Design and Implementation of VHDL Model for PCMCIA Memory Card Controller

Dr. Mohammad N. Abdullah* Yousra A. Mohammed**

Received on 23/11/2004
Accepted on 7/9/2005

Abstract
In this paper, a design and implementation of synthesizable VHDL model for control system (Controller) of the Personal Computer Memory Card International Association (PCMCIA) cards, which are called memory PC Cards (SRAM and FLASH), is presented.
The system software for this controller has been developed using OrCAD program based on VHDL(Very high speed Hardware Description language) as software environment and Xilinx Spartan-XL (XCS10XL-4 TQ144) FPGA (Filed Programmable Gate Array) chip as hardware device, so the development becomes more reliable, modular, and adaptable to new requirements.
The timing behavior of the controller is verified to ensure that it meets the performance requirements by using timing simulation tools of OrCAD program, therefore complete simulation results of read/write transfers for both an attribute and a common memories are presented in this paper.

Key word: PCMCIA memory cards, VHDL,FPGA.

تصميم نموذج VHDL و تنفيذه لمسيطر بطاقات الذاكرة نوع PCMCIA

الخلاصة
في هذه المقالة صمم (VHDL Model) ونفذ لنظام سيطرة (مسيطر) لمجموعة من البطاقات الأكثر أهمية لهذا الناقل التي تدعى بطاقات الذاكرة.
تطوير برمجيات النظام لهذا العمل يشمل عملية التصميم و التنفيذ باستخدام برنامج (OrCAD) المستند على لغة (VHDL) كبيئة برمجية و رقاقة المصفوفات البرمجية نوع (XCS10XL-4 TQ144) التي تنتجها شركة (XILINX) كأداة للكيان المادي، و بذلك يكون التصميم موثوقاً اكثر، مرناً، و قابل للتحديث لمتطلبات جديدة.
أن ألـ(timing behavior) للمسيطر قد تم تحليله للتأكد من انه يسد حاجات الأداء المُصمم لها باستخدام أدوات محاكاة الوقت لبرنامج ألـ(OrCAD)، لذلك تم تقديم في هذه المقالة بعض من نتائج المحاكاة لعمليات القراءة و الكتابة للذاكرة العامة و الذاكرة التعريفية.

Introduction

Memory cards were introduced as an alternative to the mechanical floppy and floppy drive. In other words, memory cards are used to emulate the floppy drive sub-system.

Memory cards implemented, as virtual floppy disks must also have the ability to be inserted and removed from the system at any time (as with floppy disks). This ability is accomplished through the PC Card's hot swapping capability. The system automatically detects and configures the PC Card allowing the system to

* Dept.of Computer Science, University of Technology
**Dept.of Control & System Eng., University of Technology

https://doi.org/10.30684/etj.24.6.7
University of Technology-Iraq, Baghdad, Iraq/2412-0758

treat it as a floppy disk. That is, once a PC memory card is inserted into a socket, the user can read files or write files just as a floppy drive. The only perceptible difference is the lightning speed at which these operations occur when compared to the speed of the same operations performed with a floppy drive. This is due to the much faster access to solid state media. In addition to very fast access, these virtual drives provide a data transfer medium less susceptible to harsh environmental conditions when compared to magnetic media [1, 2].

There are many types of memory cards such as, DRAM, pre-programmed ROM, SRAM and Flash. These types are not interchangeable but they can be used as data storage devices. However that memory cards are available from a number of different manufacturers though the basic structure is always the same. The basic components of these cards are [2, 3]:

- Attribute Memory: - the PCMCIA memory card must have attribute memory to support CIS (Card information Structure).
- Common Memory (Memory Array): - data files and application programs can be stored on the common memory for the applications of all kinds of portable computers.
- Controller (Address Decode and Control Logic): - which controls common and attribute memory read/write transfers. This part of memory card is the more important component with respect to this paper.

Write Protect Switch: - Which is used for data security.

In addition, memory cards that contain volatile memory devices, such SRAM cards usually use a battery for data retention.

