Chapter 3 AMBA System Implementation

3.1 AMBA Specification Introduction

The MPEG-4 IP is designed as a coprocessor which helps the embedded processor to handle the complex video compression algorithm. The ARM CPU and AMBA bus is the most popular CPU and bus architecture in the consumer electronic market now. In order to make the MPEG-4 coprocessor more applicable, it is a good choice to wrap it in AMBA system.

3.1.1 Overview of the AMBA Specification

The Advanced Microcontroller Bus Architecture (AMBA) specification defines an on-chip communication standard for designing high-performance embedded microcontrollers [11].

Three distinct buses are defined within the AMBA specification：

✧ The Advanced High-performance Bus (AHB)

The AMBA AHB is for high-performance, high clock frequency system modules. The AHB acts as the high-performance system backbone bus. AHB supports the efficient connection of processors, on-chip memories and off-chip external memory interfaces with low-power peripheral macrocell functions. AHB is also specified to ensure ease of use in an efficient design flow using synthesis and automated test techniques.

✧ The Advanced System Bus

AMBA ASB is an alternative system bus suitable for use where the high-performance features of AHB are not required. We don’t use ASB in our design.

✧ The Advanced Peripheral Bus

The AMBA APB is for low-power peripherals.

AMBA APB is optimized for minimal power consumption and reduced interface complexity to support peripheral functions. APB can be used in conjunction with either version of the system bus.

3.1.2 Objectives of the AMBA Specification

The AMBA specification has been derived to satisfy four key requirements：

✧ To facilitate the right-first-time development of embedded microcontroller products with one or more CPUs or signal processors.

✧ To be technology-independent and ensure that highly reusable peripheral and system macrocells can be migrated across a diverse range of IC processes and be appropriate for full-custom, standard cell and gate array technologies.

✧ To encourage modular system design to improve processor independence, providing a development road-map for advanced cached CPU cores and the development of peripheral libraries.

✧ To minimize the silicon infrastructure required to support efficient on-chip and off-chip communication for both operation and manufacturing test.

3.1.3 A typical AMBA-based system

An AMBA-based microcontroller typically consists of a high-performance system backbone bus, able to sustain the external memory bandwidth, on which the CPU and other Direct Memory Access devices reside, plus a bridge to a narrower APB bus on which the lower bandwidth peripheral devices are located. Fig. 3-1 shows both AHB and APB in a typical AMBA system.

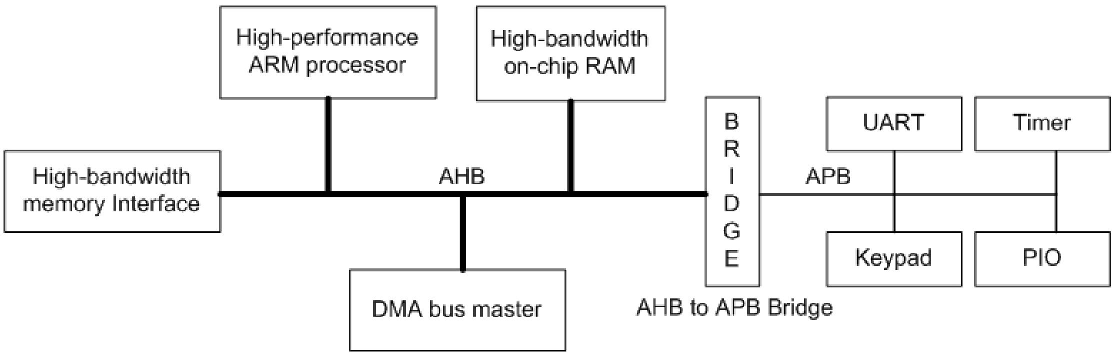
High-performance ARM processor
High-bandwidth on-chip RAM
BRIDGE
UART
Timer
High-bandwidth memory Interface
AHB
APB
Keypad
PIO
DMA bus master
AHB to APB Bridge

Figure 3-1 A typical AMBA AHB-based system

3.1.4 Bus Interconnection

The AMBA AHB bus protocol is designed to be used with a central multiplexor interconnection scheme. Using this scheme all bus masters drive out the address and control signals indicating the transfer they wish to perform and the arbiter determines which master has its address and control signals routed to all of the slaves. A central decoder is also required to control the read data and response signal multiplexor, which selects the appropriate signals from the slave that is involved in the transfer. Fig. 3-2 illustrates the structure required to implement an AMBA AHB design with three masters and four slaves.

