# Design of Crypto Assist Controller with ASIC design Flow

**Tejas Vadhavaniya** [1]
[1]VLSI & Embedded System Design
[1]Gujarat Technological University, Ahmedabad, Gujarat.

*Abstract*---In today's time, everyone wants secure communication. Crypto core is used widely to make data secure. So handling of those data is very complex and important task for fast communication. Main objective of this paper is to design a controller which can handle the traffic of data for crypto core engine. Here, Design of AHB master which can take data from slave. One efficient DMA Controller used to transfer of data from AHB master to Crypto core engine. Here also design of two buffers to overcome the mismatch of size of data bus between AHB master and crypto core engine. So by combining all these modules, design of one controller comes up which takes data from AHB master and transfer them to crypto core engine through DMA controller. Also it'll transfer data from crypto core engine to AHB master through DMA controller.

## I. Introduction

The purpose of this paper is to design interface which can provide fast and better communication between CPU and any external media. Here, as an external media, Crypto core engine is used which is taken from outside as an IP. Main input data is from the CPU which needs encryption. After that, send that data to the Crypto core engine by interfacing of AHB Master and DMA controller. And in the last, send back that encrypted data to the CPU using interfacing of DMAC and AHB master. So, to control any media like Crypto core engine which is connected to CPU, we have to use this type of interface which include DMA controller and AHB master to provide fast and reliable communication.

## II. High Level Design of Crypto Assist Controller

### A. *Specification*

Direct Memory Access Controller [4] [5] [6]

- Burst Transfer
- Interrupt and Error handling Mechanism
- DMA channel status interrupts
- Incrementing and Wrapping Address mode
- Data transfer mode
  - Block transfer mode
  - Burst Block transfer mode
- Byte , Word, Double Word transfer capability
- Asynchronous Fist In-First Out (FIFO) Design - Buffers
- Register Space

AHB Master[7]

- 32 bit Data Bus
- Burst transfer
- Incrementing bursts address mode
- Wrapping bursts address mode
- Byte, Word transfer capability
- Single clock edge operation
- Support pipelined operation
- Data interface with AES Crypto CORE

### B. *Block Diagram*

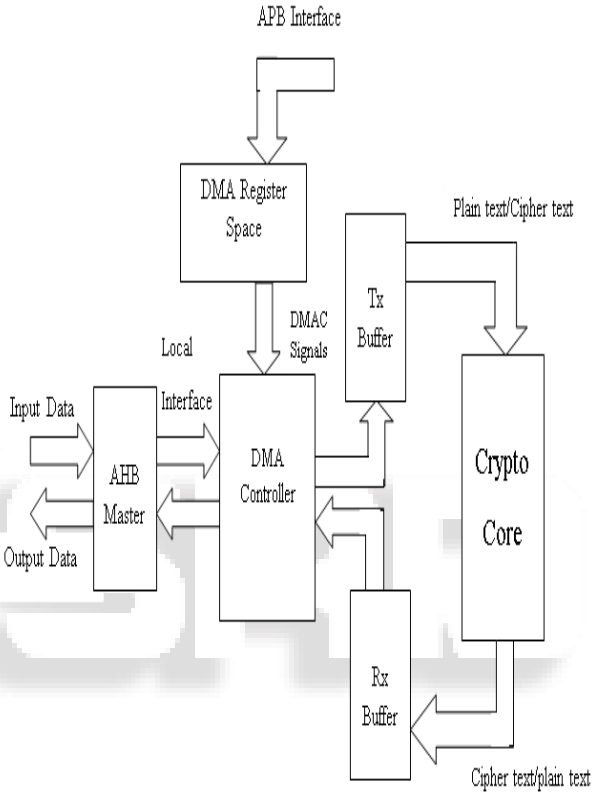


Fig. 1: Block Diagram of Crypto Assist Controller

### C. *Functional Description*

First of all, DMA has to be initiated for the transfer of data from source (input-AHB Master) to destination (Crypto core engine). There is one interface of APB with DMA controller is used to configure the DMA registers. When it set DMA registers to enable the transfer, DMAC will transfer that input data to the Crypto core engine. But the data width of Crypto core engine is 128 bits. And output data from AHB master is of 32 bit. So one transmit buffer is used here which collects the 128 bits of data in 4 clock cycle. Then it sends that 128 bit of input data to the Crypto core engine.

Crypto core engine gives the encrypted data as output which is of again 132 bit. So again there's need of buffer which will work as receiving buffer to send 32 bits of data to AHB master from the 128 bits output data output of Crypto core engine. By using DMAC and AHB master, the data will be sent to the outside.

