K.Shiva Kumar, P.Deepthi / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com
Vol. 3, Issue 1, January -February 2013, pp.1005-1010

Implementation of Complex interface bridge for LOW and HIGH bandwidth Peripherals Using AXI4-Lite for AMBA

K.SHIVA KUMAR, P.DEEPTHI
BSIT,JNTUH,Chevella
SDGI,JNTUH,IBP

Abstract—The Advanced Microcontroller Bus Architecture (AMBA) is a widely used interconnection standard for System on Chip (SoC) design. In order to support high-speed pipelined data transfers, AMBA 4.0 supports a rich set of bus signals, making the analysis of AMBA-based embedded systems a challenging Proposition. The goal of this paper is to synthesize and simulate a complex interface bridge for Advanced High performance Bus (AHB) as well as Advanced Peripheral Bus (APB) to support for both high bandwidth and low bandwidth data transfer using a single AXI4.0 lite transaction. The data transition distinction is made by N- bit comparator which is designed for switching over between APB and AHB. The Paper also involves the Back annotation for Synthesized Net list of Bridge module and to perform Functional and Timing Simulation using Xilinx.

Keywords-SoC; AMBA; AXI; APB; AHB

I. INTRODUCTION

Integrated circuits has entered the era of System-on-a-Chip (SoC), which refers to integrating all components of a computer or other electronic system into a single chip. It may contain digital, analog, mixed-signal, and often radio -frequency functions – all on a single chip substrate. With the increasing design size, IP is an inevitable choice for SoC design. And the widespread use of all kinds of IPs has changed the nature of the design flow, making On-Chip Buses (OCB) essential to the design. Of all OCBs existing in the market, the AMBA bus system is widely used as the de facto standard SoC bus.

On March 8, 2010, ARM announced availability of the AMBA 4.0 specifications. As the de facto standard SoC bus, AMBA bus is widely used in the high-performance SoC designs. The AMBA specification defines an on-chip communication standard for designing high-performance embedded microcontrollers. The AMBA 4.0 specification defines five buses/interfaces [1]:

- Advanced extensible Interface (AXI)
- Advanced High-performance Bus (AHB)
- Advanced System Bus (ASB)
- Advanced Peripheral Bus (APB)
- Advanced Trace Bus (ATB)

AXI, the next generation of AMBA interface defined in the AMBA 4.0 specification, is targeted at high performance; high clock frequency system designs and includes features which make it very suitable for high speed sub-micrometer interconnections.

- Separate address/control and data phases
- support for unaligned data transfers using byte strobes
- burst based transactions with only start address issued
- issuing of multiple outstanding addresses
- easy addition of register stages to provide timing closure

II.TOP VIEW

2.1Block Diagram

In this study, we focused mainly on the implementation aspect of an AXI4-Lite to APB bridge and also AHB bridge. It is required to bridge the communication gap between low bandwidth peripherals on APB with the high bandwidth ARM Processors and/or other high-speed devices on AHB. This is to ensure that there is no data loss between **AXI4.0 to** AHB and **AXI4.0 to** APB or wise versa.

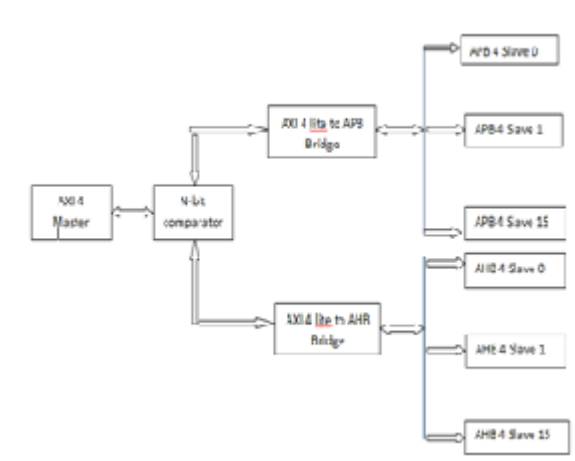

FIGURE 1. BLOCK DIAGRAM

The APB bridge provides an interface between the

high-performance AXI domain and the low-power APB domain. It appears as a slave on AXI bus but as a master on APB that can access up to sixteen slave peripherals. Read and write transfers on the AXI bus are converted into corresponding transfers on the APB. The AHB bridge provides an interface between the high-performance AXI domain and the low-power AHB domain. It is very easy to study this paper as two bridges separately and the results combined with the n bit comparator will give clear picture of the paper. First the main diagram is shown as above. Then the two bridges will be studied separately.