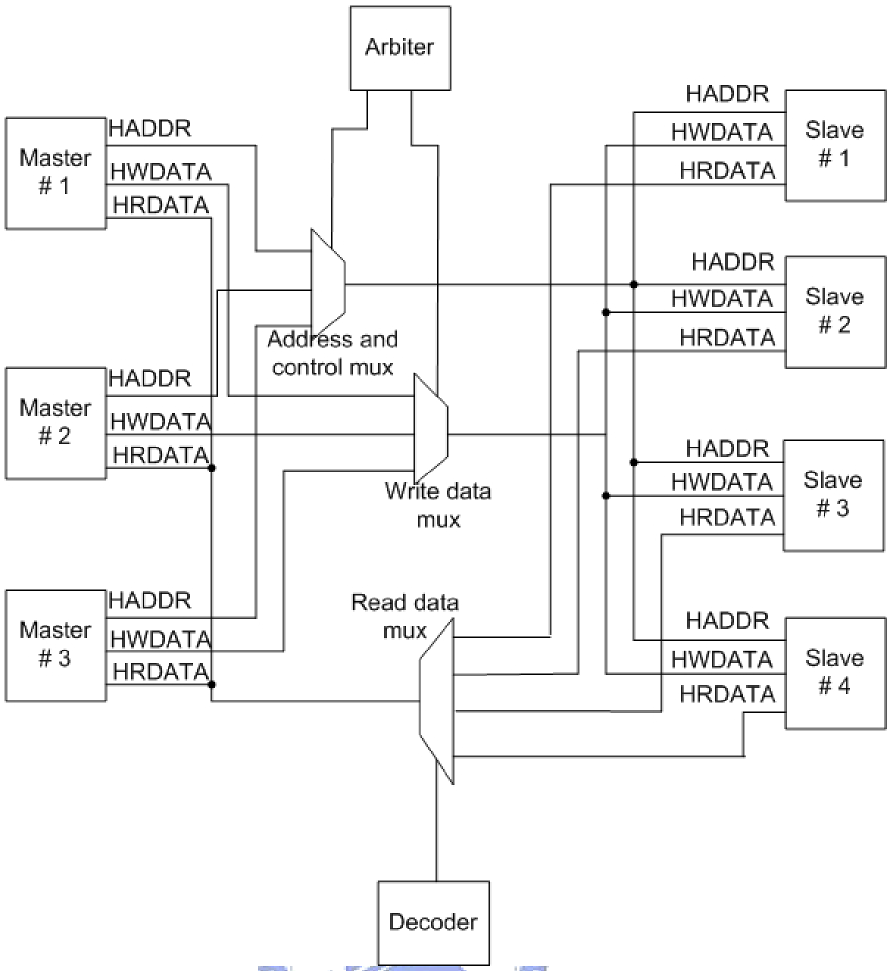

Figure 3-2 AHB interconnection

3.1.5 Overview of AMBA AHB Operation

Before an AMBA AHB transfer can commence the bus master must be granted access to the bus. This process is started by the master asserting a request signal to the arbiter. Then the arbiter indicates when the master will be granted use of the bus.

A granted bus master starts an AMBA AHB transfer by driving the address and control signals. These signals provide information on the address, direction and width of the transfer, as well as an indication if the transfer forms part of a burst. Two different forms of burst transfers are allowed:

✧ Incrementing bursts, which do not wrap at address boundaries

- ✧ Wrapping bursts, which wrap at particular address boundaries.

A write data is used to move data from the master to a slave, while a read data bus is used to move data from a slave to the master.

Every transfer consists of:

- ✧ An address and control cycle
- ✧ One or more cycles for the data.

The address cannot be extended and therefore all slaves must sample the address during this time. The data, however, can be extended using the HREADY signal. When LOW this signal causes wait states to be inserted into the transfer and allows extra time for the slave to provide or sample data.

During a transfer the slave shows the status using the response signals, HRESP [1:0]:

- ✧ **OKAY** The OKAY response is used to indicate that the transfer is progressing normally and when HREADY goes HIGH this shows the transfer has completed successfully.
- ✧ **ERROR** The ERROR response indicates that a transfer error has occurred and the transfer has been unsuccessful.
- ✧ **RETRY and SPLIT** Both the RETRY and SPLIT transfer responses indicate that the transfer cannot complete immediately, but the bus master should continue to attempt the transfer.

