Submit to: Workshop on Computer Systems

Bus Wrapper Design Methodology in SoC

Kuang-Li Wu, Jer-Min Jou, and Yeu-Horng Shiau

Abstract

In this paper, the bus wrapper design methodology is proposed in order to generate and synthesize communication interfaces in a system design context. This methodology will be used in bus-based SoCs for IP integration. To verify the practicability, we use this methodology to implementation the on-chip bus wrapper and on-board bus wrapper based on Virtual Component Interface (VCI)-compliant IPs by three cases, which are the AHB master wrapper, the AHB slave wrapper, and the PCI bus target wrapper. We can use the AHB wrapper to integrate the VCI-compliant IP into ARM development system, or use PCI wrapper to integrate the VCI-compliant IP into personal computer system.

In the bus wrapper design we use the buffer to store the address and data temporary instant of FIFO, so we only use a small amount area of bus wrapper. At the performance of the bus wrapper, we use the Mealy Machine Design method, so the input and output of the interface can be pass through the wrapper as soon as possible. It will not cause the communication latency between the interface of the bus and standard interface.

Key words: VCI, AHB, PCI, Bus Wrapper, interface conversion

Kuang-Li Wu, (the contact author)

Current affiliation: Department of Electrical Engineering, National Cheng Kung University, Tainan, Taiwan, R. O. C.

Postal address: ASIC LAB, EE 10F, NCKU, NO.1, Ta-Hsueh Road, Tainan, 701 Taiwan

E-mail address: scott@j92a21.ee.ncku.edu.tw

Telephone number: 06-2757575-62431-821

Yeu-Horng Shiau,

Current affiliation: Department of Electrical Engineering, National Cheng Kung University, Tainan, Taiwan, R. O. C.

Postal address: ASIC LAB, EE 10F, NCKU, NO.1, Ta-Hsueh Road, Tainan, 701 Taiwan

E-mail address: huh@j92a21.ee.ncku.edu.tw

Telephone number: 06-2757575-62431-821

Jer-Min Jou

Current affiliation: Department of Electrical Engineering, National Cheng Kung University, Tainan, Taiwan, R. O. C.

Postal address: ASIC LAB, EE 10F, NCKU, NO.1, Ta-Hsueh Road, Tainan, 701 Taiwan

E-mail address: jou@j92a21.ee.ncku.edu.tw

Telephone number: 06-2757575-62365

1. Introduction

The Virtual Sockets Interface Alliance (VSIA) recently released the Virtual Component Interface (VCI) Standard. The VCI make the VCI-compliant component to communication with one another easily, even if the IP is provided by different companies and different organizations. The VCI-compliant IP may connect to another IP by point-to-point communication, or via VCI- compliant bus wrappers to communication with standard bus, such as AMBA High Bus (AHB), Peripheral Component Interface (PCI), etc.

The remainder of this paper is organized as follows. In Section 2 we introduce the requirement standards. Then, we describe the bus wrapper design model in session 3. Session 4 presents the three cases of the bus wrappers. Session 4 is the conclusion.

2. Interface and Bus Standards

2.1 The VCI Standard

The virtual Component Interface standard defines a point-to-point communication with cycle-based address-mapped interface. It does not demand for a fixed architecture for implementation. It can be used at point-to-point architecture, bus architecture or star architecture, etc. [1-2]

The VCI families now have three protocol types: the Peripheral Virtual Component Interface (PVCI), the Basic Virtual Component Interface (BVCI) and the Advanced Virtual Component Interface (AVCI). All these interfaces are compatible with each other. The AVCI is a superset of BVCI, and the PVCI is the subset of BVCI. The BVCI, which has request handshake protocol and response handshake protocol, is the complete communication behavior. It is a split-transaction protocol. The PVCI only has one control handshake, so the request and response signal only can be controlled separately. The AVCI is also a split-transaction protocol. It can deal with the interleaved and transaction reorder.

The VCI used in the bus system must have a ‘wrapper’ between the interface and the bus, which is shown in Fig.1. The advantage of using the VCI is that it can improve the portability of the IP. The IP provider doesn’t need to decide the environment what it used and doesn’t need to know the interconnection type. On the bus system the IP only need to connect to a bus wrapper.

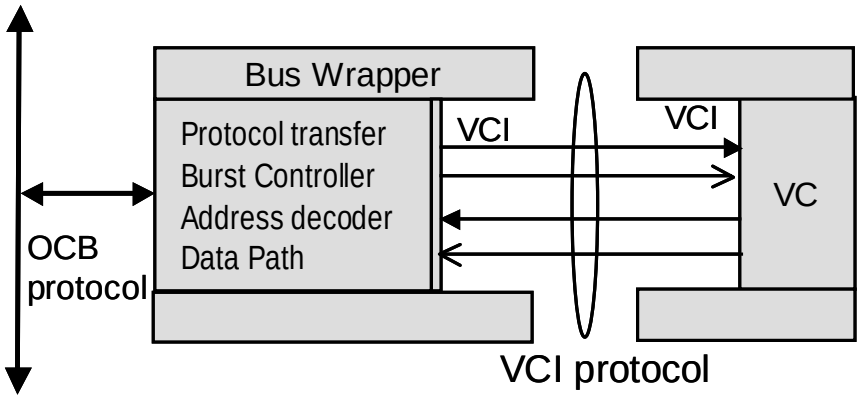

Fig1. A basic VCI-based bus wrapper

2.2 The AMBA High Performance Bus Standard

The AHB is a new generation of AMBA bus. The feature of AMBA AHB is a high-performance and high clock frequency system including burst transfers, split transactions, single cycle bus master handover, single clock edge operation, non-tristate implementation, and wider data bus configuration. Fig.2 shows the typical AMBA bus system architecture. [3]

The main AHB request transfer types are IDLE, BUSY, NONSEQ, and SEQ. The IDLE state means no operation is in the bus system. The BUSY state means the master isn’t ready to transfer data. The NONSEQ state means the first data is being transferred in a burst mode. The SEQ state means the sequence data is being transferred in the burst mode. The AHB response signals are OKEY, ERROR, RETRY, and SPLIT. The OKEY signal tells the bus master the data transfer is in controlled. The ERROR signal means something error in the bus slave. The RETRY and SPLIT signals tell the bus master that the same data should be transferred afresh.