In this paper, PCMCIA controller for memory card is proposed as the hardware approach by using VHDL language based on FPGA.

The Proposed PCMCIA Memory PC Card Controller Design

The proposed controller was implemented for SRAM and FLASH PCMCIA memory card. The controller contains a complete address and data buffer, address decoder, memory device selection logic, and read and write control logic for common and an attribute memory. Eight chip enable outputs are provided, supporting 16 memory devices.

Since the memory cards can be operated in either an 8-bit or 16-bit wide mode which allows flexible integration into various system platforms which conform to PCMCIA standard, this controller is designed to operate for both 8-bit and 16-bit modes.

The Proposed PCMCIA Memory Card Controller Operation

The controller is used to implement PCMCIA typ I compatible memory cards as shown in the system block diagram and in the internal chip block diagram (FPGA) in Fig.(1). Both PCMCIA signal lines and memory devices are connected directly to the controller with no additional components required. Active chip select lines (CS# [7:0] are determined by Address lines from [25:23] as shown in table (1).

The controller provides separate output and write enable for the upper, lower bytes of the memory array (COEH#, CWEH, COEL# and CWEL#) and for the attribute memory (CISOE and CISWE) to implement byte addressing. The assertion of these outputs under the A0, CE1#, CE0#, OE# and WE# is given by table (2) Read-Write Control Generation when REG# is high and when REG# is low.

The controller supports both Common and Attribute Memory read and write cycles of word and byte width. Common memory, the external memory devices on the PC card, are selected when REG# is high and can only be accessed in either byte or word mode. Attribute memory (external EEPROM), is selected when REG# is low and can only be accessed as the even byte.

Byte/word addressing is controlled by CE0#, CE1# and A0. OE# functions as an active low output enable. WE# functions as an active low write enable. Memory access functionality is defined in the table (2).

When Attribute Memory is selected by the assertion of REG#, only the lower data bus, DILOW [7:0] is valid and only even numbered addresses be accessed. Accordingly, an entry of "0" in the REG# of the function table (2) means the access is supported for Attribute Memory. An entry of "1" means the access is supported for Common Memory accesses. During word accesses of Common Memory, DILOW [7:0] and DIHIGH [15:8] are active. During byte Accesses (other than odd Byte Only accesses), the PCMCIA transfers take place on DILOW [7:0] and the controller performs the required byte lane swapping based on A0 to and from DIHIGH [15:8] or DILOW [7:0].

The WPin input provides write protection for common memory. The ATTWP input provides write protect case for the attribute memory when high. These signals are used for applications requiring write protection.

In the case of slow memory devices used, the RDY/BUSY# signal for accessing these Memory devices for writing is used.

FPGA Design Flow

The design flow broadly refers to the sequence of activities encompassing various design tools that begin with some abstract specification of a design and ends with a configured FPGA [4, 5,6].

The design flow described in this section is that of the OrCAD (Organization Computer Aided Design) environment, however most of the activities will have a counterpart in any vendor's design flow. The design flow begins in the design entry phase with the creation of a description of the hardware, typically using either VHDL or schematics, VHDL is chosen as a design entry for controller development. The functionality of the controller entity is defined by behavioral architecture description; Fig. (2) shows the general form of PCMCIA memory PC Card controller design as programming structure and Fig.(3) shows the hierarchy block diagram of this controller. After the design entry has been entered, the first Simulation phase (In Design) is used to debug VHDL design resources. The Compile phase uses synthesis and optimization technology to convert the design description to a netlist of logic functions, the output of this phase is shown in Fig (4). The Build phase uses place and route technology to convert the design to a specific programmable device, from the netlist's delays of this phase it can be noted that the delay of the net "nx4937" is the limiting factor that limits the whole design's speed. However, this delay is little (5.636 ns) as compared with bus speed (10 MHz) so that there is no problem from this delay in the proposed design.

. During the final Simulation phase (Timed) timing simulation verifies that the post-route design meets the performance requirements of the design.