In normal operation a master is allowed to complete all the transfers in a particular burst before the arbiter grants another master access to the bus. However, in order to avoid excessive arbitration latencies it is possible for the arbiter to break up a burst and in such case the master must re-arbitrate for the bus in order to complete the remaining transfers in the burst.

3.2 AHB Wrapper Design

In order to fit for the communication protocol of AMBA, an AHB wrapper is designed for the MPEG-4 encoder IP. Except for wrapping function, the wrapper also contains configurable register. On the IP reusable and platform independent consideration, the encoder IP is controlled by register value. The register bank is a slave on the AMBA, and the MPEG-4 IP is a master on the AMBA. Thus this wrapper has both master and slave interface. The AMBA architecture is shown in Fig. 3-3. The register bank functionality in the AMBA wrapper is described in Table 3-1.

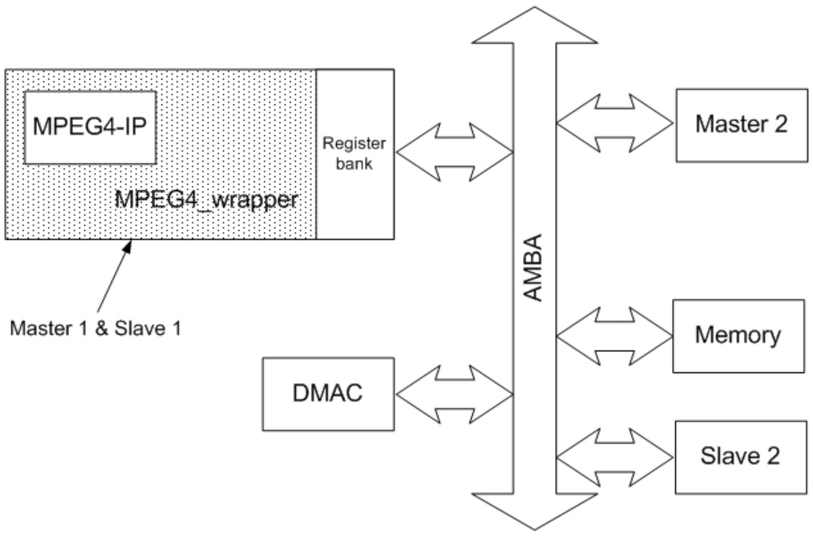

Figure 3-3 MPEG-4 encoder on AMBA architecture

Table 3-1 AMBA wrapper programmable register list

Register Name	Readable or Writable	Address(For ARM Integrator system)	Description
MPG4_status [31:0]	R/W	0xC4000000	The status of MPEG-4 IP
MPG4_VLC_info [31:0]	R/W	0xC4000008	Hardware encoding parameter
Mem1_L0_CUR_STR_ADDR [31:0]	W	0xC400000C	The start address of the Mem1 luminance data
Mem1_U_STR_ADDR [31:0]	W	0xC4000010	The start address of the Mem1 U data (V address must following the U address)
Mem2_L0_CUR_STR_ADDR [31:0]	W	0xC4000014	The start address of the Mem2 luminance data
Mem2_U_STR_ADDR [31:0]	W	0xC4000018	The start address of the Mem2 U data (V address must following the U address)
ME_DATA_OUT_ADDR [31:0]	W	0xC400001C	Start address of the SAD and the MV
EXT_RAM_END_ADDR [31:0]	W	0xC4000020	The end address of the external memory
BITSTREAM_STR_ADDR [31:0]	W	0xC4000024	The start address of the bitstream

Register functionality description:

✧ MPG4_status:

This register is set to 0x00 after system reset. This register could be configured as “I/P frame mode” by the RISC or the DMAC, and changed to idle mode by the MPEG-4 encoder.

I frame mode = 0x04.
P frame mode = 0x05.

Hardware finish = 0x08.

The hardware finish state means that the MPEG-4 encoder has finished one frame compression and stayed in the idle state.

✧ MPG4_VLC_info:

[31:30]: pre_cur_frame_status, these two bits are configured by the RISC.

In the I frame mode, the Mem1 always is current frame.

In the P frame mode:

(Please refer to section 2.2 “Motion Estimation Architecture Design”)

Table 3-2 Pre_cur_frame_status table

Pre_cur_frame_status value	Mem1 status	Mem2 status
“01”	Current frame	Previous frame
“10”	Previous frame	Current frame

[29:25]=Q_param, these bits are configured by the RISC to set the quantization step.