2.2. Block Diagram of axi- 4lite to APB bridge

The APB bridge provides an interface between the high-performance AXI domain and the low-power APB domain. It appears as a slave on AXI bus but as a master on APB that can access up to sixteen slave peripherals. Read and write transfers on the AXI bus are converted into corresponding transfers on the APB. The AXI4-Lite to APB bridge block diagram is shown in Figure. 2

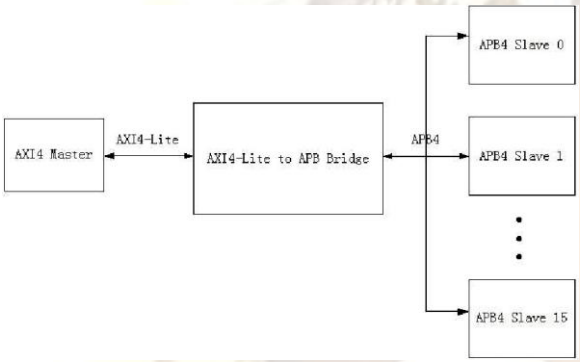

FIGURE 2. **Block diagram *of axi4- lite to APB Bridge***

A. AXI Handshake Mechanism

In AXI 4.0 specification, each channel has a VALID and READY signal for handshaking [2]. The source asserts VALID when the control information or data is available. The destination asserts READY when it can accept the control information or data. Transfer occurs only when both the VALID and READY are asserted. Figure. 3 show all possible cases of VALID/READ handshaking. Note that when source asserts VALID, the corresponding control information or data must also be available at the same time. The arrows in Figure. 3 indicate when the transfer occurs. A transfer takes place at the positive edge of clock. Therefore, the source needs a register input to sample the READY signal. In the same way, the destination needs a register input to sample the VALID signal. Considering the situation of Figure. 3(c), we assume the source and destination use output registers instead of combination circuit, they need one cycle to pull low VALID/READY and sample the VALID/RE ADY again at T4 cycle. When they sample the VALID/REA DY again at T4, there should be another transfer which is an error. Therefore source and destination should use combinational circuit as output. In short, AXI protocol is suitable register input and combinational output circuit.

The APB bridge buffers address, control and data from AXI4-Lite, drives the APB peripherals and returns data and response signal to the AXI4-Lite. It decodes the address using an internal address map to select the peripheral. The bridge is designed to operate when the APB and AXI4-Lite have independent clock frequency and phase. For every AXI channel, invalid commands are not forwarded and an error response generated. That is once an peripheral accessed does not exist, the APB bridge will generate DE CERR as response through the response channel (read or write). And if the target peripheral exists, but asserts PSLVERR, it will give a SLVERR response.

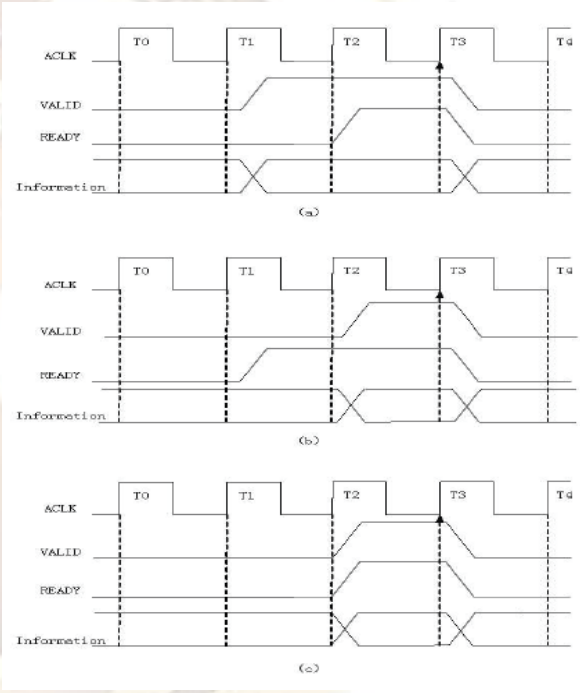

Figure 3. Handshake mechanism

2.3. Block Diagram of axi- 4lite to AHB Bridge

AHB-Lite Master Interface:

• Supports incrementing burst transfers of length 4, 8, 16, and undefined burst length

• AHB-Lite master does not issue incrementing burst transfers that cross 1 kB address boundaries

• Supports limited protection control

• Supports narrow transfers (8/16-bit transfers on a 32-bit data bus and 8/16/32-bit transfers on a 64-bit data bus)

The AXI to AHB-Lite Bridge translates AXI4 transactions into AHB-Lite transactions. The bridge

functions as a slave on the AXI4 interface and as a master on the AHB-Lite interface.