Timing Simulation Results

The results concerned here are the timing diagrams of all types of PC memory cards transactions, i.e. an attribute and common memories read transfers and also write transfers for an attribute and common memories as shown in Fig. from (5) to (8).

From these results (timing diagrams) it can be seen that the worst delay obtained from the controller design is the delay between the input stimulus PCMCIA write (WE#) or output enables (OE#) and the controller write or output enables which are equal to (6ns) for read transfers and (7ns) for write transfers, so these delays are very small value as compared with the delays take place if the controller is designed by using conventional hardware.

Conclusions

The following points are concluded from this work:

1. Using FPGA in the development of the proposed controller enjoys three unique characteristics that can enhance the development and debugging process relative to conventional custom-hardware-based systems:
   a) hardware availability (FPGA hardware is generic and can be used at the earliest point in the design cycle);
   b) programmability (FPGA hardware can be modified throughout the design cycle); and
   c) visibility (the internal state of many FPGAs is directly accessible, so that it is easier to verify designs and track down and fix bugs).
2. The Spartan-Xl (XCS10XL-4) is chosen as a target device chip because it is considered a moderate selection for the number of gates available and speed.
3. The behavioral architecture is chosen as approach in the writing of the design code because this provides very flexible code that can be modified quickly and also provides little or no insight into physical implementation of the entity.
4. As a final note, investigating the previously mentioned results of place and route and timing simulation lead to the fact that the delay between signals in this design is suitable and within the limit of PCMCIA bus speed.

References

[1] Don Anderson and Tom Shanley; "CARDBUS System Architecture"; MindShare, Inc.; 1996.

[2] Don Anderson; "PCMCIA System Architecture/16-BIT PC Cards"; MindShare, Inc.; Second Edition; 1995.

[3] PC/104 consortium; PC/104 PCMCIA Module/User's Manual;

First Edition; Printed in Taiwan; July; 1997.

[4] Sudhakar Yalamanchill; "Introductory VHDL: From Simulation To Synthesis"; Prentice-Hall, Inc.; 2001.

[5] Kevin Skahill; VHDL for Programmable Logic; Addison Wesley Longman, Inc.; 1996.

[6] Mark Zwolinski; Digital System Design with VHDL; Prentice-Hall, Inc.; 2000.

Table (1) Chip Enable Decoder Operation

Address [25:23]	CS# [7:0]
000	11111110
001	11111101
010	11111011
011	11110111
100	11101111
101	11011111
110	10111111
111	01111111

Table (2) Read-Write Control Generation, Byte selection and Data steering functions for Common Memory and Attribute Memory

Mode	REG#	OE#	WE#	CE1#	CE2#	A0	COEL#	COEH#	COWEL#	COWEH#	CISOE#	CISWE#	DHIGH [15:8]	DLOW [7:0]
Output Disable	1 or 0	1	1	X	X	X	1	1	1	1	-	-	High Z	High Z
Standby	1 or 0	0	1	1	1	X	1	1	1	1	-	-	High Z	High Z
Byte read, even (8 and 16-bit modes)	1	0	1	1	0	0	0	1	1	1	-	-	High Z	Even-Byte
Byte read, odd (8-bit mode)	1	0	1	1	0	1	1	0	1	1	-	-	High Z	Odd-Byte
Odd byte only read (16-bit mode)	1	0	1	0	1	X	1	0	1	1	-	-	Odd-Byte	High Z
Word read (16-bit mode)	1	0	1	0	0	X	0	0	1	1	-	-	Odd-Byte	Even-Byte
Byte write, even (8 and 16-bit modes)	1	1	0	1	0	0	1	1	0	1	-	-	High Z	Even-Byte
Byte write, odd (8-bit mode)	1	1	0	1	0	1	1	1	1	0	-	-	High Z	Odd-Byte
Odd byte only write (16-bit mode)	1	1	0	1	1	X	1	1	1	0	-	-	Odd-Byte	High Z
Word write (16-bit mode)	1	1	0	0	0	X	1	1	0	0	-	-	Odd-Byte	Even-Byte
Attribute Memory read	0	0	1	1	0	0	-	-	-	-	0	1	High Z	Even-Byte
Attribute Memory write	0	1	0	1	0	0	-	-	-	-	1	0	High Z	Even-Byte

Note: 1. "X" means don't care condition for A0.