[24:5]=byte_cnt, these bits are configured by the MPEG-4 encoder. This value tells how many bytes the bitstream has been encoded by the MPEG-4 encoder for this frame.

[4:0]=bit_pos, these bits are configured by the MPEG-4 encoder. This value tells which position the last bit of bitstream is. A bit position example is shown in Fig. 3-4.

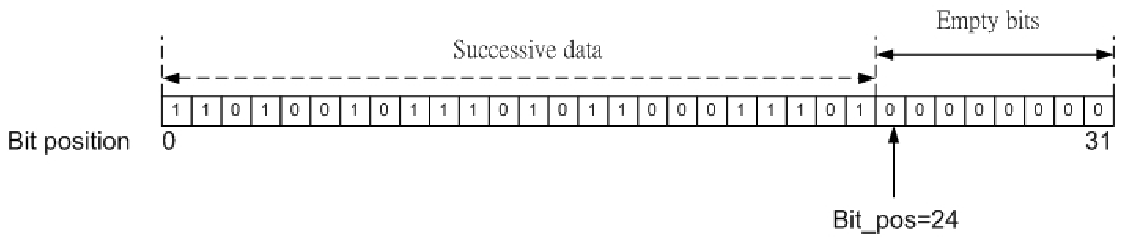

Figure 3-4 A bit position example

✧ Mem1_L0_CUR_STR_ADDR:

This register is configured by the RISC. This register indicates the start address of the luminance data in the Mem1.

✧ Mem1_U_STR_ADDR:

This register is configured by the RISC. This register indicates the start address of the U data in the Mem1. And the V data address must be following the U data.

✧ Mem2_L0_CUR_STR_ADDR:

This register is configured by the RISC. This register indicates the start address of the luminance data in the Mem2.

✧ Mem2_U_STR_ADDR:

This register is configured by the RISC. This register indicates the start address of the U data in the Mem2. And the V data address must be following the U data.

✧ ME_DATA_OUT_ADDR:

This register is configured by the RISC. This register indicates the start address of ME output data. The ME output data are 396 words with continuous address which are correspond to 396 macro blocks in raster scan order in a P frame. The ME output data format is shown in Fig. 3-5. These data could be used optionally as the statistic for frame mode decision.

0000	Level0_SAD(16 bits)	Level0_MVx(6 bits)	Level0_MVx(6 bits)

Figure 3-5 ME output data format

✧ EXT_RAM_END_ADDR:

This register is configured by the RISC, and this register indicates the end address of the external SSRAM.

✧ BITSTREAM_STR_ADDR:

This register is configured by the RISC, and this value indicates the start address of the encoded bitstream.

3.3 AMBA 2.0 Compatible DMAC Design

FPGA prototyping is done on the ARM Integrator, and in order to reduce the software overhead on moving image pixel, a direct memory access controller (DMAC) is adopted. The status of this DMAC is shown in Fig. 3-6.