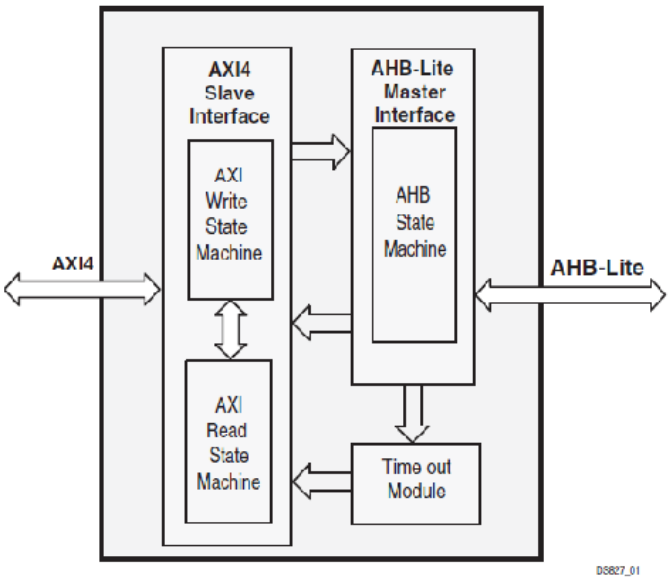

Figure 1: **AXI to AHB-Lite Bridge Block Diagram**

AXI4 Slave Interface

The AXI4 Slave Interface module provides a bi-directional slave interface to the AXI. The AXI address width is fixed at 32 bits. AXI data bus width can be either 32 or 64 bit based on the parameter C_S_AXI_DATA_WIDTH. AXI to AHB-Lite Bridge supports the same data width on both AXI4 and AHB-Lite interfaces. When both write and read transfers are simultaneously requested on AXI4, the read request is given a higher priority than the write request.

AXI Write State Machine

AXI write state machine is part of the AXI4 slave interface module and functions on AXI4 write channels. This module controls AXI4 write accesses and generates the write response to AXI. If bridge time out occurs, this module completes the AXI write transaction with SLVERR response.

AXI Read State Machine

AXI read state machine is part of the AXI4 slave interface module and functions on AXI4 read channels. This module controls the AXI4 read accesses and generates the read response to AXI. If bridge time out occurs, this module completes the AXI read transaction with SLVERR response.

AHB-Lite Master Interface

The AHB-Lite master interface module provides the AHB-Lite master interface on the AHB-Lite. The AHB-Lite
address width is fixed at 32 bit and data bus width can be either 32 or 64 bit, based on the parameter C_M_AHB_DATA_WIDTH.
C_M_AHB_DATA_WIDTH cannot be set by the user, because it is updated automatically with C_S_AXI_DATA_WIDTH.

AHB State Machine

AHB state machine is part of the AHB-Lite Master interface module.When AXI4 initiates the write access, the AHB state machine module receives the control signals and data from AXI4 slave interface, then transfers the same to the equivalent AHB-Lite write access. This module also transfers the AHB-Lite write response to the AXI4 slave interface.
When AXI4 initiates the read access, the AHB state machine module receives the control signals from AXI4 slave
interface, then transfers the same to the equivalent AHB-Lite read access. This module also transfers AHB-Lite read
data and read response to the AXI4 slave interface.

Time out Module

The time out module generates the time out when the AHB-Lite slave is not responding to the AHB transaction.
This is parameterized and generates the time out only when C_DPHASE_TIMEOUT value is nonzero.The time out module waits for the duration of the C_DPHASE_TIMEOUT number of AXI clocks for AHB-Lite slave response, then generates the time out if the AHB slave is not responding.