2. Odd-Byte or Even-Byte are data read from or write to odd or even-byte of common and attribute memory.
3. "High-Z" indicates high impedance condition for data bus lines.
4. "1" and "0", these symbols refer to Active-High and Active-Low conditions respectively.
5. "-" this symbol indicates that corresponding signals do not operate in these states.

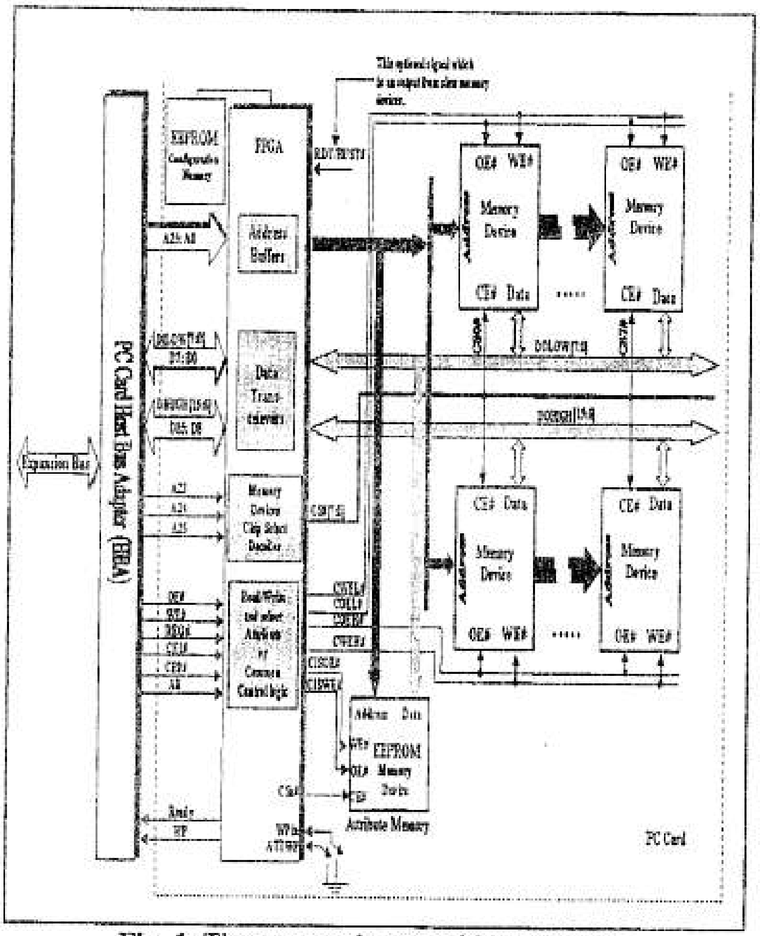


Fig. 1. The proposed system block diagram

```
PCMCIA Memory Card Controller program
Entity Declaration (define the type and mode for all signals)
Signals Initialization
main ()
{
        if (REG# = 0) then
           (Attribute memory is selected for reading or writing
           configuration information from/to card information
           structure (CIS) according to the states of card enable
           signals CE# [1:0], output enable, write enable, A0,
           RDY, and ATTWP signal)
        else if (REG# = 1) then
          (Common memory is selected for reading or writing
          working data that are used by users according to the states
          of card enable signals CE# [1:0], output enable, write
          enable, A0, RDY, and WPin signal)

        end if
}        /* end of PCMCIA Memory Card Controller program*/
```