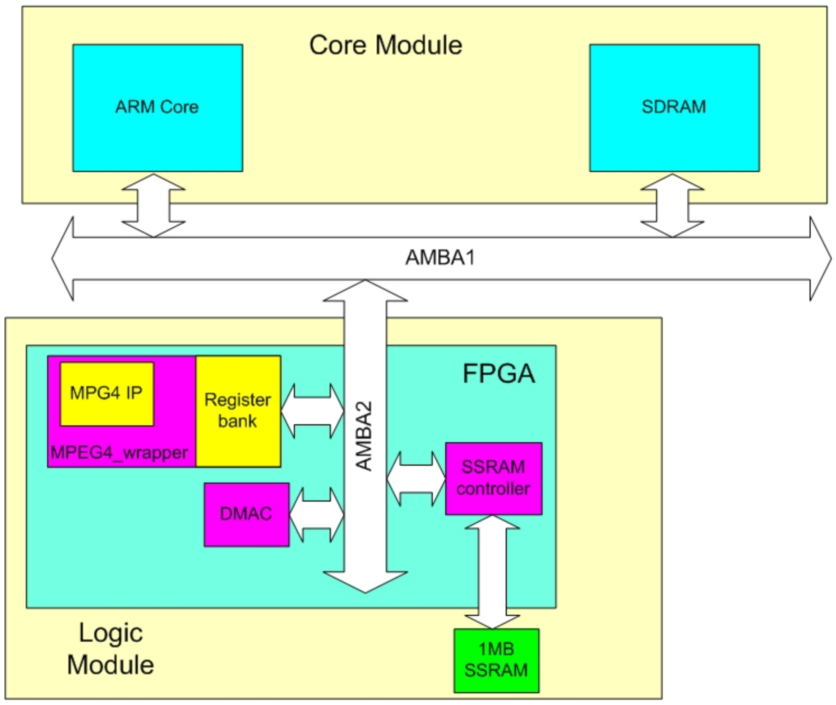

Figure 3-6 MPEG-4 encoder system on ARM Integrator

On one hand the DMAC contains configurable register which could control the DMAC, and on the other the DMAC also being a master on the AMBA. The DMAC acts both as master and slave on the AMBA. The control registers in the DMAC is shown in Table 3-3.

Table 3-3 DMAC control register list

Register Name	Readable or Writable	Address (For ARM Integrator system)	Description
DMA_status	R/W	0xC4100004	The status of the DMAC
DMA_y_rd_start	R/W	0xC410000C	The start address of the luminance data
DMA_u_rd_start	R/W	0xC4100010	The start address of the U data
DMA_v_rd_start	R/W	0xC4100014	The start address of the V data
DMA_Frame_info	R/W	0xC4100018	Image size
SW_last_data	R/W	0xC410001C	The last word of the software encoded bitstream
SW_bitstream_start	R/W	0xC4100020	The start address of the bitstream destination

Register functionality description:

✧ DMA_status:

[31:28]：

I frame = 0x4，P frame=0x5

[7:0]：

DMA read from the DRAM and write to the SSRAM Mem1 = 0x01

DMA read from the DRAM and write to the SSRAM Mem2 = 0x02

DMA read from the SSRAM and write to the DRAM = 0x03

DMA initialize the SSRAM = 0x04

DMA interrupt = 0x10

DMA idle = 0x05

✧ DMA_y_rd_start:

This register is configured by the RISC, and it tells the start address of the source of the luminance data. (On ARM Integrator, the image source is DRAM, and the image destination is SSRAM)

✧ DMA_u_rd_start:

This register is configured by the RISC, and it tells the start address of the source of U data.

✧ DMA_v_rd_start:

This register is configured by the RISC, and it tells the start address of the source of the V data.

✧ DMA_Frame_info:

[27:14]：(Y image width)/4. (Data bus is 32bits, and pixel value is 8 bits)

[13:0]：Y image height.

This register is configured by the RISC.

✧ SW_last_data:

This register is configured by the RISC, and this register stores the last word of bitstream which is not fully completed by software. The DMAC will cascade the last word of the software encoded bitstream and the first word of the hardware encoded bitstream.

✧ SW_bitstream_start:

This register is configured by the RISC, and this register tells the start address of bistream destination. (In ARM Integrator the bitstream source is SSRAM, and the destination is DRAM)

Basically the DMAC is a finite state machine. The state machine diagram is shown in Fig. 3-7. When the system is reset, the DMAC is stay in the idle state. In the beginning, the DMAC_status register must be written 0x04 to initialize the SSRAM to 0xFFFFFFFF. When the RISC write 0x01 or 0x02 to the DMA_status register, the DMAC will start to move image from the DRAM to the SSRAM and after moving data the DMAC starts the MPEG-4 IP to encode. When the MPEG-4 IP finishes encoding, the MPEG-4 IP will jump back to idle state and write 0x03 to the DMA_status register and the DMAC will be started to move bitsteam from the SSRAM to the DRAM. After moving bitstream to the DRAM, the DMAC will set an interrupt signal to the RISC and notify the RISC one frame encoding has been accomplished.