2.4. Migration from AXI To AHB

With modern Systems on Chip including multi-core clusters, additional DSP, graphics controllers and other sophisticated peripherals, the system fabric poses a critical performance bottleneck. The AHB protocol, even in its multi-layer configuration cannot keep up with the demands of today's SoC. The reasons for this include:
1. AHB is transfer-oriented. With each transfer, an address will be submitted and a single data item will be written to or read from the selected slave. All transfers will be initiated by the master. If the slave cannot respond immediately to a transfer request the master will be stalled. Each master can have only one outstanding transaction.
2. Sequential accesses (bursts) consist of consecutive transfers which indicate their relationship by asserting HTRANS/HBURST accordingly.
3. Although AHB systems are multiplexed and thus have independent read and write data buses, they cannot operate in full-duplex mode.

An AXI interface consists of up to five channels which can operate largely independently of each other. Each channel uses the same trivial handshaking between source and destination (master or slave, depending on channel direction), which

simplifies the interface design. Unlike AHB concept is not an afterthought but is the central focus of the protocol design. In AXI3 all transactions are bursts of lengths between 1 and 16. The addition of byte enable signals for the data bus supports unaligned memory accesses and store merging.

The communication between master and slave is transaction-oriented, where each transaction consists of address, data, and response transfers on their corresponding channels. Apart from rather liberal ordering rules there is no strict protocol-enforced timing relation between individual phases of a transaction. Instead every transfer identifies itself as part of a specific transaction by its transaction ID tag. Transactions may complete out-of-order and transfers belonging to different transactions may be interleaved. Thanks to the ID that every transfer carries, out-of-order transactions can be sorted out at the destination.

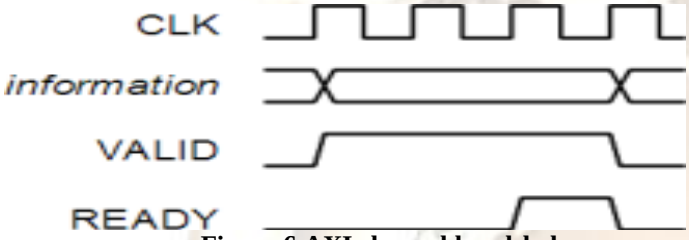

Figure 6.AXI channel handshake

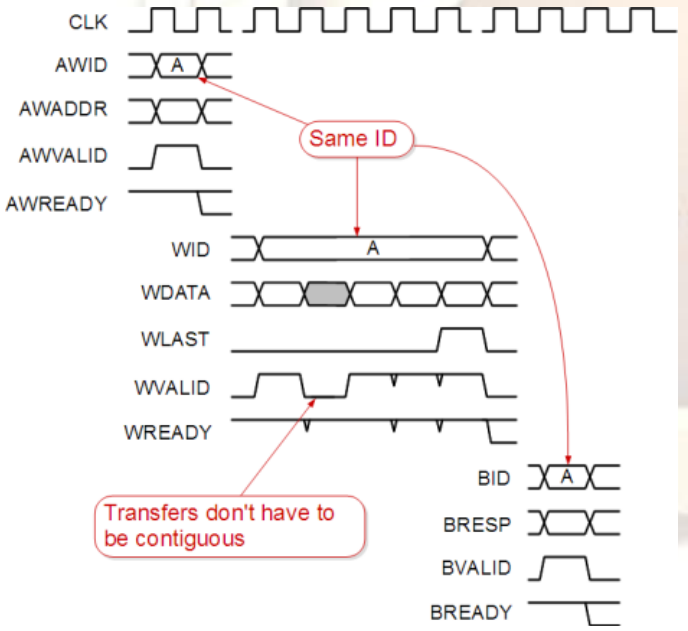

Figure 7. AXI write burst

This flexibility requires all components in an AXI system to agree on certain parameters, such as write acceptance capability, read data reordering depth and many others. Due to the vast number of signals that make up a read/write AXI connection, routing a large AXI fabric could be thought of as rather challenging. However, the independent channels in an AXI fabric make it possible to choose a different routing structure depending on the expected data volume on that channel. Given a situation where the majority of transactions will transfer more than one data item, data channels should be routed via crossbar so that different streams can be processed at the same time. Address and response channels experience rather lower traffic and could perhaps be multiplexed. Some experts consider it an advantage to provide AXI only at the interface level, while a special packetized routing protocol is used inside the fabric, a so called Network-on-Chip.

