CPLD IMPLEMENTATION OF 8255 PROGRAMMABLE PERIPHERAL INTERFACE

A dissertation submitted in partial fulfillment of the requirement for the award of

MASTERS DEGREE
In
CONTROL & INSTRUMENTATION ENGINEERING

By

SUMIT MISHRA
Enrollment No:10/C & I /2002
RollNo: 4351

Under the guidance of

Prof. PARMOD KUMAR

DEPARTMENT OF ELECTRICAL ENGINEERING
DELHI COLLEGE OF ENGINEERING
NEW DELHI – 110 042

DEPARTMENT OF ELECTRICAL ENGINEERING
DELHI COLLEGE OF ENGINEERING

CERTIFICATE

This is to certify that this report entitled, "**CPLD IMPLEMENTATION OF 8255 PROGRAMMABLE PERIPHERAL INTERFACE** ",submitted by , Sumit Mishra in the partial fulfillment of the requirement for the award of **Masters Degree** in **Control and Instrumentation Engineering** , embodies the work done under my supervision.

Prof. Parmod Kumar
Head of Department
DCE, Delhi.

Date:

ACKNOWLEDGEMENT

With deep sense of gratitude I express my sincere thanks to Prof. Parmod Kumar(Head of the department Electrical engineering) Electrical Engineering department ,Delhi College Of Engineering, Delhi ,for his constant encouragement and guidance rendered during execution of this project work. I am thankful to him for extending time to time support and for providing necessary facilities .Without wise counsel and able guidance of Prof. Parmod Kumar ,it would have been impossible to complete the project in this manner.

I am also thankful to Prof. B.N. Mishra Chairman Logic Eastern India Pvt. Ltd for providing me necessary facilities for completion of this work.

Sumit Mishra

CONTENTS

1 INTRODUCTION

1.1 Introduction to 8255ppi

The 8255 is a programmable peripheral interface which is used for parallel data transfer / acquisition. It is used as a general-purpose device for interfacing parallel I/O devices to the system data bus. It has three 8-bit ports: Port A, Port B, and Port C, which are arranged in two groups of 12 pins. The 8255 can be programmed to operate in three modes: Mode 0, Mode 1, and Mode 2. Each port has a unique address, and data can be read from or written to a port, when the 8255 is interfaced in I/O Mapped I/O Mode, by issuing either an IN or OUT instruction respectively. In addition to the addresses assigned to the three ports, another address is assigned to the control register into which control words are written for programming the 8255 to operate in various modes. The 8255 is generally assigned four consecutive addresses in the I/O space.

1.2 Motivation for thesis :

Data acquisition is the most important part of signal processing. The features of 8255 PPI are suitable for multi channel data acquisition system and for sending control signal to the field instruments .It is a single entity and it requires additional circuitry like Digital to Analog and Analog to Digital Converter to be able to accept data from the field

and to send analog control signals if required as per the case be. This complete setup requires considerable board area. If we design and implement it on a Programmable Logic Device Like CPLD the area requirement will be reduced to a single Integrated Chip i.e. the PLD itself. This is possible because contemporary Programmable Logic Devices have high gate density and higher operating clock frequency, so the speed aspect is also taken care of. Secondly if we ever there be requirement to change the architecture the PLD can be reprogrammed without affecting the pin locking already in place(ISP-In System Programmability).

2 VHDL

2.1 Introduction

VHDL is an acronym for VHSIC Hardware Description Language (VHSIC is an acronym for Very High Speed Integrated Circuits). It is a hardware description language that can be used to model a digital system at many levels of abstraction, ranging from the algorithmic level to the gate level. The complexity of the digital system being modeled could vary from that of a simple gate to a complete digital electronic system, or anything in between. The digital system can also be described hierarchically. Timing can also be explicitly modeled in the same description. The VHDL can be regarded as in integrated amalgamation of the following languages :

sequential language +

concurrent language +

net-list language +

timing specifications +

waveform generation language => VHDL

Therefore, the language has constructs that enable us to express the concurrent or sequential behavior of a digital system with or without timing. It also allows us to model the system as an interconnection or components. Test waveforms can also be generated using the same

constructs. All the above constructs may be combined to provide a comprehensive description of the system in a single model.

2.2 History of development

The requirements for the language were first generated in 1981 under the VHSIC program. In this program, a number of U.S. companies were involved in designing VHSIC chips for the Department of Defense (DoD). At that time, most of the companies were using different hardware description languages to describe and develop their integrated circuits. As a result, different vendors could not effectively exchange designs with one another. Also, different vendors provided DoD with descriptions of their chips in different hardware description languages. Reprocurement and reuse was also a big issue. Thus, a need for a standardized hardware description language for the design, documentation, and verification of digital systems was generated.

A team of three companies, IBM, Texas Instruments, and Intermetrics, were first awarded the contract by the DoD to develop a version of the language in 1983. Version 7.2 of VHDL was developed and released to the public in 1985. There was strong industry participation throughout the VHDL language development process, especially from the companies that were developing VHSIC chips. After the release on version 7.2, there was an increasing need to make the language in industry-wide standard. Consequently, the language was transferred to the IEEE for

standardization in 1986. After a substantial enhancement to the language, made by a team of industry, university, and DoD representatives, the language was standardized by the IEEE in December 1987; this version of the language is known as the IEEE Std 1076-1987. The official language of description appears in the IEEE Standard VHDL Language Reference Manual, available from IEEE. The language has also been recognized as an American Nation Standards Institute (ANSI) standard.

2.3 Advantages of using VHDL for modelling digital hardware

The following are the major capabilities that the language provides along with the features that differentiate it from other hardware description languages.

The language can be used as an exchange medium between chip vendors and CAD tool users. Different chip vendors can provide VHDL descriptions of their components to system designers. CAD tool users can use it to capture the behaviour of the design at a high level of abstraction for functional simulation.

The language can also be used as a communication medium between different CAD and CAE tools. For example, a schematic capture program may be used to generate a VHDL description for the design, which can be used as an input to a simulation program.

The language supports hierarchy; that is, a digital system can be modeled as a set of interconnected components; each components, in turn, can be modeled as a set interconnected subcomponents.

The language supports flexible design methodologies : top-down, bottom-up, mixed.

The language is not technology-specific, but is capable of supporting technology-specific features. It can also support various hardware technologies; for example, we may define new logic types and new components; we may also specify technologies-specific attributes. By being technology-independent, the same model can be synthesized into different vendor libraries.

It supports both synchronous and asynchronous timing models.

Various digital modeling techniques, such as finite-state machine descriptions, descriptions, and Boolean equations, can be modeled using the language.

The language is publicly available, human-readable, machine-readable, and above all, it is not proprietary.

The language supports three basic different description styles: structural, dataflow, and behavioral. A design may also be expressed in any combination of these three descriptive styles.

It supports a wide range of abstraction levels ranging from abstract behavioral descriptions to very precise gate-level descriptions. It does

not, however, support modeling at or below the transistor level. It allows a design to be captured at a mixed level using a single coherent language.

Arbitrarily large designs can be modeled using the language, and there are not limitations imposed by the language on the size of a design.

The language has elements that make large-scale design modeling easier, for example, components, functions, procedures, and package.

Test benches can be written using the same language to test other VHDL models.

Nominal propagation delays, min-max delays, setup and hold timing, timing constraints, and spike detection can all be described very naturally in this language.

The use of generics and attributes in the models facilitate back annotation of static information such as timing or placement information.

Generics and attributes are also useful in describing parameterized designs.

A model can not only describe the functionality of a designs, but can also contain information about the design itself in terms of user-defined attributes, such as total area and speed.

A common language can be used to describe library components from different vendors. Tools that understand VHDL models will have no difficulty in reading models from a variety of vendors since the language is a standard.

Models written in this language can be verified by simulation since precise simulation semantics are defined for each language construct.

Behavioral models that conform to a certain synthesis description style are capable of being synthesized to gate-level descriptions.

The capability of defining new data types provides the power to describe and simulate a new design technique at a very level of abstraction without any concern about the implementation details.

2.4 Modeling technique using VHDL

Entity declaration : The entity declaration specifics the name of the entity being modeled and lists the set of interface ports. Ports are signals through which the entity communicates with the other models in its external environment.

ARCHITECTURE BODY

The internal details of an entity are specified by an architecture body using any of the following modeling styles :

1. **Structural style** : As a set of interconnected components (to represent structure). It represents the lowest level of abstraction i.e. no abstraction at all.

2. Data Flow Style : As a set of concurrent assignments statements (to represent data flow). It represents an abstraction level higher than structural style in which be specify only logical expression.