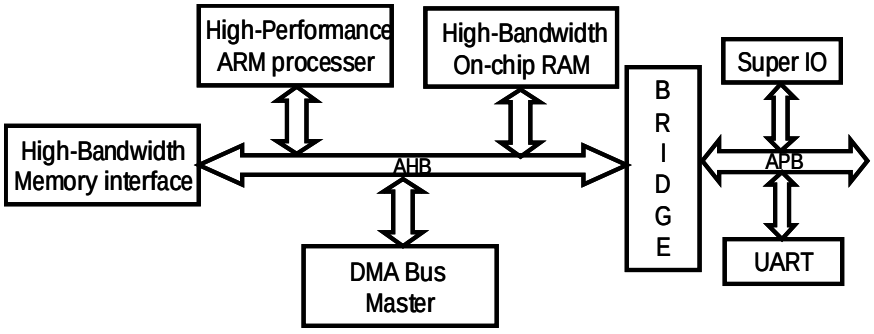

Fig2. A typical AMBA AHB-based system

2.3 The PCI Bus Standard

The Peripheral Component Interconnect Standard defines a board-level bus interface for many I/O devices to connect to the processor, main memory, etc, like as Fig. 3. The PCI standard is a split-transaction protocol based on two types of transactions, posted and delayed. Posted transactions complete at the originating device before they reach their ultimate destination. Delayed transaction termination is used by target that cannot complete the initial data phase within the requirements of the specification. The target independently completes the request on the destination bus using the marker to indicate the delayed transaction completion or not. [4-5]

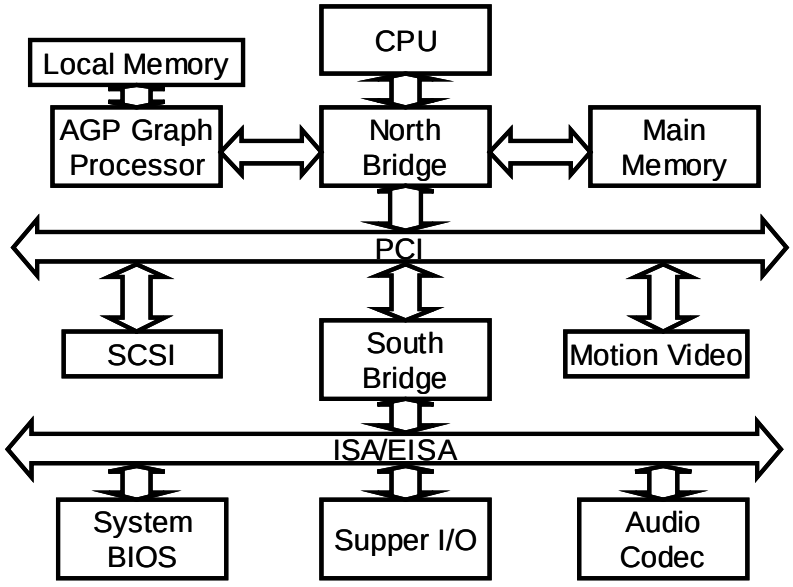

Fig3. PCI System Block Diagram.

3. Bus Wrapper Design Models

In order to speed up the design of the bus wrapper, we must find a methodology to let the bus wrapper design more and more easy. We must convert the protocol between the standard interface and the bus interface.

In the paper [6], it proposed the interface generation process. Communication synthesis is an integral step in a system design methodology. It begins with a partitioned specification, wherein various components or behaviors entities interact and communicate via shared variables from the un-partitioned specification. The goal of communication synthesis should be to produce a complete specification and must describe the structure and functionality of the system. Hardware objects in the specification should be synthesizable and ready for hand-off to available synthesis tools. Figure 4 shows the communication synthesis flow.

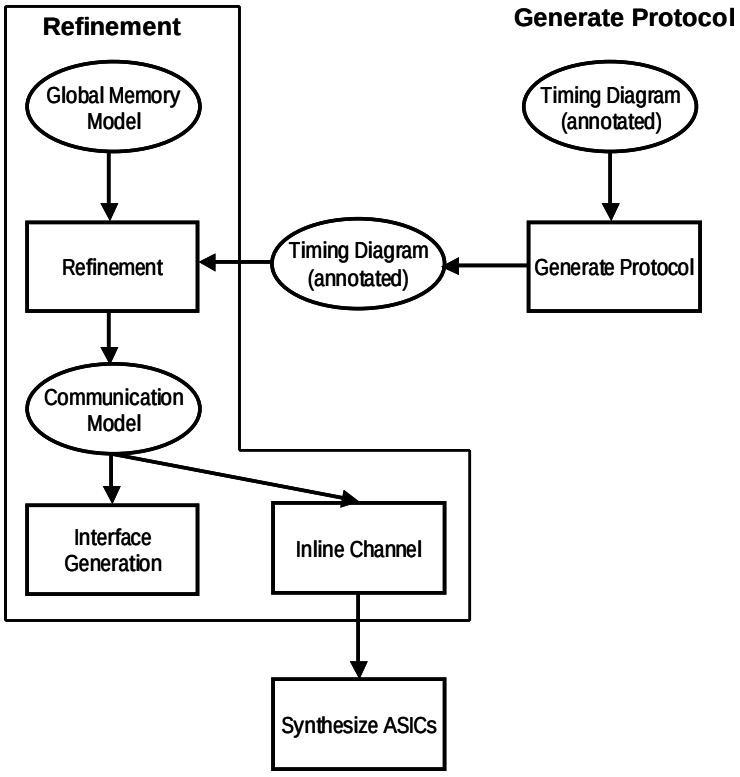

Figure 4 Communication Synthesis Flow

Communication synthesis consists of five main tasks:

1. Generate protocol: the first step in communication synthesis is the generation of protocol. It used the annotated timing diagram to describe the given protocol. The annotated timing diagram must contain the causal relationship of the wave which is show in the figures 5(a) and figures 5(b).

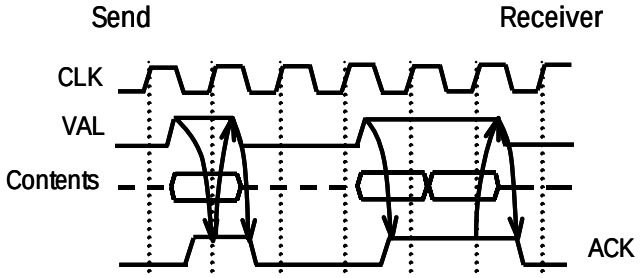

Figure 5(a) Annotated Timing Diagram (Synchronous)

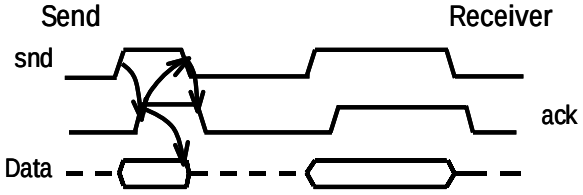

Figure 5(b) Annotated Timing Diagram (Asynchronous)

2. Channel refinement: starting from a partitioned behavioral specification with components communicating through shared global variables. It must use an abstract communication channel to transmission the shared variables by a specified protocol. This process is referred to as channel refinement. These channels can either be containing the description for a simple bus and handshake protocol, or they may describe complex communication via PCI, AHB or some such standard protocol. The designer may wish to merge channels based on width, variable lifetime, access frequency or etc.