The AHB is a single-channel, shared bus. The AXI is a multi-channel, read/write optimized bus. Each bus master, or requesting bus port, connects to the single-channel shared bus in the AHB, while each AXI bus master connects to a Read address channel, Read data channel, Write address channel, Write data channel, and Write response channel. The primary throughput channels for the AXI are the Read/Write data channels, while the address, response channels are to improve pipelining of multiple requests. Assume there are four masters on each bus going to three slaves. The four master ports might include microprocessor, Direct Memory Access (DMA), DSP, USB. The three slaves might include on-chip RAM, off-chip SDRAM, and an APB bus bridge.

To approximate the bandwidth of the two busses, one must count the number of read/write channels of the AXI Bus – six for three bus slaves. This suggests that the AHB Bus should support some multiple of bus width and/or speed to match the data throughput. The System Model can vary these combinations with simple parameter changes, however, the AHB bus speed was assumed to be double the AXI Bus, and two times the width. This will make the comparison of the two busses more realistic.

To evaluate the efficiency of both busses, different burst sizes were selected; small, medium, and large. Small equates to the width of the AHB Bus, medium equates to two AHB Bus transfers, and large equates to four AHB bus transfers.

If the AXI is a 64 bit bus running at 200 MHz, then the AHB will be a 128 bit bus running at 400 MHz. The burst sizes will be: small (16 Bytes), medium (32 Bytes), and large (64 Bytes).

III. SIMULATION & IMPLEMENTATION

The timing diagrams shown in Figure.8& 9 illustrate the AXI4-Lite to APB bridge operation and AXI4-Lite to AHB bridge operation for various read and write transfers. It shows that when both read and write requests are active, read is given more priority.

