012-23947