3. Interface generation: if we use the previously designed components, we must generate a interface conversion between the different protocol of the components by the form of transducers. It use the dual, ordered relations between opposing signal groups. It is very similar to our bus wrapper

design. We must generate the signals of the interface to satisfy the protocols of either component it is linking. This object can be realized as a FSM with other synthesizable component. [7-8]

4. Inline channel: this is used in an undefined component. The communication functionality of the component can be allocated to certain interface protocol. This communication behavior can the be handed-off to be synthesized with the rest of the component's functional behavior.

5. Synthesize ASICs: after protocols have been inlined and transducers have been generated, synthesis of the components, including interface components, may be completed using current high-level synthesis techniques.

The interface generation process is intended to develop interface transducers between fixed components with differing protocols. It is similar our bus wrapper design type. We also need a transducer to let the different protocol together. The bus wrapper is used to transfer the interface between different protocols. It makes different hardware can communication with each other. The control flow is the main kernel of bus wrapper designs, and the data flow often consists of buffers or FIFOs.

In order to perform transducer, the protocols must present by scheduled signal assignments.

```
AHB Protocol

While (1)
begin
    wait until (HBUSREQx=1 && arbiter_grant)
// arbitration
    wait until (clk=posedge)
        HGRANTx=1;
end

While (HGRANTx=1)
begin
    wait until (HTRANS=NONSEQ)    //transfer
the first address
        i=1;
        HADDR_var_AHB[1]=HADDR[1];
    If (data_done==1)
        HREADY=1;
        HRESP=OKEY;
    else
        error_control();
    end

    while (burst_AHB!=complete)
    begin
wait until (HTRANS=NOP or SEQ or BUSY)
    if (HTRANS=NOP)
        break;
    elseif (HTRANS=SEQ)
    begin
        HADDR_var_AHB[i+1]=HADDR[i+1];
        HWDATA_var_AHB[i]=HWDATA[i];
        HRDATA[i]= HRDATA_var_AHB[i];
        If (data_done==1)
            HREADY=1;
            HRESP=OKEY;
        else
            error_control();
        i=i+1;
        count=1;
    end
    elseif (HTRANS=BUSY)
    begin
        HADDR_var_AHB[i+1]=HADDR[i+1];
        HWDATA_var_AHB[i]=HWDATA[i];
        HRDATA[i]= HRDATA_var_AHB[i];
        If (data_done==1)
            HREADY=1;
            HRESP=OKEY;
        else
                error_control();
            end
        end
        wait until (clk=posedge)
    end
    HWDATA_var_AHB[i]=HWDATA[i];
//transfer the last data
    HRDATA[i]= HRDATA_var_AHB[i];
    If (data_done==1)
        HREADY=1;
        HRESP=OKEY;
    else
        error_control();
    end
    wait until (clk=posedge)
end
```

The AHB Protocol scheduled signal assignments

```
BVCI Protocol
While (1)
begin
    while (burst_VCI!=complete)
    begin
        CMDVAL=1;
        ADDRESS[i]= ADDRESS_var_VCI;
        WDATA[i]= WDATA_var_VCI;
        EOP=0;
        i=i+1;

        wait until (CMDACK=1)

        wait until (clk=posedge)
    end

    CMDVAL=1;
    ADDRESS[i]=ADDRESS_var_VCI;
    WDATA[i]=WDATA_var_VCI;
    EOP=1;

    wait until (CMDACK=1)

    wait until (clk=posedge)

end

While (1)
begin
    While (burst_VCI!=complete)
    begin
        wait until (RSPVAL=1)
        RDATA_var_VCI =RDATA[i];
        REOP_var=REOP;
        RSPACK=1
        i=i+1;
        wait until (clk=posedge)
    end
    wait until (RSPVAL=1)
    RDATA_var_VCI =RDATA[i];
    REOP_var=REOP;
    RSPACK=1;
    wait until (clk=posedge)
end
```

The BVCI Protocol scheduled signal assignments

```
Interface Process
While (1)
Begin
    Wait until (CMDVAL=1)
    i=1
    Temp1[i]=ADDRESS[i];
    Temp2[i]=WDATA[i]
    EOP_var=EOP;
    i=i+1
    HBUSREQ=1

    wait until (HGRANTx=1)
    wait until (clk=posedge)

    HTRANS=NONSEQ;
    HADDR[i]=temp1[i];

    wait until (HREADY=1 && HRESP=OKEY)
    CMDACK=1;

    wait until (clk=posedge)

    while(burst_VCI!=complete)
    begin
        while (burst_AHB!=complete)
        Begin
            HTRANS=BUSY;
            Wait until (CMDVAL=1)
            Temp1[i]=ADDRESS[i];
            Temp2[i]=WDATA[i]
            EOP_var=EOP;
            HBUSREQ=1

            wait until (HGRANTx=1)
            HTRANS=SEQ;
            HADDR[i]=temp1[i];
            HWDATA[i -1]=temp2[i -1]
            temp3[i-1]=HRDATA[i -1];
            wait until (HREADY=1 &&
HRESP=OKEY)
            RSPVAL=1;
            RDATA[i -1]=temp3[i-1];
            REOP=0;
            i=i+1

            Wait until (RSPACK=1)
            wait until (clk=posedge)
        end
    end

    HWDATA[i -1]=temp2[i -1]

    wait until (HREADY=1 && HRESP=OKEY)
    RSPVAL=1;
    temp3[i-1]=HRDATA[i-1];

end
```

The BVCI to AHB interface process scheduled signal assignments

At first, we must classify the interface signals into four parts:

1. Main communication protocol signals: they can indicate when the address or data is ready, or they can tell the other side to transfer data, such as CMDVAL, CMDACK, EOP…in the VCI standard, or HREADY, HTRANS…in the AHB system, or FRAME#, IRDY#, TRDY#, STOP#…in the PCI system.
2. Burst signals: they contain burst control signals and predictive address signals, and can tell the

other side what state now. PLEN, CLEN… in the VCI standard, or HBURST… in the AHB system.