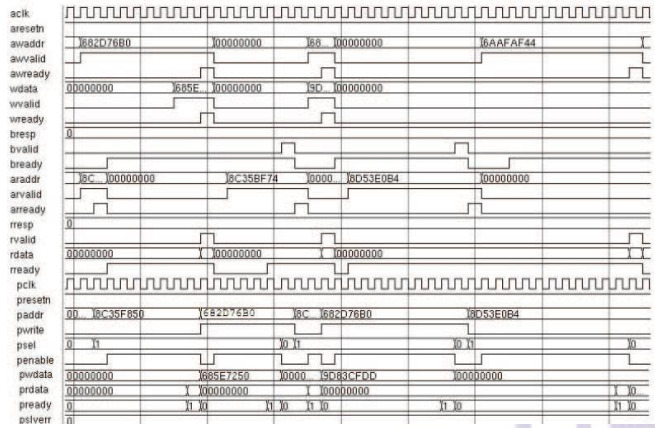

Figure 8.APB typical read and write transfer

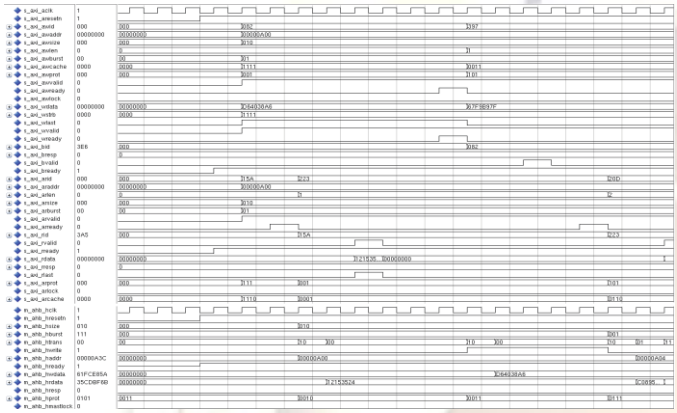

Figure 9.AHB typical read and write transfer

After synthesis and STA, the summary reports gererated by the EDA tools (Synopsys Design CompilerTM and Synopsys Prime TimeTM) are shown in fallowing Figure.

```
****************************************
Report : constraint
         -all_violators
Design : AXI4Lite2APB
Version: D-2010.03-SP2
Date   : Mon Feb 28 16:17:34 2011
****************************************

This design has no violated constraints.

1
```

```
****************************************
Report : area
Design : AXI4Lite2APB
Version: D-2010.03-SP2
Date   : Mon Feb 28 16:22:19 2011
****************************************

Library(s) Used:

    typical (File: /home/derek/design/tsmc/typical.db)

Number of ports:                274
Number of nets:                 802
Number of cells:                531
Number of references:            24

Combinational area:       2732.813953
Noncombinational area:    7819.921537
Net Interconnect area:    179200.938904

Total cell area:          10552.735491
Total area:               189753.674395
1
 Path Group: ACLK
 Path Type: max

 Point                                    Incr       Path
 ------------------------------------------------------------
 clock ACLK (rise edge)                   0.00       0.00
 clock network delay (ideal)              0.00       0.00
 APCLK_reg[0]/CK (DFFRX1)                 0.00       0.00 r
 APCLK_reg[0]/Q (DFFRX1)                  0.33       0.33 f
 U801/Y (NAND4X1)                         0.08       0.41 r
 U755/Y (CLKINVX1)                        0.07       0.48 f
 U770/Y (CLKBUFX3)                        0.16       0.64 f
 U883/Y (OAI2BB2XL)                       0.12       0.76 r
 RDATA_reg[7]/D (DFFRX1)                  0.00       0.76 r
 data arrival time                                   0.76

 clock ACLK (rise edge)                   5.00       5.00
 clock network delay (ideal)              0.00       5.00
 RDATA_reg[7]/CK (DFFRX1)                 0.00       5.00 r
 library setup time                      -0.14       4.86
 data required time                                  4.86
 ------------------------------------------------------------
 data required time                                  4.86
 data arrival time                                  -0.76
 ------------------------------------------------------------
 slack (MET)                                        4.11


 Startpoint: CurrentState_reg[1]
             (rising edge-triggered flip-flop clocked by PCLK)
 Endpoint: PADDR_reg[0]
           (rising edge-triggered flip-flop clocked by PCLK)
 Path Group: PCLK
 Path Type: max

 Point                                    Incr       Path
 ------------------------------------------------------------
 clock PCLK (rise edge)                   0.00       0.00
 clock network delay (ideal)              0.00       0.00
                                                      <ming.rpt   54 1 62%
```

V.CONCLUSION

In this study, we provide an implementation of AXI4-Lite to APB bridge and AXI4-Lite to AHB bridge which has the following features:

- 32-bit AXI slave and APB & AHB master interfaces.
- PCLK clock domain completely independent of ACLK clock domain.
- Support up to 16 APB & AHB peripherals.

- support the PREADY signal which translate to wait States on AXI.
- an error on anytransfer results in SLVERR as the AXI read/write response.

VI.REFERENCES

[1] ARM, "AMBA Protocol Specification 4.0", www.arm.com, 2010.
[2] Ying-Ze Liao, "System Design and Implementation of AXI Bus",National Chiao Tung University, October 2007.
[3] Clifford E. Cummings, "Coding And Scripting Techniques For FSMDesigns With Synthesis-Optimized, Glitch-Free Outputs," SNUG (Synopsys Users Group Boston, MA 2000) Proceedings, September2000.
[4] Clifford E. Cummings, "Synthesis and Scripting Techniques forDesigning Multi-Asynchronous Clock Designs," SNUG 2001
[5] Chris Spear, "SystemVerilog for Verification, 2nd Edition", Springer,www.springeronline.com, 2008.
[6] Lahir, K., Raghunathan A., Lakshminarayana G., "LOTTERYBUS: anew high-performance communication architecture for system-on-chip deisgns," in Proceedings of Design Automation Conference, 2001.
[7] Sanghun Lee, Chanho Lee, Hyuk-Jae Lee, "A new multi-channel onchip-bus architecture for system-on-chips," in Proceedings of IEEE international SOC Conference, September 2004.
[8] Martino Ruggiero, Rederico Angiolini, Francesco Poletti, DavideBertozzi, Luca 86
[9] Benini, Roberto Zafalon, "Scalability Analysis of Evolving SoCInterconnect Protocols," Int. Symposium on System-on-Chip, 2004.
Lukai Cai, Daniel Gajski, "Transaction level modeling: an overview," inProceedings of the 1st IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis, October 2003.
[10] Min-Chi Tsai, "Smart Memory Controller Design for VideoApplications," Master thesis: National Chiao Tung University, July 2