3. Behavioural style : As a set of sequential assignment statements (to represent behavior). It represents the highest level of abstraction in which we specify only the behaviour of the entity.

AN EXAMPLE

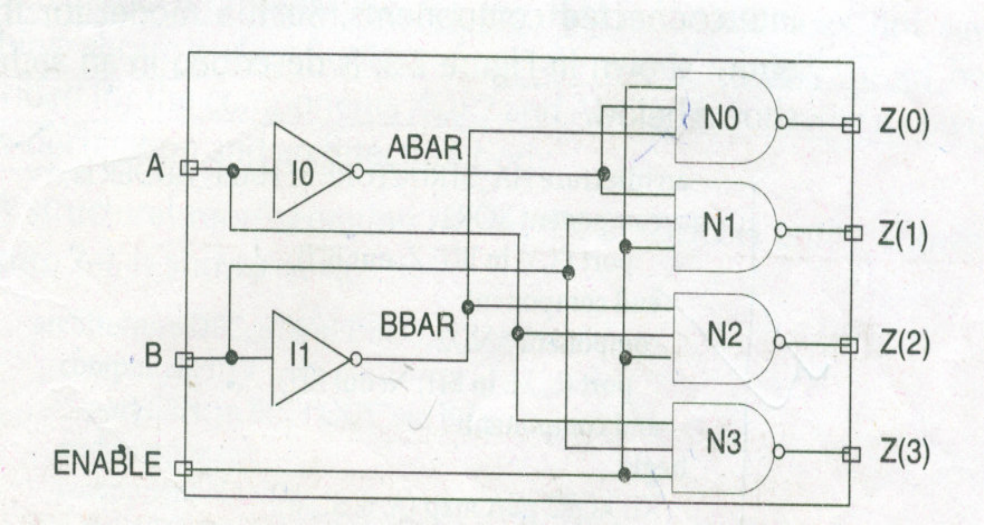


Figure 1: 2 X 4 DECODER CIRCUIT

2.4.1 Structural style :

```
entity DECORDER2×4 is
        port (A, B, ENABLE: in BIT; Z: out BIT_VECTOR(0to 3));
end DECODER2×4;
architecture DEC_STR of DECORDER2×4 is
        component INV
PORT(PIN: in BIT;POUT:out BIT);
```

```
end  component;

component NAND3

port(DO,D1,D2:in BIT;DZ:out BIT);

end component;

signal ABAR,BABAR:BIT;

begin

VO:INV port map(A,ABAR);

V1:INV port map(B,BBAR);

NO:NAND3 port map(ENABLE,ABAR,BBAR,Z(0));

N1: NAND3 port map(ABAR,B,ENABLE,Z(1));

N2: NAND port map(A,BBAR,ENABLE,Z(2));

N3: NAND port map(A,B,ENABLE,Z(3);

End DEC_STR;
```

2.4.2 Data Flow style :

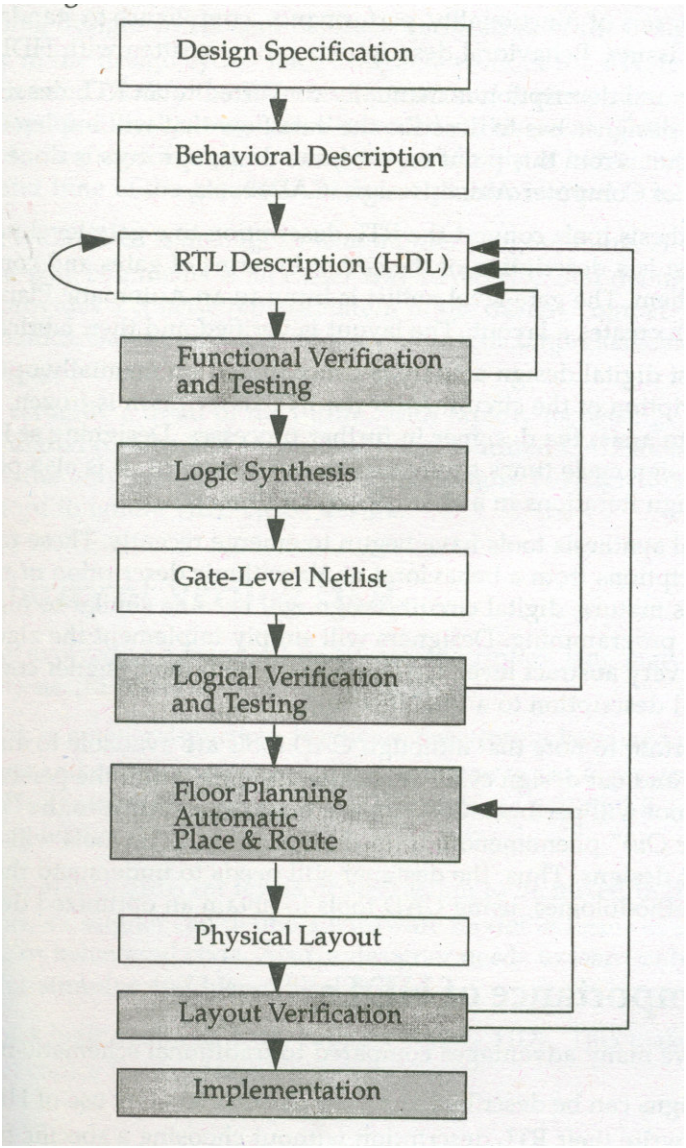
```
        entity DECORDER2×4 is

        port (A, B, ENABLE: in BIT; Z: out BIT_VECTOR(0to 3));

end DECODER2×4;

architecture DEC_DATAFLOW of DECORDER2×4 is

signal ABAR,BBAR:BIT;

BEGIN

Z(3)<=not(A and B and ENABLE);

Z(0)<=not(ABAR and BBAR and ENABLE);

BBAR<= not B;

Z(2)<=not (A and BBAR and ENABLE);

ABAR<=not A;

Z(1)<=not(ABAR and B and ENABLE);

End DEC_DATAFLOW;
```

2.4.3 Behavioural style :

```
        entity DECORDER2×4 is
        port (A, B, ENABLE: in BIT; Z: out BIT_VECTOR(0to 3));
end DECODER2×4;


architecture DEC_SEQUENTIAL of DECODERS2×4 is
begin
        process (A, B, ENABLE)
                variable ABAR, BBAR: BIT;
begin
        ABAR : = not A;
        BBAR := not B;
        If ENABLE = '1' then
                Z(3) <= not (A and B);
                Z(0) <= not (BBAR and BBAR);
                Z(2) <= not (A and BBAR);
                Z(1) <= not (ABAR and B);
Else
        Z <= "1111";
```

```
end if;

end process;

end DEC_SEQUENTIAL;
```

3 DESIGN FLOW



3.1 Analysis :

The analysis phase consists of writing a specification. The specification can be written in VHDL or ordinary language. The purpose of the specification is to find out WHAT HAS TO BE DONE. The specification can be described in VHDL and then verified in a VHDL simulator.

3.2 Design :

The design phase means transforming the specification into an architecture and VHDL code. The phase starts with defining an architecture (block diagram). When the architecture is ready, the VHDL code is written for the various components (blocks) or ready-made components copied from a library. Then the function of the design is verified in a simulator. When the result agrees with the specification, the designer can go on to the next phase. In this phase the major challenge is HOW SHOULD THE ARCHITECTURE AND COMPONENTS BE DESIGNED?

3.3 Technology Mapping :

It is parameters such as price, performance and supply, etc., that determine which technology will be selected. This phase is now largely automated. The time constraints are described in a format which can be read by the synthesis tool. If the synthesis tool cannot meet the time constraints, the design phase must be repeated in full or in part. An

approved synthesis produces a technology-dependent netlist (schematic), which is an input file for other tools.

3.4 Prototyping :

A prototype is then built and compared with the specification Programming of an FPGA (Field Programmable Gate Array) for the design is called prototyping. If the RESULT is the same as the SPECIFICATION, the circuit is ready. This comparison is called validation.

4 COMPLEX PROGRAMMABLE LOGIC DEVICE

4.1 Advantages of using CPLD

•It is the best prototyping solution because CPLD comes with in built hard ware to implement

•Cost effective solutions.

•Involves less risk.

•Design security.

•Consumes less board area.

•Reconfigurable computing.

•Best suits Flexibility

•In system programmability.

•Less project development time.

•Best hardware verification for design.