3. Command signals: they contain read/write command, and error signals, and can tell the other side what state now. CMD, or RERROR… in the VCI standard, or HWRITE, HRSP… in the AHB system, C/BE#, PERR#, SERR#…in the PCI system belong to these signals.
4. Data transfer signals: they are main data buses such as WDATA, RDATA…in the VCI, or HWDATA, HRDATA in the AHB, or AD# in the PCI systems. In the PCI system transaction is always in the burst transfer mode, so it doesn't contain the burst signals.
5. Address decoder signals: the address decoder signals should decode the current address and the next address. It uses some FLAGs to let the target can count the next address. The address decoder signals are like ADDRESS, COUNT…in the VCI, or HADDR, HBURST in the AHB, or AD# in the PCI systems. In the PCI system when the data is transferred, the address always increases four every clock, so it doesn't contain the flags.

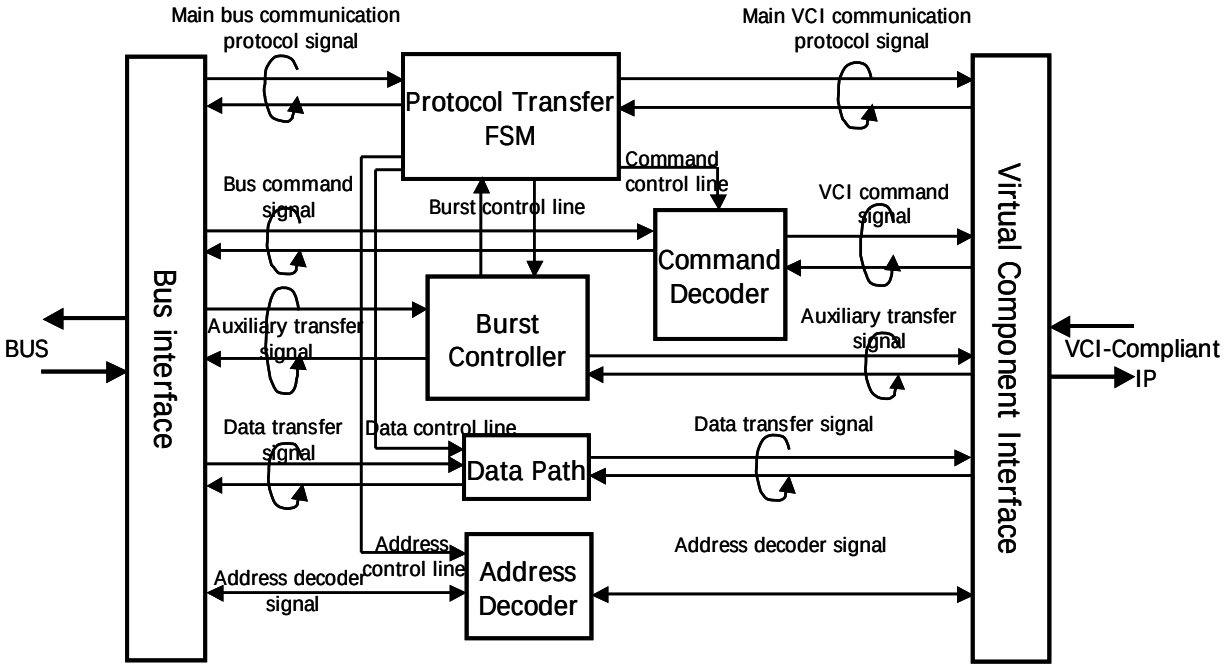

Fig6. The main module of Bus Wrapper

After classifying the interface signals, we can design four modules in the bus wrapper at first. Fig. 6 shows the modules of the bus wrapper.

1. Main Finite State Machine: the FSM is the first design part of bus wrapper. The bus states are usually basis units of bus wrapper states. The input/output signals of VCI interface assist the states transfer. The bus wrapper must correctly transfer both of the signals, and let the communication have no errors in it.
2. Burst Controller: the burst controller deals with the burst transfer. It must decide how to translate the burst signals. If it cannot translate the signals, it must guarantee a single data transfer correctly.
3. Command decoder: it used to translate the command between the standard interface and bus interface.
4. Data flow: The data flow often contains simple buffers when the data width is the same. If the data width is different, the data width must be converted.
5. Address decoder: the decoder can convert the local address and global address. This must depend on the bus architecture. In the AHB system, we only need to keep the high address off. In the PCI system, we must implement the configuration registers.

4. Bus Wrappers

In the bus wrapper, the complexity of control flow is very high, and is the most important part. In order to let the data can pass the wrapper as fast as possible, we try our best to let the signals passing at first time.

4.1 AHB Bus Master Wrapper

In the AHB bus system, the master must produce the bus state, so the wrapper state machine must

react to the bus state.