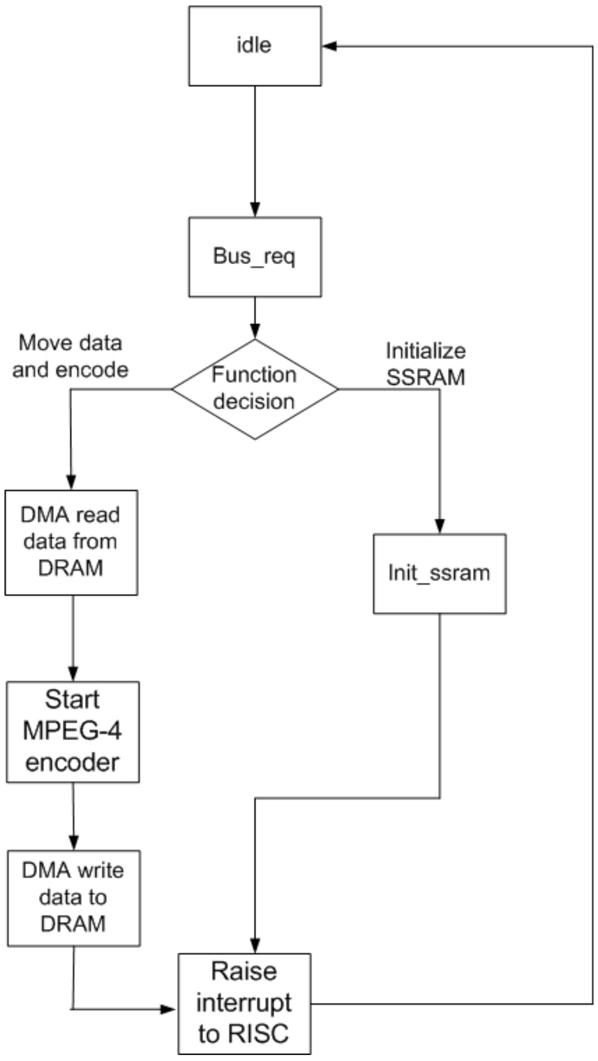

Figure 3-7 DMAC state machine diagram

The whole encoding procedure is fully hardware handling and without software overhead. The encoding procedure is shown in Fig. 3-8.

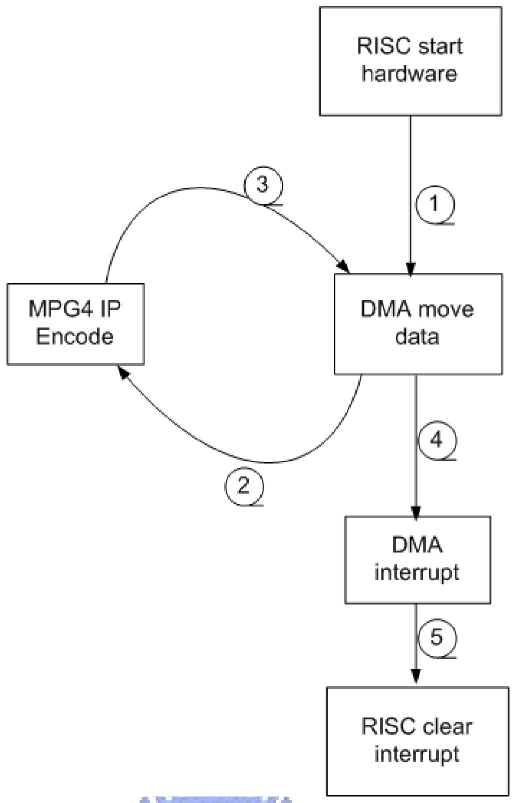

Figure 3-8 Encoding procedure

3.4 AMBA System Architecture on ARM Integrator

In order to verify the hardware design rapidly, the ARM Integrator is choosen to do FPGA prototyping. The ARM Integrator contains ARM CPU, AMBA bus and FPGA. The system architecture of the MPEG-4 encoder on the ARM Integrator is shown in Fig. 3-6. Each component's functionality is described as following:

✧ ARM: This is an ARM CPU, which will handle the image source capture, the VOL and the VOP header package and the hardware control (including the DMAC and the MPEG-4 IP control).

✧ SDRAM: It stores the MPEG-4 encoder images source and the firmware program.

✧ MPEG-4 IP: It is the MPEG-4 encoder IP, which is introduced in the Chapter2.

✧ AMBA wrapper: It is the AMBA wrapper which is introduced in the section 3.2.

✧ DMAC: It is the AMBA 2.0 compatible DMAC which is introduced in the section 3.3.

✧ SSRAM controller: It is the SSRAM controller which transfers the AMBA control signal to the SSRAM control signal.

✧ SSRAM: It is the external memory which introduced in the tail of the section 2.2.

Due to the finite pin number on FPGA, an AMBA2 is built in the FPGA. There is one another advantage by using this kind of two levels bus architecture. The AMBA1 could be released for the ARM to fetch next image from image capture devices or mass storage device when the MPEG-4 encoder IP is encoding. Therefore, when the DMAC finishes moving bitstream to the DRAM, the DMAC could get the newest image source from the DRAM instantly without wasting time on waiting the ARM fetches new image into the SDRAM.