4.2 CoolRunner XPLA3 Family

XPLA3 is from the CoolRunner CPLD family. The XPLA3 family includes devices ranging from 32 to 384 macrocells. XPLA3 was created to maintain the same competitive advantages as the existing CoolRunner

families, add additional features, increase performance, and offer a substantially lower cost. The CoolRunner XPLA3 is a non-volatile (Flash based), 0.35µ CMOS CPLD which offers ultra low power consumption, a flexible architecture, and high-speed capabilities.

4.3 CoolRunner XPLA3 Architecture

Designers want CPLD devices that offer high speed, high density, and the flexibility to make changes to their design at any stage of the design process. A particular device's ability to meet all of these critical needs efficiently is often constrained by the basic architecture of the CPLD.The basic components of CPLD architecture that affect the device's speed, density, and design flexibility can be broken into four distinct areas. These four areas are the basic interconnect methodology, logic block architecture, logic allocation method, and the timing model of the device. The CoolRunner XPLA3 architecture is the result of extensive research into the effect architecture has on these critical system needs and delivers a third generation solution that is superior to previous architectures. From a high-level, the architecture of CoolRunner XPLA3 CPLDs appears similar to many other CPLD architectures. As shown in Figure 2, the XPLA3 architecture consists of logic blocks containing macrocells interconnected by a routing matrix. Each XPLA3 logic block contains 16 macrocells. The routing matrix is called the ZIA (Zero-power Interconnect Array) and provides 36 true and complement signals to

each logic block. A 4-bit Universal Bus is used to provide an individual asynchronous clock (UCLK), reset (URST), preset (UPST), and output enable (UOE). These bus lines are driven by four multiplexers (muxes), with the mux inputs consisting of a single control p-term from each Logic Block. The 32 macrocell version of XPLA3 will have two logic blocks, which means a maximum of two universal control functions can be implemented. All other family members can have up to four universal functions. Four external clock signals (Global Clocks) are muxed down to two, selectable at each logic block.

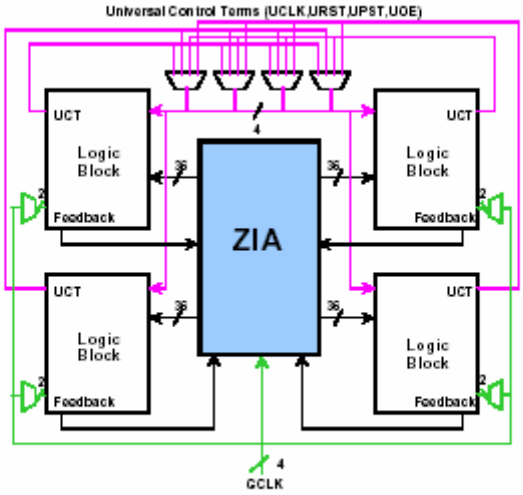


Figure 2: XPLA3 High-level Architecture (64 Macrocell device shown)

4.3.1 Zero Power Interconnect Array

The basic premise of CPLD architecture is the construction of large devices that are built upon multiple PLD blocks that are connected via an interconnect matrix (see Figure 3). In CPLDs, this interconnect resource is supposed to act like a crosspoint switch to route signals from

the Inputs, I/Os, and macrocell feedbacks to the logic blocks where these signals are needed. The interconnect must also be very fast to support the high speeds that designers expect in today's CPLDs. The ability to lock pins is problematic when interconnect fails in its ability to route signals under worst case conditions. The ideal performance of an interconnect is to fully emulate a crosspoint switch, where every input to the array can be connected to every output of the array under fixed pinouts. Some first generation devices used full crosspoint switch arrays, and as a result offered 100% routability, at a significant price. As shown in Figure 4, building a crosspoint switch with these devices required a fuse at every intersection of the input and output line in the array. For a 128 macrocell device, this would translate into more than 65,000 connections. More significantly, these fully populated crosspoint switches were relatively slow, accounting for an 8 ns to 15 ns delay.

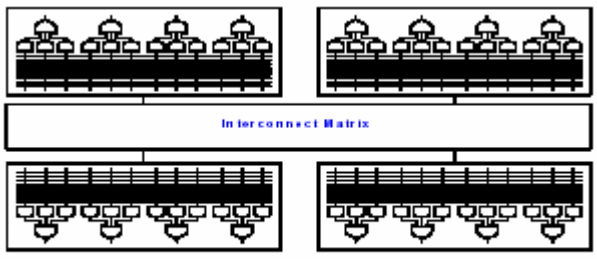


Figure 3: Basic CPLD Architecture

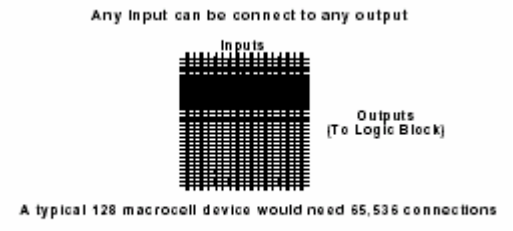


Figure 4: Cross Point Switch

The next step in interconnect evolution was the use of multiplexers to emulate crosspoint switches, a technique that all contemporary devices employ. Figure 5 shows a set of 16 muxes that are two bits wide which form an interconnect that has 32 inputs and 16 outputs. The use of muxes has two immediate effects. The first is that the delay through the interconnect is typically equivalent to a single mux delay, which is typically well under 0.5 ns. The second effect is the reduction of connections required to implement the interconnect. As a result, the number of connections required for a 128 macrocell device can be reduced from approximately 65,000 to less than 2,000.

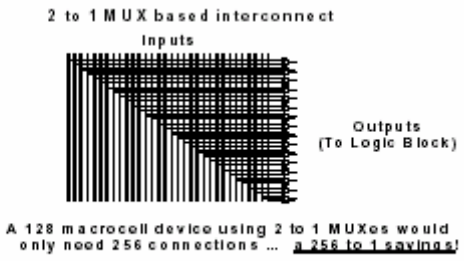


Figure 5: 2:1 Mux Based Interconnect

Unfortunately, if the architecture of this interconnect is not well engineered, signal blocking can occur. The issue with this 2:1 Mux approach is that only half of the inputs can enter the logic block. The key to building an efficient non-blocking muxed based interconnect is to include many overlapping, wide muxes which give each input multiple chances to get into the logic block. The main trade-off is routability (non-blocking) and costs (silicon area and performance).Enough resources

(wide Muxes) need to be included to ensure not only that the device will be able to route the design but must also be architected to facilitate last minutes design changes once the pins have been locked.

The XPLA3 interconnect (see Figure 6) was architected by the software group that was responsible for ensuring that it would be capable of re-routing fixed pinout designs. By extensively simulating the width of the muxes and the number deployed, a mux based interconnect can be designed such that the probability of signal blocking is statistically very low.The XPLA3 interconnect employs a sufficiently large number of input muxes, of sufficient width, to guarantee routability under worst case conditions. The final interconnect architecture was subjected to over 16 million iterations of worst case fixed pinout routing. This resulted in worst case signal routing of 99.997% when every I/O, input pin, and macrocell is in use and has a fixed pinout. If only 35 of the 36 logic block inputs are used, 100% of the 16 million fixed signal routings completed successfully. It is believed that this type of solution allows designers total freedom to make design iterations without the fear of having to re-layout the PCB. Not every CPLD architecture is able to support interconnect routing that is this robust.

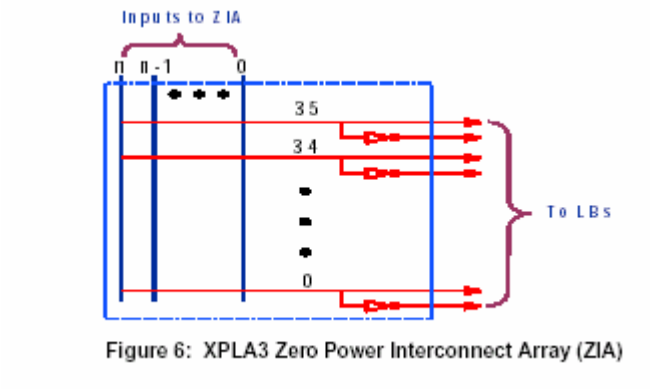


Figure 6: XPLA3 Zero Power Interconnect Array (ZIA)

.