1. Main Finite State Machine (Fig. 7):

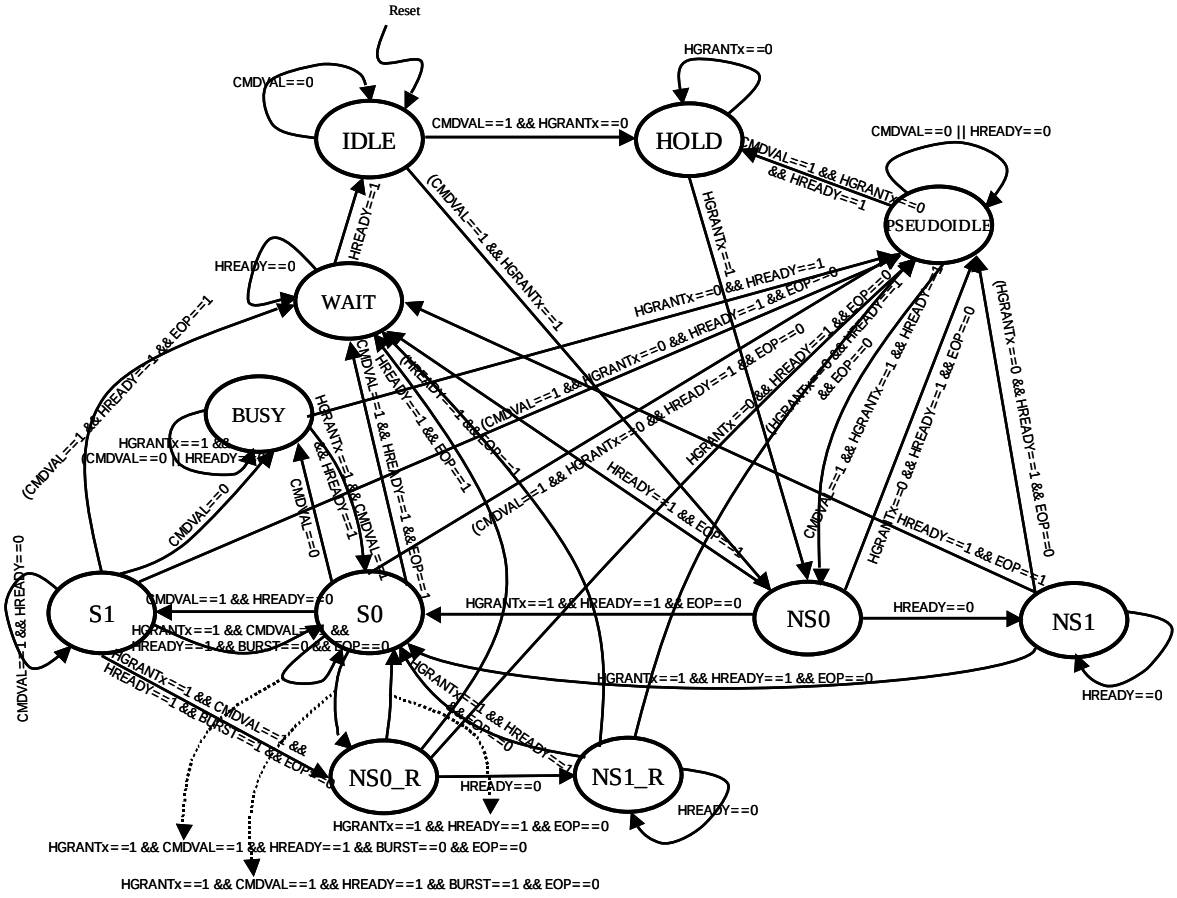

Fig 7. Finite Machine State of AHB Master

IDLE state which is the initial state must keep the signals stable when there are no data transfer in the wrapper. HOLD state is waiting for arbitration, and must sustain the bus request signal. NS0 state and NS1 state mean the AHB bus now is in the NONSEQUENCE state. NS0 state means the data is transfer complete, and NS1 state means the transfer is not ready. S0 state and S1 state mean the AHB bus now is in the SEQUENCE state. S0 state means the data is transfer complete, and S1 state means the transfer is not ready. BUSY state means the AHB bus now is in the BUSY state, and the bus must wait the VCI initiator. WAIT state indicates the final data transfer. RET state and ERR state deal with

the extra situations. The PSEUDOIDLE, NS0_R, and NS1_R are like IDLE, NS0, and NS1. They recover the data transfer when other bus master that has high priory snatches the bus grant.

2. Burst Controller: we use the PLEN signal to indicate the burst numbers by the counter. The “burst” internal signal controls the FSM to decide the transfer that is completed or must go on.

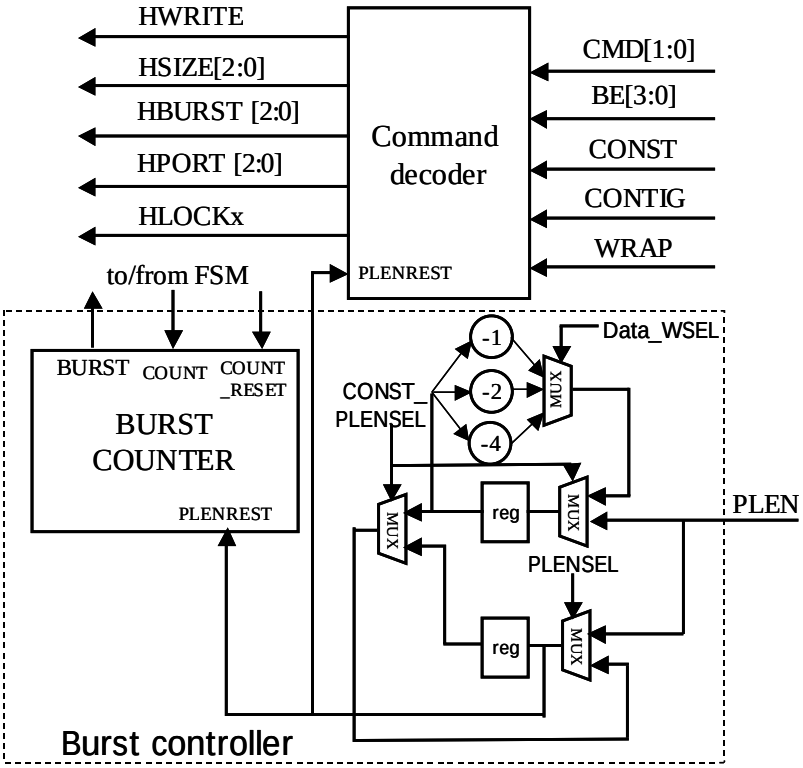

Fig 8 the burst controller of AHB master wrapper

3. Command decoder: we translate the CMD signal of VCI to the HWRITE of the AHB. It indicates the read/write command signal. The burst controller and the command decoder is as shown in the figure 8.

4. Data flow: because our choices are the same data width, we only use simple buffers to translate the data bus.

5. Address decoder: because the AHB bus system has easy address decoder, we only need to keep the high address off, and need no other calculated logic circuit.

Fig 9 shows the final wrapper architecture of AHB master.