Fig. 2. The general form of PCMCIA memory PC Card controller design

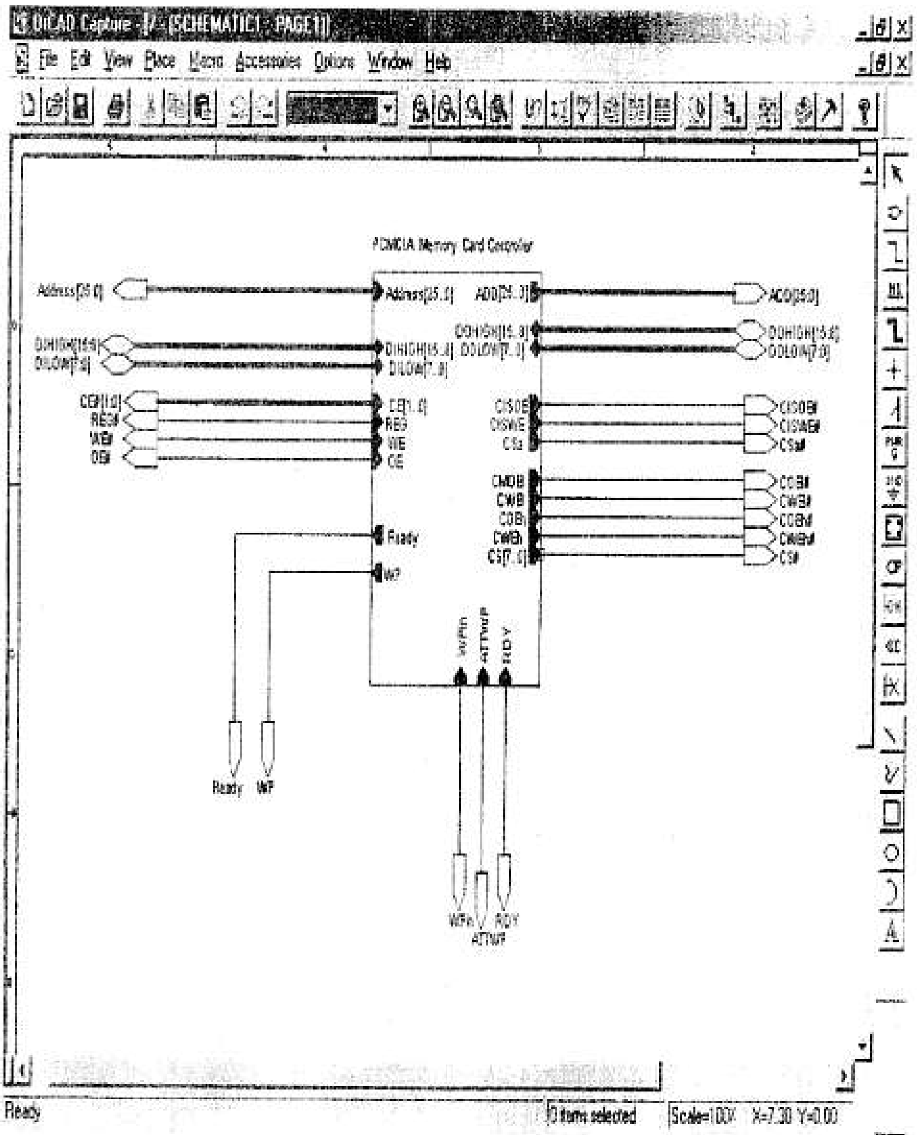
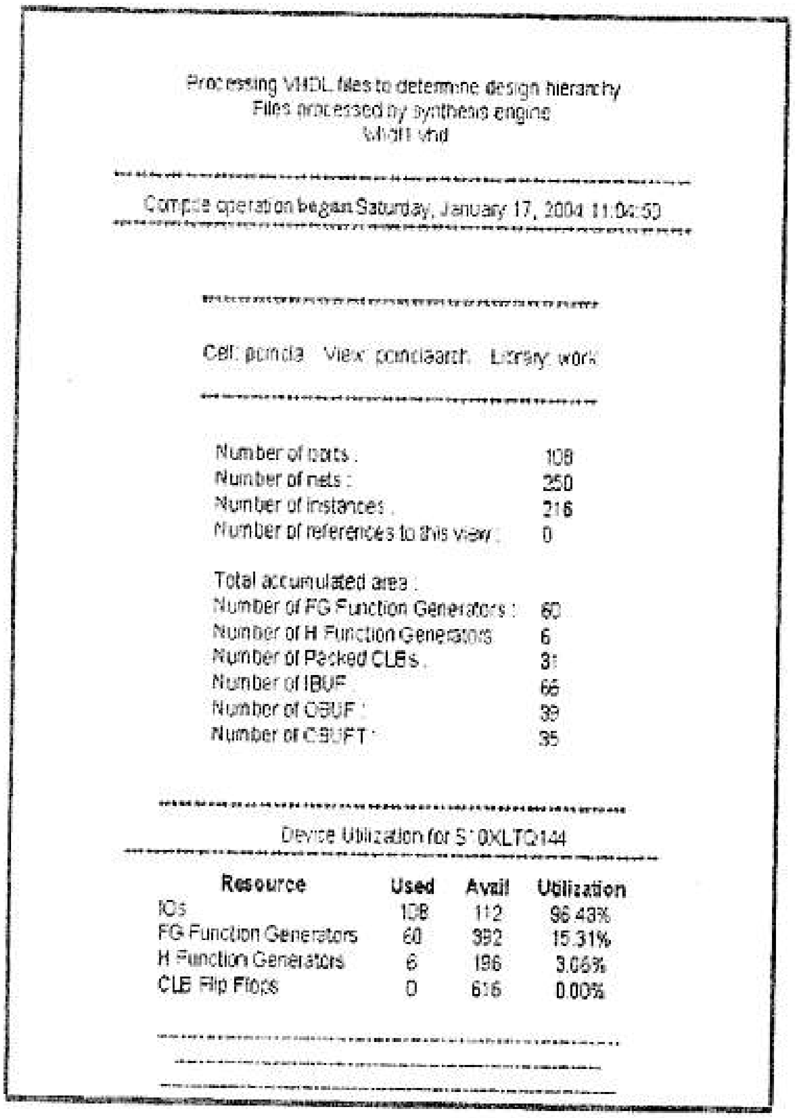