4.3.2 Logic Block Fan-in

The fan-in to each logic block in a CoolRunner XPLA3 CPLD from the ZIA is advertised as 36.However, the architecture of the XPLA3 CPLD actually provides 40 routing channels to each logic block. The software defaults to using a logic block fan-in of 36 and can utilize any 36 of the 40 fan-in to the logic block, i.e., the 36 routing channels utilized by the software are not dedicated. These four extra fan-in signals are reserved and can be enabled in software when necessary

4.3.3 Logic Block

Figure 7 illustrates the logic block architecture. There are 36 pairs of true and complement inputs from the ZIA that feed 48 product terms (PTs) in the array. Each logic block contains a PLA (Programmable Logic Array) which provides a pool of 48 PTs that can be used as macrocell clocks, control terms (reset, preset, clock-enables, or output-enables), or as needed by the 16 macrocells in the logic block. These 48 product

terms can be used by one or all 16 of the macrocells in the logic block. None of the product terms are dedicated, therefore, product terms that are not used for control terms or macrocell clocks may be used for macrocell logic.Within the 48 product terms there are:

• Eight product terms, PT[0-7], can be used to generate eight local control terms (LCT[0:7]) that are available for use, by each macrocell, as asynchronous clocks, resets, presets, and output enables. One of the control product terms in each Logic Block is made available to the Universal Control Term Bus. It can drive any one of the four Universal Control Terms through the Universal Control Term Mux.

• Sixteen product terms, PT[16:31], are coupled with the associated programmable OR gate into the VFM (Variable Function Multiplexer). The VFM increases logic optimization by implementing any two input logic function before entering the macrocell.

• Sixteen product terms, PT[32:47], can be used as asynchronous clocks or as clock enables.

• Eight fold-back NAND product terms, PT[8:15], are available for ease of fitting and pin locking. These fold-back NAND structures increase the virtual width of the product term and allow more logic to be placed in less silicon area.

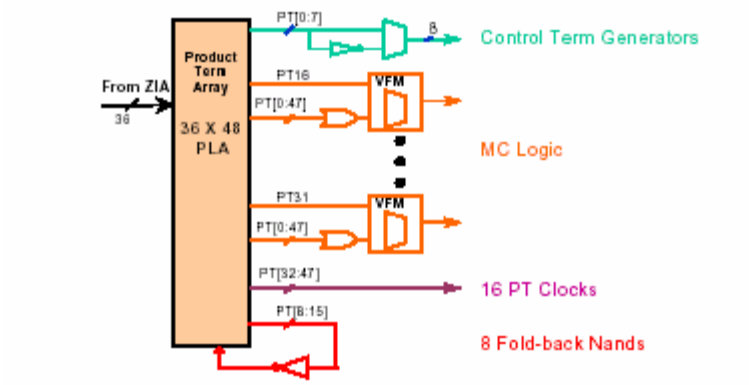


Figure 7: XPLA3 Logic Block Architecture

A "Power Up" input to the ZIA, active during the initialization of the PLD, allows individual macrocells to be reset or preset depending on the customer configuration pattern.

4.3.4 Variable Function Mux (VFM)

A Variable Function Mux is a small part of the PLA logic that feeds directly into each macrocell. Every macrocell has its own VFM. As shown in Figure 8, this VFM can be thought of as a flexible programmable logic element that can synthesize any two input logic element, such as XOR, XNOR, etc. Each VFM has two inputs, a single product term input and a sum of products input. Both of these inputs come directly from the PLA; the sum of products input may consist of up to 48 product terms.

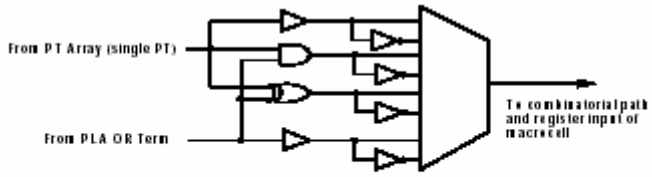


Figure 8: Variable Function Mux

4.3.5 Fold-back NANDs

Fold-back NANDs are powerful architectural additions for synthesis capable tools and provide increased density. Figure 9 illustrates how a Fold-back NAND is constructed. The key operation of a Fold-back NAND is to allow the software synthesis to use De Morgan's Theorem to create"virtual" PTs. In other words, equations can be re-written using De Morgan's Theorem to reduce the number of PTs needed to implement the equivalent functionality.

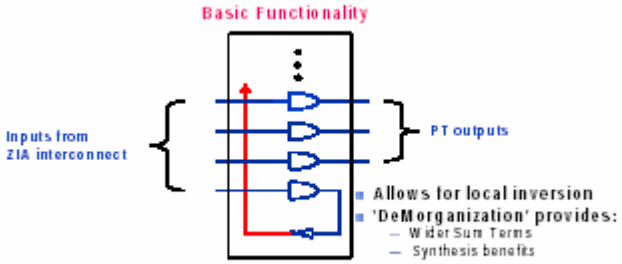


Figure 9: Fold-back NAND

Figure 10 shows an example of how Fold-back NANDs can be used to reduce the number of required product terms. Fold-back NANDs are especially effective in state machines and decoder designs. XPLA3 devices use Fold-back NANDs to cost effectively implement higher density logic

(Fold-back NANDs require much less silicon area than additional product terms).

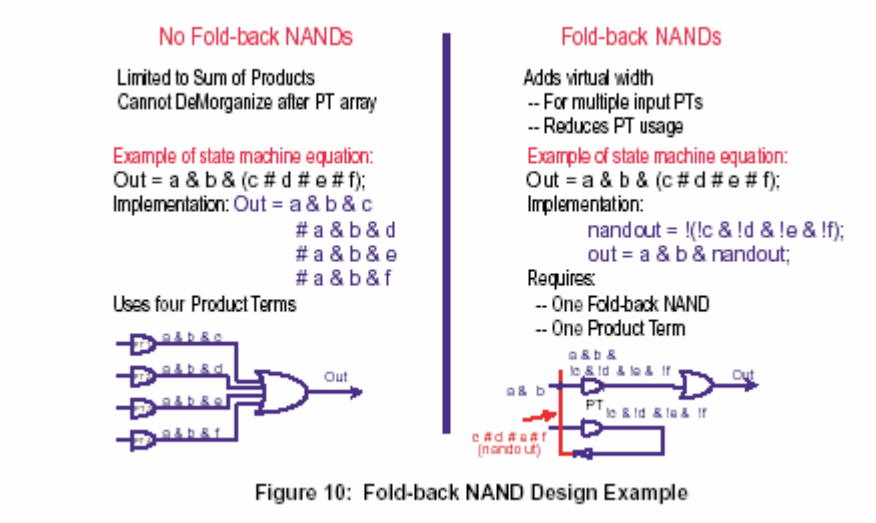


Figure 10: Fold-back NAND Design Example

4.3.6 PAL versus PLA

There are two basic types of product term generators: a PAL (programmable AND, fixed OR) and a PLA (programmable AND, programmable OR). Both types of logic use an AND array to build up a product term; the difference between the two is how the product terms are summed.The PAL is comprised of a programmable AND array followed by a fixed number input OR element. Each OR element has a certain number of dedicated PTs and sharing of these PTs is not allowed. The PLA array differs from the PAL in that the AND array is followed by a programmable width OR array (see Figure 11). Having a programmable OR Array allows PTs to be shared between macrocells (effectively increasing device density), to have excellent pin locking (every PT is

available to every macrocell), and provides a very simple timing model. The XPLA3 architecture is based on a PLA; all non-CoolRunner CPLDs employ a PAL.

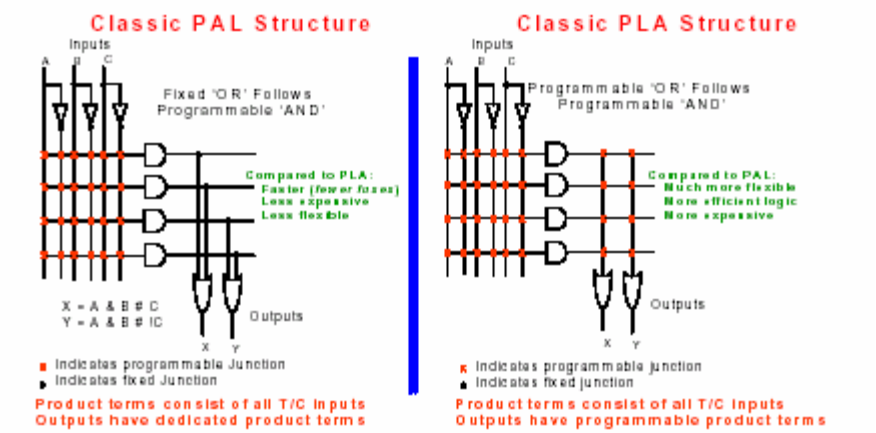


Figure 11: Basic PAL / PLA Structures

4.3.7 Product Term Sharing

Many logic designs, such as decoders and state machines, have common logic components. If an architecture allows for sharing of resources (e.g., PLA), this common logic can be built up one time and provided to all higher expressions that require this common sub-expression. This sharing of logic means that common logic does not have to be duplicated. Figure 12 is intended to help illustrate the PAL / PLA product term sharing differences.