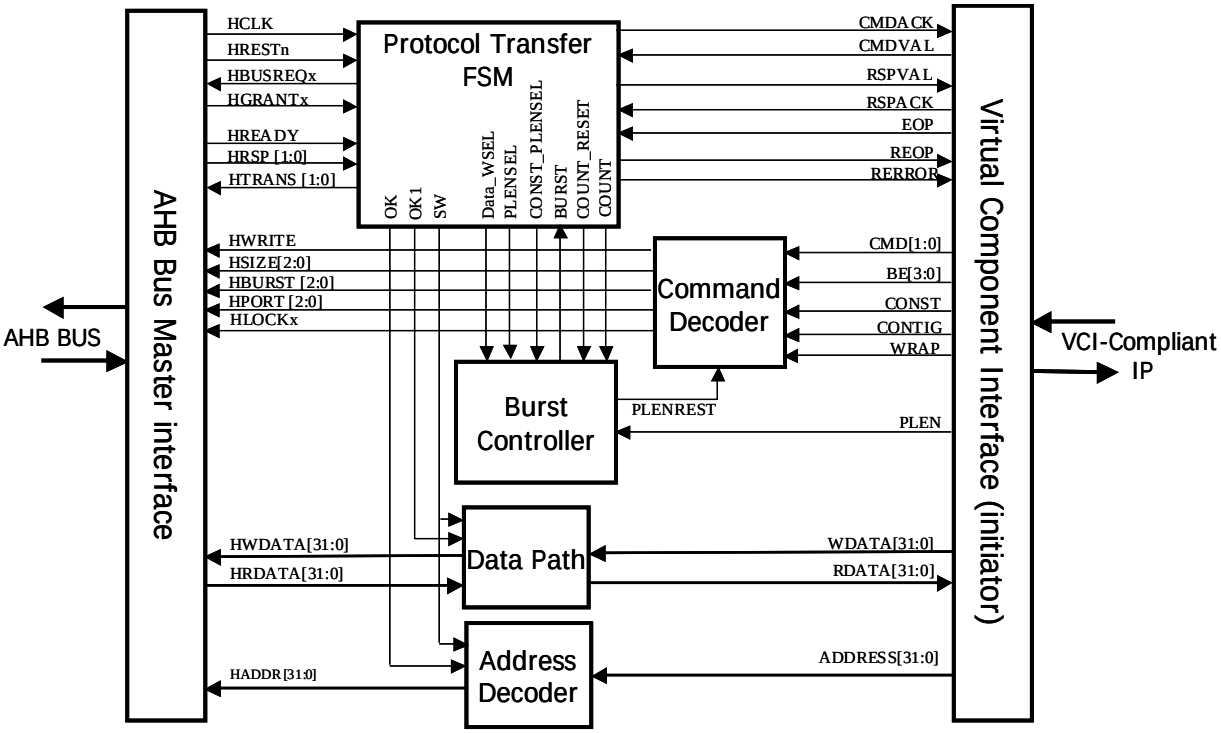

Fig9. The wrapper architecture of AHB Master

4.2 AHB Bus Slave Wrapper

In the AHB bus system, the slave must responses the requirements in the master and the situations in the slave.

1. Main Finite State Machine (Fig. 10):

IDLE state which is the initial state must keep the signals stable when there are no data transfer in the wrapper. HOLD state is waiting for CMDACK in the VCI, and is in the AHB bus NONSEQUENCE state. VAL state enters to the AHB bus SEQUENCE state, and continues to receive or transfer the data. BUSY state means the AHB bus now is in the BUSY state, and the bus must wait the master. END state indicates the final data transfer. RET state and ERR deal with the extra situations.

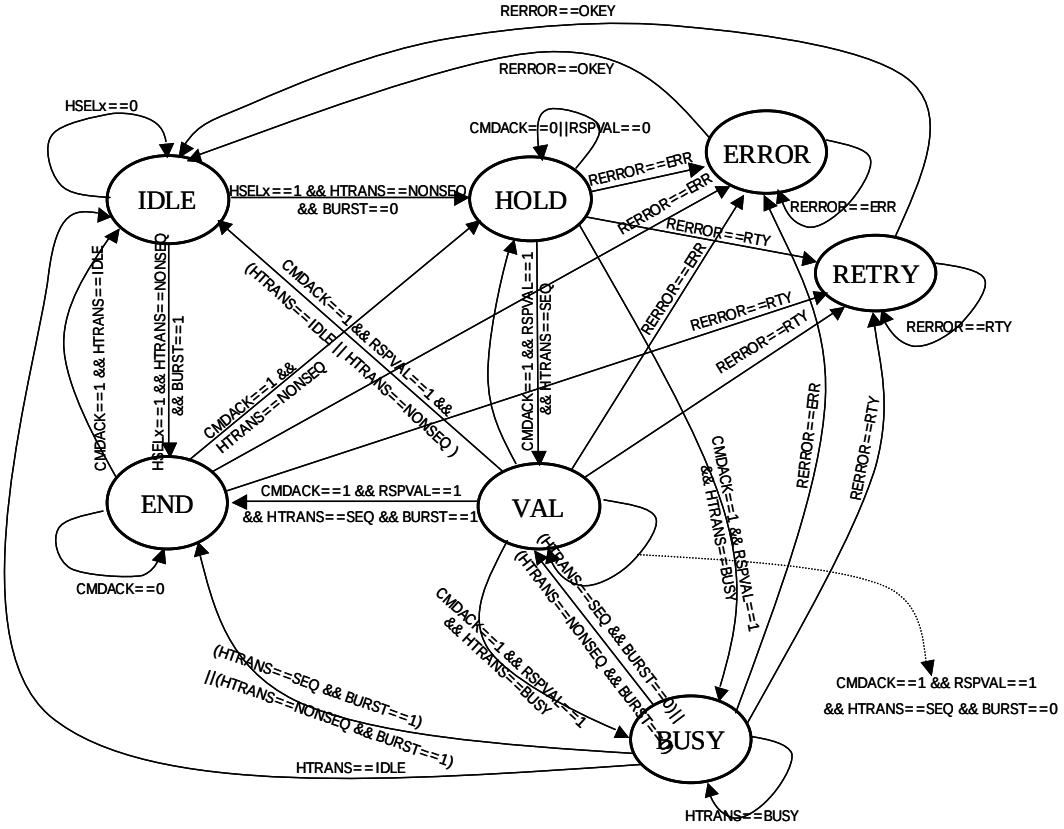

Fig 10. Finite Machine State of AHB Slave

2. Burst Controller: we use the HBURST to translate to PLEN signal. Because the HBURST has fixed burst numbers such as 16, 8, 4, 1, we can easily translate the burst communications.

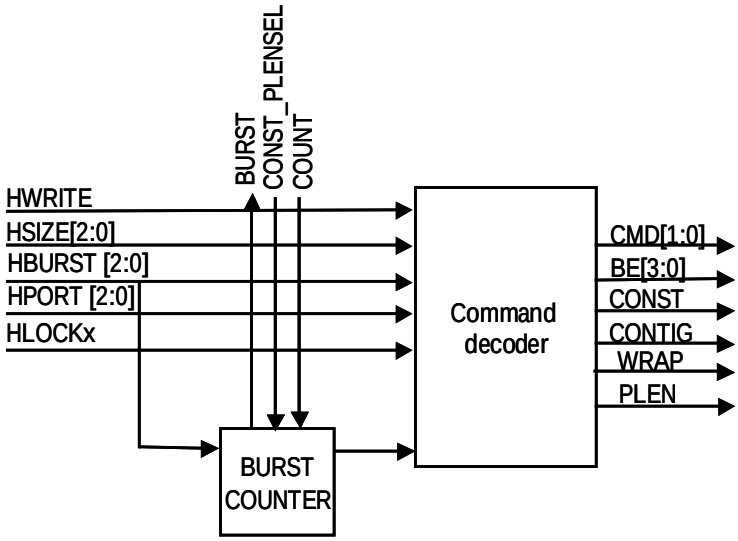

Fig 11. the burst controller of AHB slave wrapper

3. Command decoder: we translate the HWRITE of the AHB to the CMD signal of VCI. It indicates the read/write command signal. The burst controller and the command decoder is as shown in the figure 11.