 

## III. DETAILED DESIGN OF CRYPTO ASSIST CONTROLLER

### *A. AHB Master*

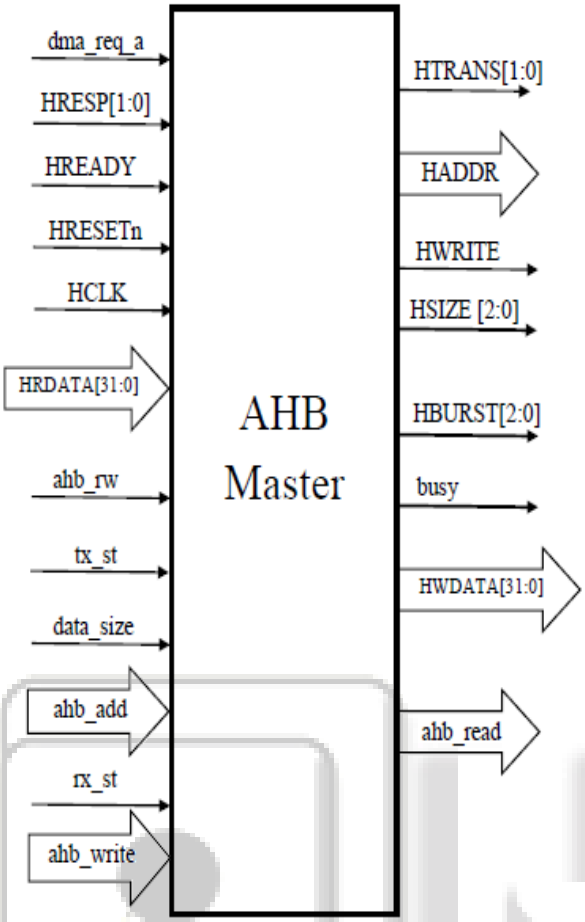


Fig. 2: Pin Diagram of AHB Master

*1) Basic Transfer*[7]

An AHB transfer consists of two distinct sections:

- The address phase, which lasts only a single cycle.
- The data phase, which may require several cycles.
- This is achieved using the HREADY signal.

In a simple transfer with no wait states:

- The master drives the address and control signals onto the bus after the rising edge of HCLK.
- The slave then samples the address and control information on the next rising edge of the clock.
- After the slave has sampled the address and control it can start to drive the appropriate response and this is sampled by the bus master on the third rising edge of the clock.

*2) Transfer Type*

Every transfer can be classified into one of four different types, as indicated below by HTRANS [1:0].[7]

00: IDLE
01: BUSY
10: NONSEQ
11: SEQ

*3) Burst Operation*

Burst mode is a generic computing term referring to any situation in which a device is transmitting data repeatedly without waiting for input from another device or waiting for an internal process to terminate before continuing the transfer of data. AHB master supports burst operation. Four, eight and sixteen-beat bursts are defined in the AMBA AHB protocol, as well as undefined-length bursts and single transfers. Both incrementing and wrapping bursts are supported in the protocol.[7]

- Incrementing bursts access sequential locations and the address of each transfer in the burst is just an increment of the previous address.
- For wrapping bursts, if the start address of the transfer is not aligned to the total number of bytes in the burst (size x beats) then the address of the transfers in the burst will wrap when the boundary is reached. For example, a four-beat wrapping burst of word (4-byte) accesses will wrap at 16-byte boundaries. Therefore, if the start address of the transfer is 0x34, then it consists of four transfers to addresses 0x34, 0x38, 0x3C and 0x30.

*4) Control Signals*

Each transfer will have a number of control signals that provide additional information about the transfer. These control signals have exactly the same timing as the address bus. However, they must remain constant throughout a burst of transfers.[7]

*a) Transfer Direction*

When HWRITE is HIGH, this signal indicates a write transfer and the master will broadcast data on the write data bus, HWDATA [31:0]. When LOW a read transfer will be performed and the slave must generate the data on the read data bus HRDATA [31:0].

*b) Transfer Size*

The size is used in conjunction with the HBURST [2:0] signals to determine the address boundary for wrapping bursts. HSIZE [2:0] indicates the size of the transfer.

*5) Slave Transfer Response*

After a master has started a transfer, the slave then determines how the transfer should progress. Whenever a slave is accessed it must provide a response which indicates the status of the transfer. The HREADY signal is used to extend the transfer and this works in combination with the response signals, HRESP [1:0], which provide the status of the transfer.[7]

- The HREADY signal is used to extend the data port ion of an AHB transfer. When LOW the HREADY signal indicates the transfer is to be extended and when HIGH indicates that the transfer can complete.
- A typical slave will use the HREADY signal to insert the appropriate number of wait states into the transfer and then the transfer will complete with HREADY HIGH and an OKAY response, which indicates the successful completion of the transfer.
- The ERROR response is used by a slave to indicate some form of error condition with the associated transfer. Typically this is used for a protection error, such as an attempt to write to a read-only memory location.