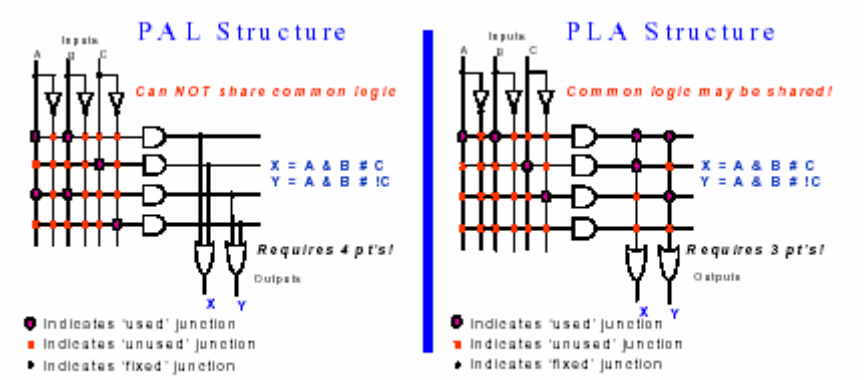


Figure 12: Product Term Sharing Design Example

4.3.8 Product Term Allocation Method

There are two different approaches used by CPLD manufactures to allocate logic: product terms steering and PLA implementation. Product-term steering dedicates a certain number of product terms to each macrocell in the logic block. Steering mechanisms are used that allow a macrocel's product terms to be steered to adjacent macrocells when needed. The PLA array is a programmable AND, programmable OR structure, therefore all product terms in the array are available to all of the macrocells in the logic block; there are no dedicated product terms. In addition, all product terms that are common to multiple macrocells in the logic block can be implemented once and shared by all macrocells in the logic block.

Figure 13 illustrates a product-term steering type of logic-block architecture. Note that this logic-block architecture is not used in XPLA3 devices. In this figure, each macrocell has four dedicated product terms. When a macrocell needs additional product terms, the product terms

from an adjacent macrocell are steered to this macrocell. This macrocell whose product terms were re-directed may now be stranded and un-usable.

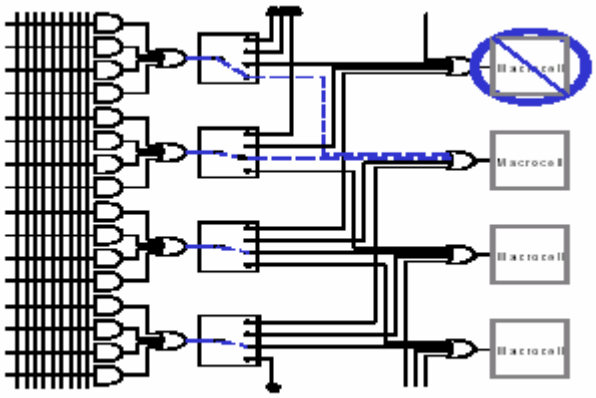


Figure 13: Logic Block Architecture with Product-term Steering

Consider the case where a design has been completed and the board-level debugging has begun. It is determined that a design change is necessary in the CPLD logic and this design change requires that a certain macrocell now requires seven product terms. If the CPLD logic block architecture is similar to that shown in Figure 13 and the surrounding macrocells are currently utilized as outputs, this design change can not be implemented. Taking this example further, if a design change is such that the pinout can be maintained, there is a high probability that the timing of the design will change as additional product terms are steered to implement the new logic.

If, however, this design is targeted to a CoolRunner XPLA3 CPLD, unused product terms in the PLA can be used to implement the pinouts

design change without affecting any of the macrocells in the logic block or the device pinout as shown in Figure 14. All product terms in the PLA are available to each macrocell in the logic block, therefore, the pinout can be maintained. An additional advantage of the PLA structure is that product terms that are common to multiple macrocells in the logic block are implemented once and shared.

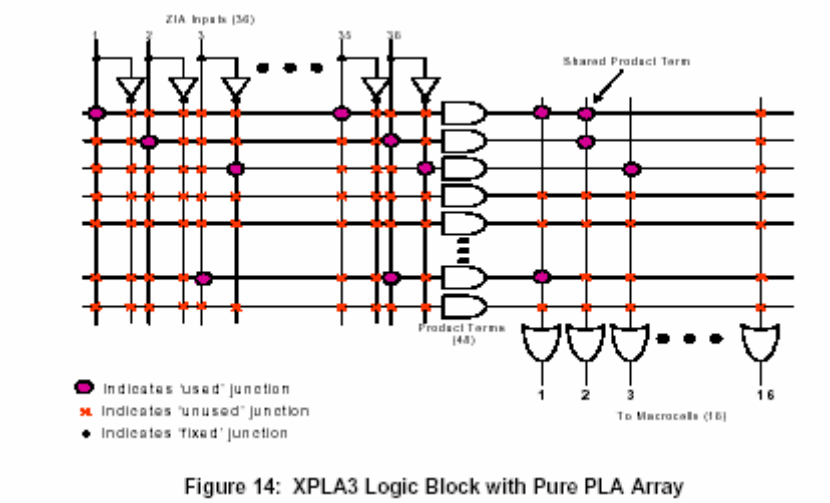


Figure 14: XPLA3 Logic Block with Pure PLA Array

CPLD manufacturers who implement product-term steering will advise designers to move to the next larger macrocell count device if their current device utilization is > 80% so that pinout can be maintained if design changes are necessary. Note that this recommendation means that a design using 102 macrocells of a 128 macrocell device should be implemented in the next larger CPLD (typically a 256-macrocell device) if there is a possibility of design changes! Because of the high probability of implementing design changes due to debug, feature additions, or system test, this recommendation can significantly increase the cost of a

systemand is a waste of silicon that has already been purchased. CPLDs that utilize product-term steering do not allow designers to maximize the benefits of In-System Programmability (ISP).Note that the key to maintaining a fixed pinout is not only the efficiency of the routing matrix but how the product terms are allocated within the logic block. With the PLA structure implemented in a CoolRunner XPLA3 CPLD, fixed pinouts are maintained after logic changes even at device utilizations of > 99%, making the XPLA3 architecture optimal for ISP.

4.3.9 Macrocell

Each macrocell, as shown in Figure 15, can support combinatorial or registered inputs, a universal preset and reset for each macrocell and configurable D, T, or L registers with maximum clocking flexibility. There are two feedback paths to the ZIA: one from the macrocell and one from the I/O pin. When the I/O is used as an output, the output buffer is enabled, and the macrocell feedback path can be used to feed back the logic implemented in the macrocell. When the I/O pin is used as an input, the output buffer will be tri-stated and the input signal will be fed into the ZIA via the I/O feedback path. The logic implemented in the buried macrocell can be fed back to the ZIA via the macrocell feedback path. macrocells which are buried within a Logic Block and not connected to an I/O are identical to the non-buried macrocells. Each macrocell can be used to implement either register or combinatorial functions.

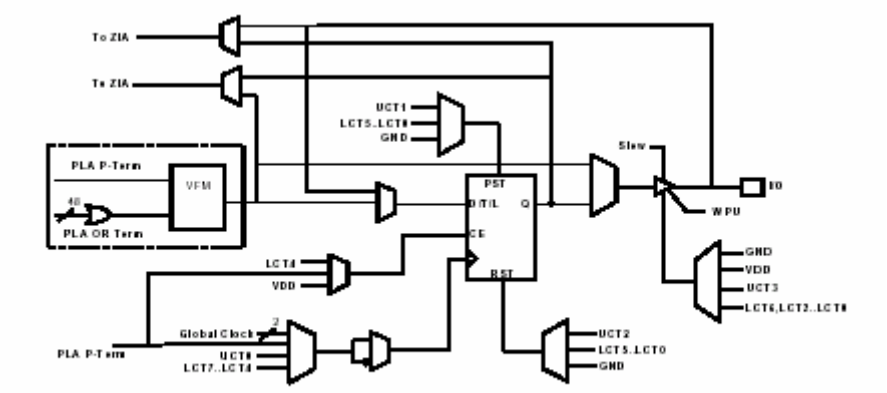


Figure 15: XPLA3 Macrocell

Register Functionality

The data input to each macrocell register is derived from the output of Variable Function Multiplexer. Each macrocell register can be configured as a D-, T-, or Latch-type flip-flop; this flip-flop may also be configured to be an input register (see the section entitled, "Input Register Configuration" for more details). Each flip-flop has both asynchronous preset and reset capabilities. There are seven different preset and reset sources: one universal control term (one for preset [UCT1] and one for reset [UCT2]) and six shared local control terms (LCT[0-5]).