4. Data flow: because our choices are the same data width, we only use simple buffers to translate the data bus.

5. Address decoder: because the AHB bus system has easy address decoder, we only need to keep the high address off, and need no other calculated logic circuit.

Fig 12 shows the final wrapper architecture of AHB slave.

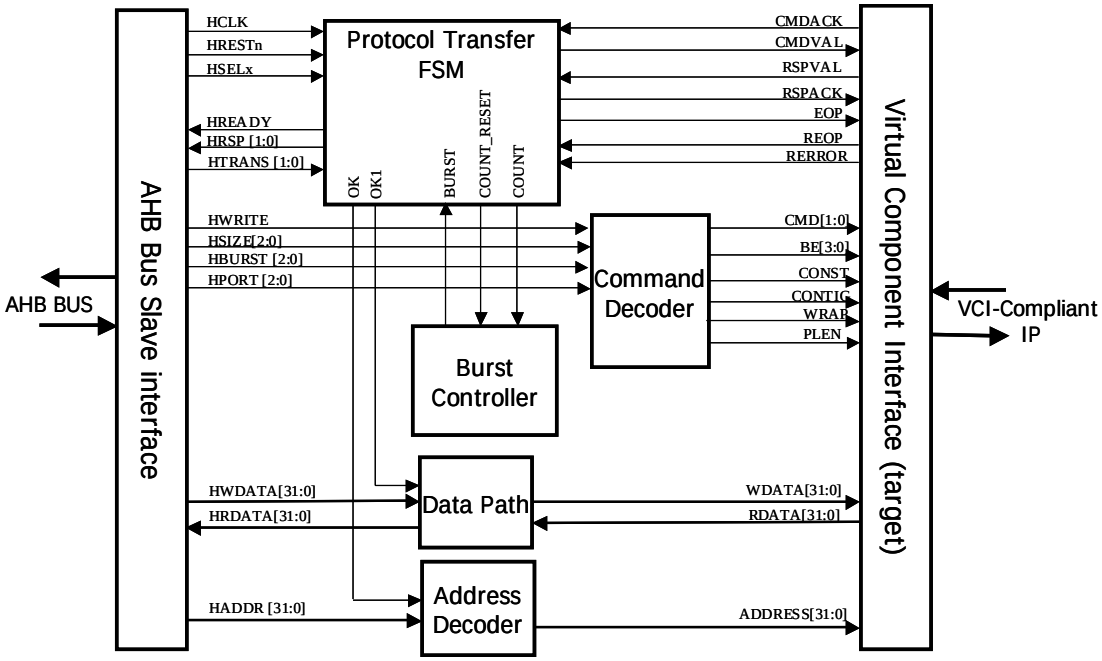

Fig. 12 The wrapper architecture of AHB

4.3 PCI Bus Target Wrapper

In the PCI bus system, the slave must decoder the AD# line to ensure what transfer on the bus system is, and response the transfer.

1. Main Finite State Machine (Fig. 13):

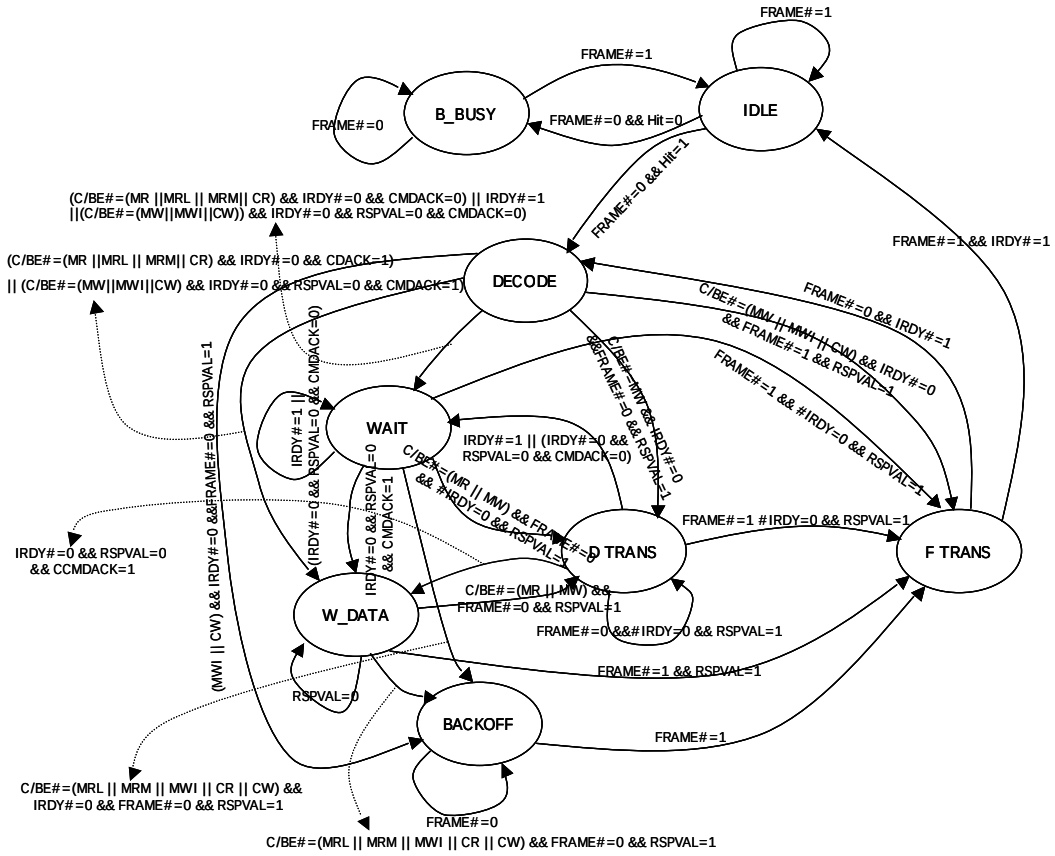

Fig13. Finite Machine State of PCI target

IDLE state which is the initial state must keep the signals stable when there are no data transfer in the wrapper. The B_BUSY state means the bus is already in, which is used to prevent the transfer error. DECODE state decodes the command and begin a burst transfer. WAIT state is waiting for address or turnaround state for charging the capacitance. WAITDATA state is waiting for data transfer. BACKOFF state represents the PCI target interrupt the transmission. D_TRANS state transfers the data. F_TRANS state indicates the final data of the transfer.