Fig. 3. Controller hierarchy block diagram



```
        Processing VHDL files to determine design hierarchy
                Files processed by synthesis engine
                           WhoH vhd

     Compile operation began Saturday, January 17, 2004 11:04:50


          Cell: pcmcia   View: pcmciaarch   Library: work


          Number of ports :                          108
          Number of nets :                           250
          Number of instances :                      216
          Number of references to this view :        0

          Total accumulated area :
          Number of FG Function Generators :         60
          Number of H Function Generators            6
          Number of Packed CLBs :                    31
          Number of IBUF :                           66
          Number of OBUF :                           39
          Number of OBUFT :                          35


                 Device Utilization for S10XLTQ144

            Resource             Used    Avail    Utilization
      IOs                        108     112      96.43%
      FG Function Generators     60      392      15.31%
      H Function Generators      6       196      3.06%
      CLB Flip Flops             0       616      0.00%
```

Fig. 4. Synthesizer output

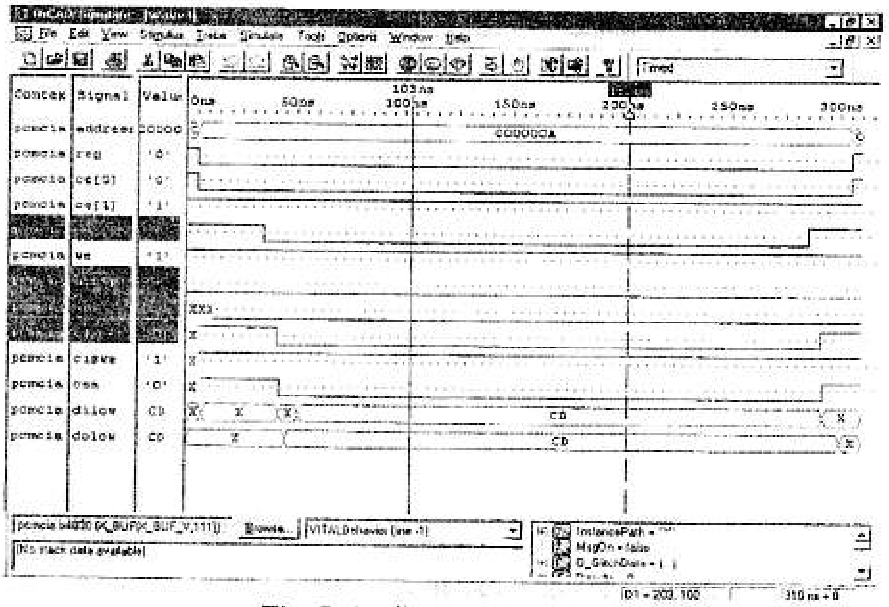


Fig. 5. Attribute Memory Read

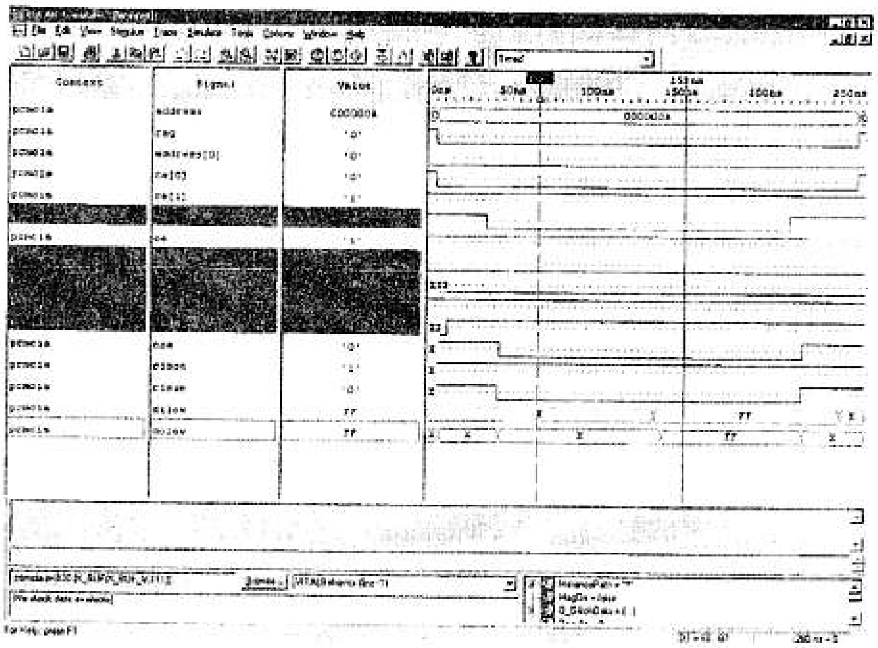
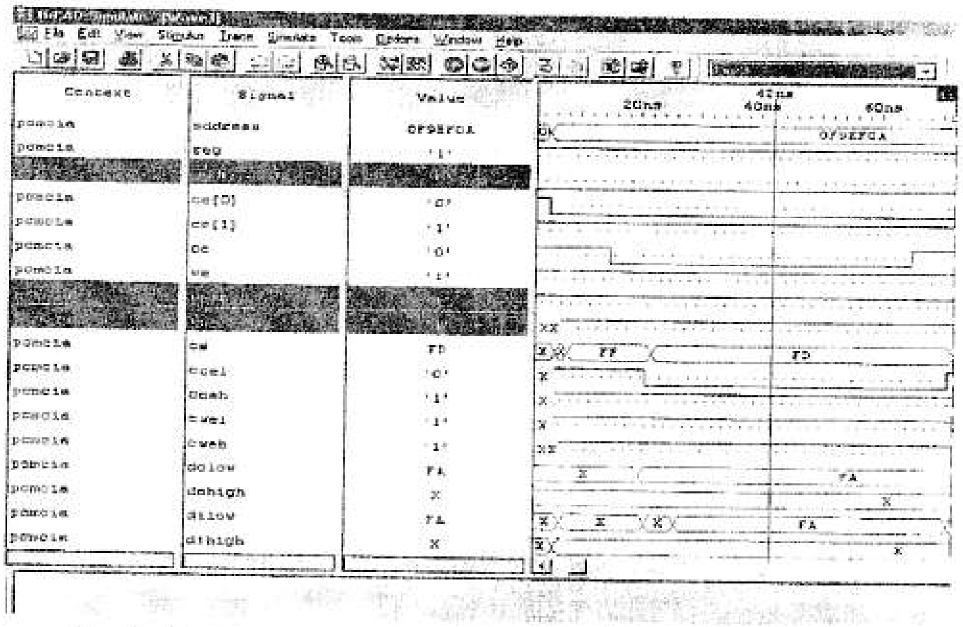
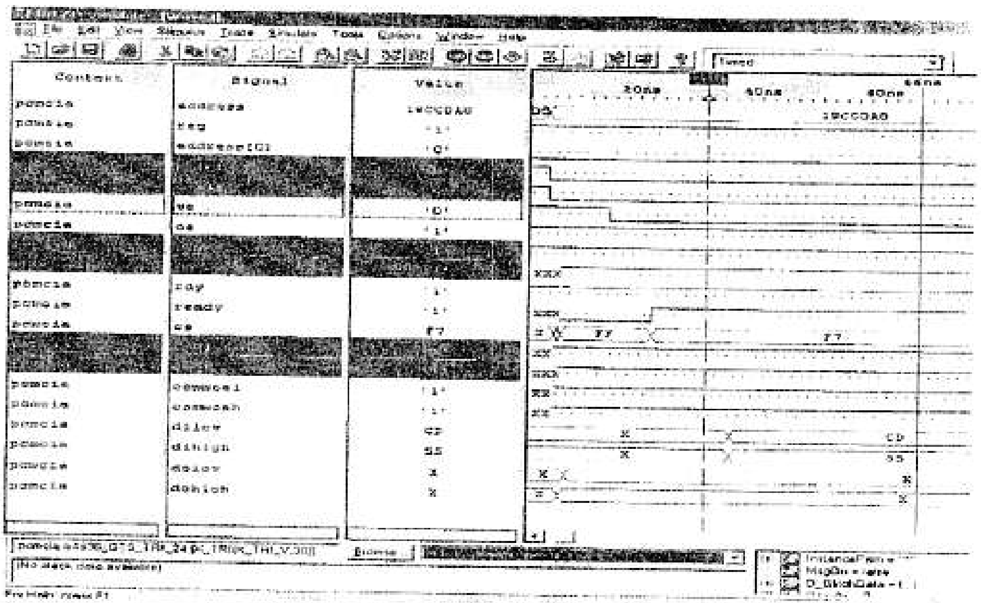


Fig. 6. Attribute Memory Write



Context	Signal	Value
pcmcia	address	0F9EFDA
pcmcia	reg	'1'
pcmcia	ce[0]	'0'
pcmcia	ce[1]	'1'
pcmcia	oe	'0'
pcmcia	we	'1'
pcmcia	cs	FD
pcmcia	ccel	'0'
pcmcia	cceh	'1'
pcmcia	cwel	'1'
pcmcia	cweh	'1'
pcmcia	dolow	FA
pcmcia	dohigh	X
pcmcia	dilow	FA
pcmcia	dihigh	X

Fig. 7. Common Memory Read (Even byte access for 8 and 16-bit modes)



Context	Signal	Value
pcmcia	address	1ACCDA0
pcmcia	reg	'1'
pcmcia	address[0]	'0'
pcmcia	we	'0'
pcmcia	oe	'1'
pcmcia	rdy	'1'
pcmcia	ready	'1'
pcmcia	cs	F7
pcmcia	commcel	'1'
pcmcia	commceh	'1'
pcmcia	dilow	CD
pcmcia	dihigh	55
pcmcia	dolow	X
pcmcia	dohigh	X

Fig. 8. Common Memory Read (Even byte access for 8 and 16-bit mode