Clocking

Each macrocell register can be clocked from any one of ten sources

• There are two global clocks that are derived from the four external clock pins via a 4:2 multiplexer.

• There is one universal clock signal (UCT0) sourced by a universal control term.

• There are four Local Control Terms (LCT4-LCT7) which can be used as clock signal and can be individually configured as either a product term or sum term equation created from the 36 signals available inside the logic block.

• There is one dedicated product term clock per macrocell. In addition to having ten possible clock sources, polarity (rising or falling edge) is also selectable at each macrocell. Hardware clock enables are also available for added clock control.

Input Register Configuration

The XPLA3 device macrocells may have their registers configured as input registers; this means that signals from a pin may be directly latched by the register without having to pass through the interconnect array. The setup time for this is 2 ns and is accompanied by a 0 ns hold time. As shown in Figure 16, when implementing an input register the preceding macrocell logic is still available for use as a buried combinatorial node. This logic may be fed back to the interconnect array for distribution elsewhere in the device.

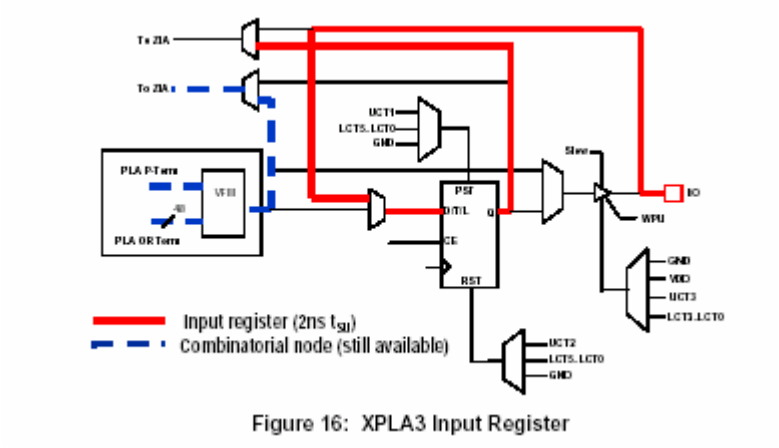


Figure 16: XPLA3 Input Register

4.3.10 I/O Cell

The XPLA3 devices are implemented on a 0.35□□process and run off of a single supply VDD of 3.3V. All I/Os are 5V tolerant and provide timing, voltage, and current characteristics required by the PCI specification. XPLA3 devices are PCI compatible but not compliant. CoolRunner CPLDs are not compliant because the specification requires overshoot signal conditioning diodes that the XPLA3 devices do not provide. If a designer wishes to implement a PCI driver/ receiver, the designer will be required to provide external diode clamps. With the exception of the clamp diode, XPLA3 devices meet the stringent requirements of the PCI specification. The Output Enable (OE) has eight possible states as shown in Table 1.

Table 1: XPLA3 Output Enable Selection

OE Decode	I/O Pin State
0	3-state
1	Function CT0
2	Function CT1
3	Function CT2
4	Function CT6
5	Universal OE (UCT3)
6	Enable
7	3-state with Weak Pull-up

The XPLA3 output buffers incorporate weak pull up resistors (option # 7) to provide internal termination when I/Os are unused. These pull-ups are not available for used I/Os. Please note that dedicated inputs (global clocks) do not have a pull-up resistor and therefore should be properly terminated (pulled Hgh or Low) if left unconnected. XPLA3 devices also provide slew rate control for each macrocell output pin. The user has the option to enable the slew rate control to reduce EMI. The nominal delay for using this is 3 ns.

4.3.11 Timing Model

The final consideration when selecting a CPLD is the timing model. CPLDs should offer fast, deterministic timing that remains invariant as design changes are made. More specifically, since late changes often involve adding additional logic, the ability of the device to maintain predictable timing as the logic width increases is important. Unfortunately, many devices have speeds that are attainable only under

a limited set of conditions. As additional logic complexity is introduced, the timing may suffer and be significantly different from the 'peak' speed promised. It is a good idea to review the timing information and/or model for the device being considered. Sometimes it can be very difficult to determine what the timing will be if 16 (or more) product terms are used.In non-CoolRunner architectures, the user may be able to fit the design, but may not be sure whether system timing requirements can be met until after the design has been fit into the device. This is because timing models of these architectures are very complex (dependent on many variables such as: the number of parallel expanders borrowed, sharable expanders, different routing channels, etc). Figure 17 shows the XPLA3 timing model which has three main parameters, including T_{PD}, T_{SU}, and T_{CO}. As a result of the simplicity of this timing model, designers can make reasonably accurate estimations of the performance of their design before they began using the device. It is important to note that the PLA timing is deterministic regardless of the number of PLA terms that are used or the number that are shared by multiple outputs.

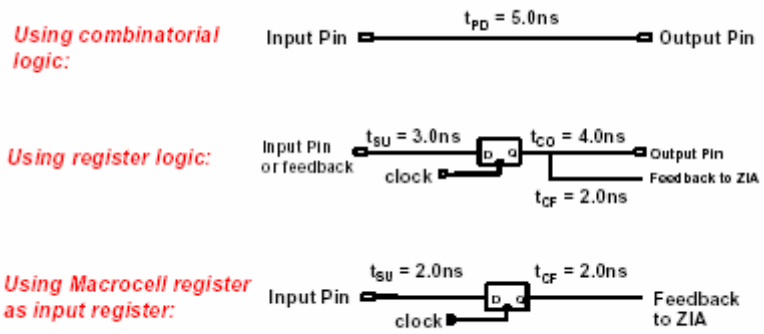


Figure 17: XPLA3 Timing Model

5 Specification

The functioning of the 8255 in each of the three modes is described.

5.1 Mode 0: Basic Input/Output

In this mode, in addition to Port A and Port B, PC_0-PC_3 and PC_4-PC_7 of Port C can be considered as two individual 4-bit ports. Therefore, four ports, each of which can be configured either as an input port or an output are available. It follows that there are sixteen possible input/output configurations. Here the ports are simple input or output ports: data is written to or read from the specified port without handshaking. The data sent out to the output ports are latached, where as inputs are not latched.

5.2 Mode 1 : Strobed Input / Output

In this mode, input or output data transfer is effected by strobes, otherwise called handshaking signals. The two groups, Group A and Group B, can be configured separately, with each group consisting of an 8-bit port and a 4-bit port. The 8-bit port can be programmed for imput or output (unidirectional) operation with latched output and latched input facilities. The 4-bit port is used for handshaking. In the output

mode, data is continuously present on the output pins once the port is written into by the CPU.

5.3 Mode 2 : Strobed Bidirectional Bus I/O

This mode allows bidirectional data transfer (transmission and reception) over a single 8-bit data bus using handshaking signals. This feature is available only in Group A with Port A as the 8-bit bidirectional data bus; and PC_3-PC_7 are used for handshaking purpose. In this mode too, both inputs and outputs are latched. Due to use of a single 8-bit data bus for bidirectional data transfer, the data sent out by the CPU through Port A appears on the bus connecting it to the peripheral, only when the peripheral requests it.

5.4 BIT SET-RESET FEATURE (BSR)

In addition to the above modes, individual bits of Port C can be set or reset by sending out a single OUT instruction to the control register. When Port C is used for control / status operation, this feature can be used to set or reset individual bits as if they were output ports.

6 FUNCTIONAL ORGANIZATION

The 8255 Programmable Peripheral Interface is used as a general purpose device to interface peripheral devices to the microcomputer system bus. The Figure 18 on the next page shows the internal block diagram of the 8255 Programmable peripheral interface. The functions of each block are described next, with respect to block diagram.