2. Burst Controller: because the PCI transactions don't have the fixed burst length, so we cannot implement the burst length counter.

3. Data flow: the PCI bus system combines the address line and data line, so we must use the tri-state buffer to separate the address and data bus.

4. Address decoder: the address decoder must have a configuration register (CR). At the PCI system

starting up, the operation system (OS) will read the CR, and write the address space. The PCI target will use the CR to locate their memory or I/O address. The architecture of the address decoder is as shown in the figure 14.

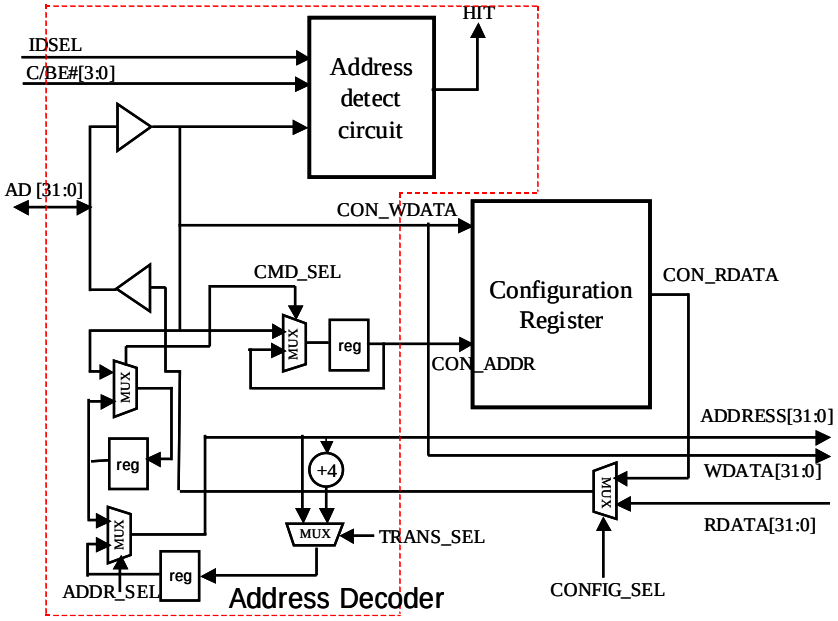

Fig 14 Configuration Register and Address Decoder

Fig 15 shows the final wrapper architecture of PCI target.

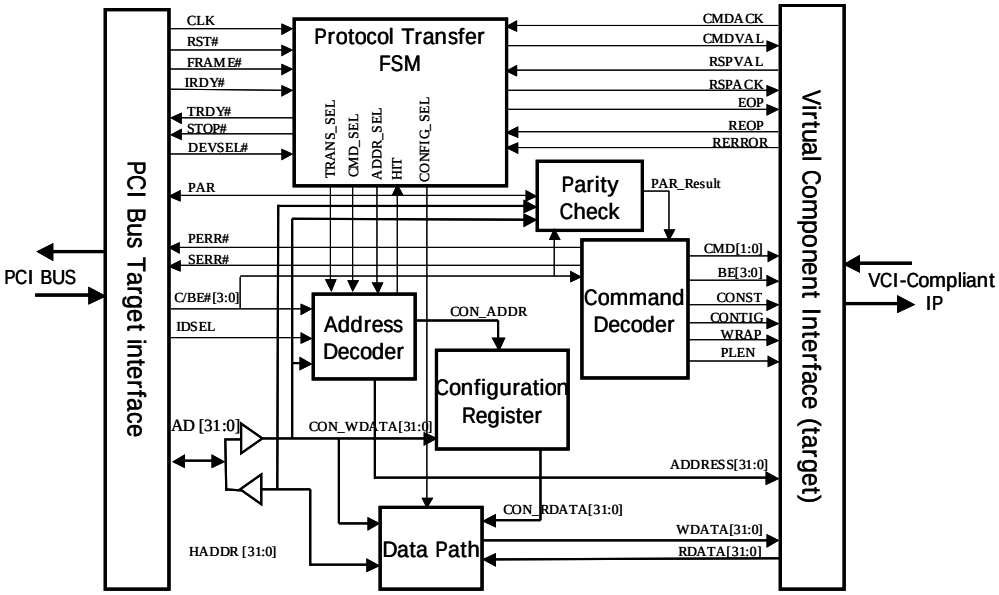

Fig.15 The wrapper architecture of PCI target

5. Conclusion

The current trend in the design domain, the system-on-a-chip is the main method, so the use of the IP will become more and more popular. How to build the IP supermarket is the world trend in the future. The interface of the system is a leading role of the SoC. It is very convenient for system integration. The system designer only needs to plan the system architecture rather than interface conversion.

We introduce the Virtual Component Interface Standard for our standard IP interface. The popular on chip bus system, Advanced High-Performance Bus (AHB), is our first goal to convert between VCI. The second we convert the VCI between the PC system by Peripheral Interconnect Bus (PCI) which is the on board bus.

We proposed the bus wrapper design methodology with interface protocol conversion. By this methodology we can convert the different interface and different protocol using system design method. We also implement the on-chip-bus wrapper and on-board-bus wrapper. It will be used in bus-based SoCs for IP integration. We can use the AHB wrapper to integrate the VCI-compliant IP into ARM development system, or use PCI wrapper to integrate the VCI-compliant IP into personal computer system. It will cause the system integration more and more easily.

Reference

[1] On-Chip Bus Development Working Group. “Virtual Component Interface Standard Version 2”, April 2001.

[2] Geneviève Cyr, Guy Bois, Mostapha Aboulhamid, Jacques Baillairgé. “Synthesis of communication interfaces using VSIA recommendations” Proc. of DATE 2001, Munich, Allemagne/*Germany*, 03/2001.

[3] “AMBA™ Specification Revision 2.0”, May 13,1999.

[4] “PCI system architecture fourth edition” by MindShare Inc., Tom Shanley and Don Anderson.

[5] PCI Special Interest Group, “PCI Local Bus Specification Revision 2.2” December 18, 1998.

[6] J. D. Kleinsmith and D. D. Gajski, “Communication Synthesis for Reuse”, Technical Report ICS 98-06, University of California, Irvine, February 1998.

[7] S. Narayan, D.D. Gajski. “Interfacing System Components by Generation of Interface Processes.” Proceedings of the 32nd Design Automation Conference. June 1995.

[8] J. Akella and K. L. McMillan, "Synthesizing Converters Between Finite State Protocols", IEEE International Conference on Computer Design: VLSI in Computers and Processors, Cambridge, MA, USA, 14-16 Oct. 1991, pp. 410-13.