*B. DMA Controller*

DMA Controller communicate with four modules namely its register space, transmit buffer, and receive buffer and AHB master.

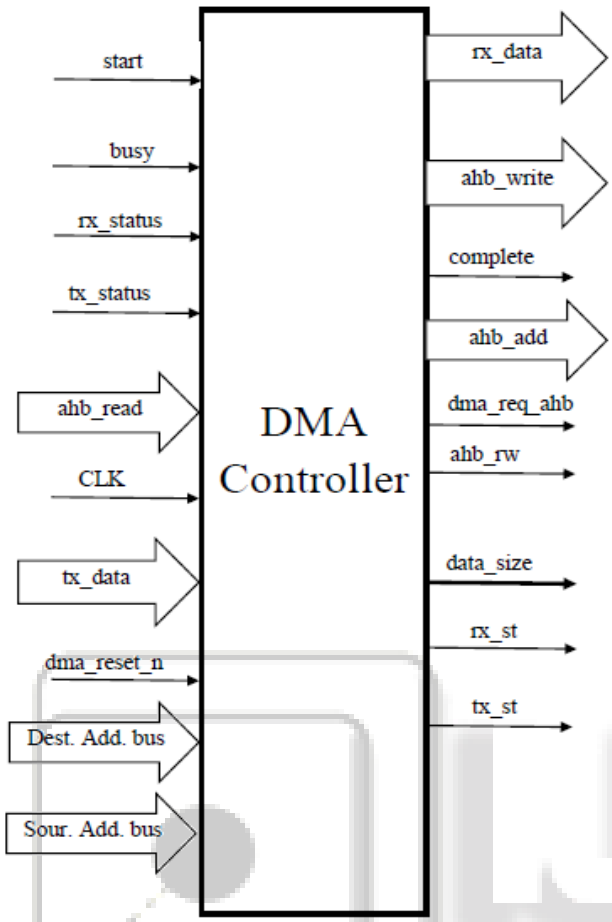


Fig. 3: Pin Diagram of DMA Controller

Direct Memory Access (DMA) allowdevices to transfer data without subjecting the processor a heavy overhead. Otherwise, the processor would have to copy each piece of data from the source to the destination. This is typically slower than copying normal blocks of memory since access to I/O devices over a peripheral bus is generally slower than normal system RAM. During this time the processor would be unavailable for any other tasks involving processor bus access.

But it can continue to work on any work which does not require bus access. DMA transfers are essential for high performance embedded systems where large chunks of data need to be transferred from the input/output devices to or from the primary memory.

- The direct memory access (DMA) I/O technique provides direct access to the memory while the microprocessor is temporarily disabled.
- A DMA controller temporarily borrows the address bus, data bus, and control bus from the microprocessor and transfers the data bytes directly between an I/O port and a series of memory locations.
- The DMA transfer is also used to do high-speed memory-to memory transfers.
- Two control signals are used to request and acknowledge a DMA transfer in the microprocessor-based system.

*C. DMA Register Space*

Register space contains the different registers to control the DMA controller. The DMA controller registers can be accessed by the CPU. In register space, we can use registers as per our need. DMA Register Space contains registers like DMA Source Address Register, DMA Destination Address Register, DMA Control Register,DMA Status Register. We can use these all register to fulfil our requirements and also we can add registers in it as per our design need.

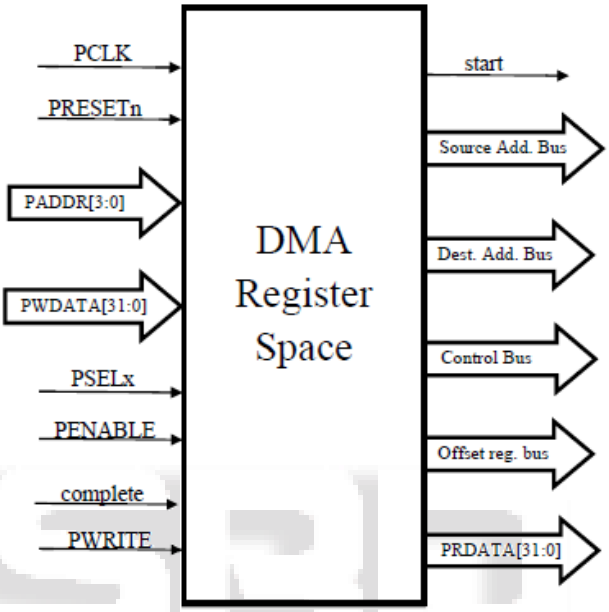


Fig. 4: Pin Diagram of DMA Register Space

*D. Buffers*

In the design, there is need of two buffers. One is transmit buffer and other is receiving buffer. Transmit buffer is to transmit data from AHB master to crypto core engine. Receiving buffer is used to transfer data between crypto core engine and AHB master. There's need of buffer due to mismatch in width of data bus of Crypto core engine and AHB master. AHB master has data width of 32 bits. Where, Crypto core engine has data width of 128 bits. So with the help of buffers, problem of mismatch in data bus can be solved.