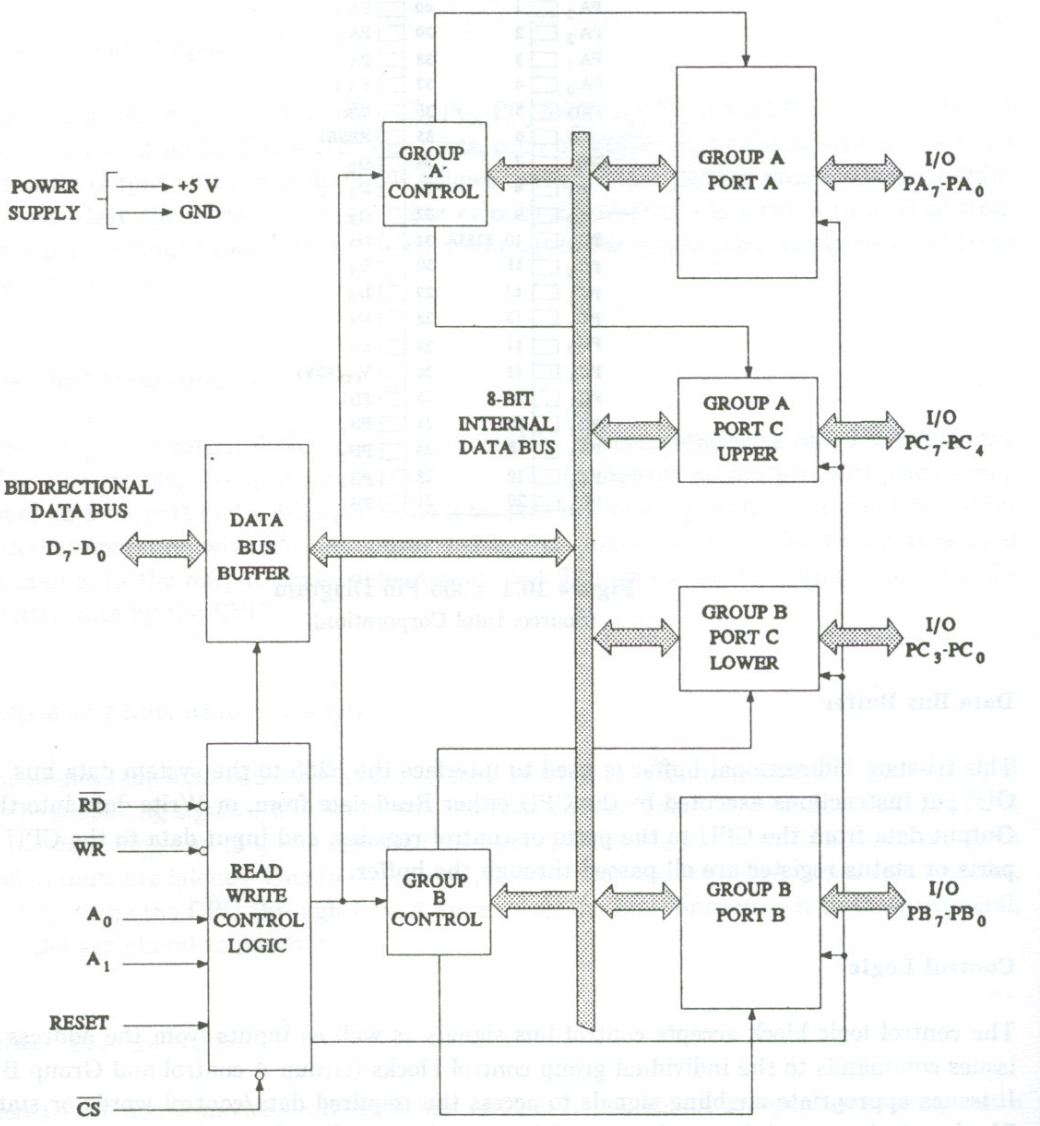


Figure 18: BLOCK DIAGRAM OF 8255 PPI

6.1 DATA BUS BUFFER

This tri-state bidirectional buffer is used to interface the 8255 to the system data bus. Input or Output instructions executed by the CPU either Read data from, or Write data into the buffer. Output data from the CPU to the ports or control register, and input data to the CPU from the ports or status register are all passed through the buffer.

6.2 CONTROL LOGIC

The control logic block accepts control bus signals as well as inputs from the address bus, and issues commands to the individual group control block (Group A control and Group B control). It issues appropriate enabling signals to access the required data / control words or status word. The input pins pertaining to the control logic section are described below.

6.2.1 CHIP SELECT

This is an active low input which must be enabled for data transfer operation between the CPU and the 8255.

A_1	A_0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	
					INPUT (READ) OPERATION
0	0	0	1	0	Port A to Data Bus
0	1	0	1	0	Port B to Data Bus
1	0	0	1	0	Port C to Data Bus
					OUTPUT (WRITE) OPERATION
0	0	1	0	0	Data Bus to Port A
0	1	1	0	0	Data Bus to Port B
1	0	1	0	0	Data Bus to Port C
1	1	1	0	0	Data Bus to Control Register
					DISABLE FUNCTION
X	X	X	X	1	Data Bus Tri-stated
1	1	0	1	0	Illegal Condition
X	X	1	1	0	Data Bus Tri-stated

Table 2: PORT AND REGISTER SELECT SIGNAL SUMMARY

6.2.2 A_0 and A_1

These input signals along with RD and WR inputs control the selection of the control / status word registers or one of the three ports. The table 2 above summarizes the status of A_0, A_1, $\overline{CS}$, $\overline{RD}$, and $\overline{WR}$ to access the control word / ports. A_0 and A_1 are generally connected to the A_0, A_1 bits of the address bus; the 8255 therefore occupies four consecutive locations in the I/O space.

6.2.3 $\overline{RD}$ (Read)

When this input pin is made low, the CPU can read the data in the ports or the status word, through the data buffer.

6.2.4 $\overline{WR}$ (Write)

When this input pin is made low, the CPU can write data onto the ports or onto the control register through the data bus buffer.

6.2.5 RESET

When this input is made high, the control register is cleared and all the ports are set to the input mode.

6.3 Ports A , B ,C

Port a has an 8-bit latched and buffered output and an 8-bit input latch. Port B has an 8-bit data I/O latch / buffer and an –bit data input buffer. Port C has one 8-bit unlatched input buffer and an 8-bit output latch / buffer. Port C and be split into two parts and each can be used for control signal outputs/ inputs for Port A and B in the handshake mode. Each of the Group A and Group B control block receives control words from the CPU through the data buffer and internal data bus, accepts commands from the control logic block, and issues appropriate commands to the ports associated with it. The Group A control block

controls Port A and PC_7-PC_4, while the Group B control block controls Port B and PC_0-PC_3.

7 PROGRAMMING AND OPERATION

7.1 Operation

A high on the RESET pin causes all 24 lines of the three 8-bit ports to be in the input mode. All flip-flops are cleared and the interrupts are reset. This condition is maintained even after the RESET goes low. The port of the 8255 can then be programmed for any other mode by sending out a single output instruction to the control register. Also, the current mode of operation can be changed by writing a single mode word onto the control register, when required.

Modes for Group A and Group B can be separately defined with Port C taking on responsibilities as dictated by the mode definitions of Ports A and B. If Group A is programmed for Mode 0 and Group B programmed for Mode 1, Port A and PC_4-PC_7 can be programmed for either input or output, while Port B can be programmed for input or output with PC_0-PC_2 used for handshaking.

The mode definition format and bit set-reset format are as shown in the figure19A-B on the next page. The control words for both mode definition and Bit Set-Reset are loaded into the same control register, with bit D_7 used for specifying whether the word loaded into the control register is a mode definition word or Bit Set-Reset word. If D_7 is high, the word is

taken as a mode definition world, and if it is low, it s taken as a Bit Set-Reset word.