*1) Receive Buffer*

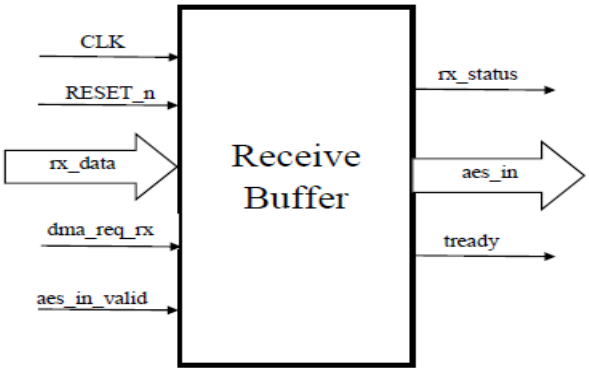


Fig. 5: Pin Diagram of Receive Buffer

 

*2) Transmit Buffer*

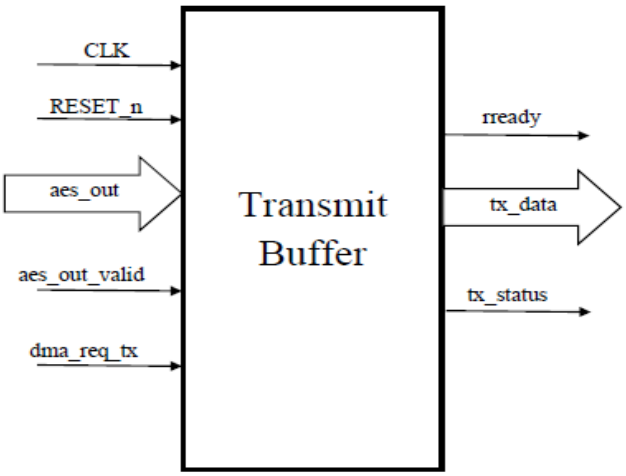


Fig. 6: Pin Diagram of Transmit Buffer

## IV. CONCLUSION

In this paper, I shown the design of Crypto Assist Controller. I followed ASIC design flow for designing this Crypto Assist Controller. Use of AHB Master makes it more fast and efficient for handling data of Crypto Core Engine and DMA Controller keeps CPU free to do other tasks. So, using this Crypto Assist Controller we can transfer and handle the large size of data for AES Crypto Core Engine.

## ACKNOWLEDGEMENT

This work was greatly supported by Gujarat Technological University and C-DAC ACTS, Pune by providing the guidance and support for design of Crypto Assist Controller.

## REFERENCES

[1] Chen-Hsing Wang; Jen-Chieh Yeh; Chih-Tsun Huang; Cheng-Wen Wu, "Scalable Security Processor Design And Its Implementation", *Asian Solid-State Circuits Conference*, 2005, Page(s): 513 – 516 Cited by: Papers (1),Patents (1)

[2] Guoliang Ma; HuHe, "Design and Implementation of an Advanced Dma Controller on Amba-Based Soc" ASIC, 2009. ASICON'09. *IEEE 8th International Conference* on 2009, Page(s): 419– 422

[3] Kuan Jen Lin, Chuang Hsiang Huang, Cheng ChiaLo, "Design and Implementation of A Schedulable Dmac on an Amba-Based Sopc Platform" *Circuits and Systems, 2006. APCCAS 2006. IEEE Asia Pacific Conference on 2006*, Page(s): 279–282, Cited by: Papers (1)

[4] AMBA DMA controller DMA-330 Revision: r1p0, *Technical Reference Manual by ARM Limited*, 2009

[5] Direct Memory Access (DMA) Controller Module, *TEXAS Instruments*, 2013

[6] Quartus II Handbook Version 9.1 Volume 5: Embedded Peripherals, *Altera Corporation*, 2009

[7] AMBA Bus Specification, Revision 2.0, *ARM Ltd*.

[8] Cyclone V Device Handbook Volume 3: Hard Processor System Technical Reference Manual, *Altera Corporation*, 2012.

[9] Chia-Hao Yu, Chung-Kai Liu, Chih-Heng Kang, Tsun-Hsien Wang, Chih-Chien Shen, Shau-Yin Tseng, "An Efficient Dma Controller For Multimedia Application in Mpu Based Soc" *Multimedia and Expo*, 2007 *IEEE International Conference* on 2007, Page(s): 80– 83, Cited by: Paper (1)

[10] Quartus II Handbook Version 9.1 Volume 5: Embedded Peripherals, *0*, 2009

 