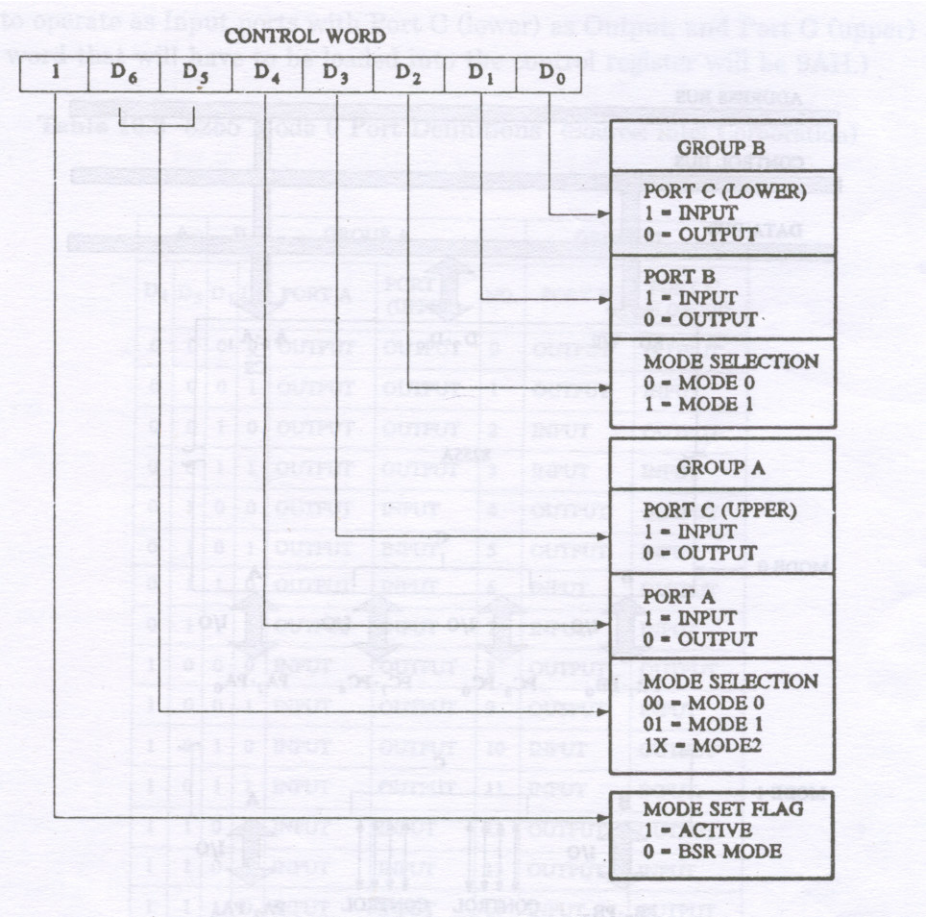


Figure 19A: 8255 PPI MODE DEFINITION FORMAT

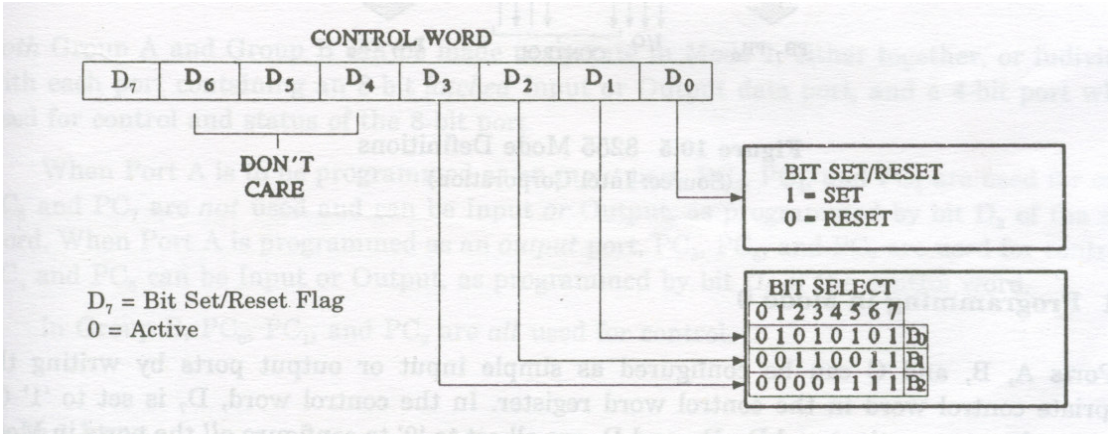


Figure 19 B: 8255 PPI BIT SET/ RESET FORMAT

The appropriate bits are set or reset depending on the type of operation desired, and loaded into the control register (which is accessed when A_1, A_0 both are '1', $\overline{WR}$ and $\overline{CS}$ both are '0'). It is to be noted that Group B does not have provision for operation in Mode 2.

The eight possible combinations of the states of bits D_1-D_3 (B_2 B_1 B_0) in the Bit Set-Reset format (henceforth referred to as BSR) determine the particulate bit in PC_0-PC_7 being set or reset as per the status of bit D_0. A BSR word is to be written for each bit that is to the set or reset. For example, if bit PC_2 is to be set and bit PC_7 is to be reset, the appropriate BSR words that will have to be loaded into the control register will be, 0XXX101 and 0XXX1110, respectively, where X can be either '0' or '1'.

The BSR word can also be used for enabling or disabling interrupt signals generated by Port C when the 8255 is programmed for Mode 1 or

Mode 2 operations. This is done by setting or resetting the associated bits of the interrupts.

7.2 Programming in Mode 0

The Ports A, B, and C can be configured as simple input or output ports by writing the appropriate control word in the control word register. In the control word D_7 is set to '1' (to define a mode set operation) and D_6, D_5, and D_2 are all set to '0' to configure all the ports in Mode 0 operation. The status of bits D_4, D_3, D_1 and D_0 then determine whether the corresponding ports are to be configured as Input or Output.

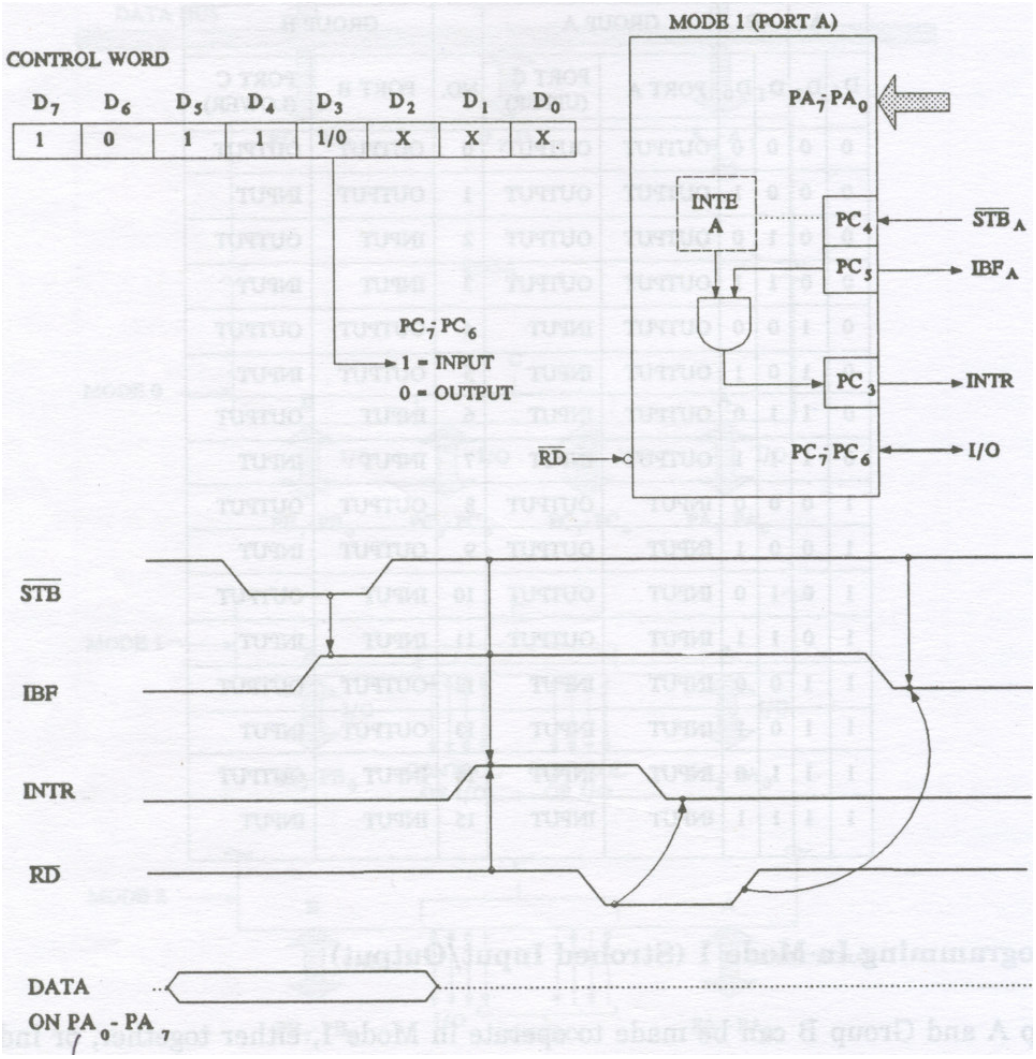
7.3 Programming In Mode 1 (Strobed Input / Output)

Both Group A and Group B can be made to operate in Mode 1, either together, or individually, with each port containing an 8-bit latched Input to Output data port, and a 4-bit which is used for control and status of the 8-bit port.

When Port A is to be programmed as an input port, PC_3, PC_4, and PC_5 are used for control. PC_6 and PC_7 are not used and can be Input or Output, as programmed by bit D_3 of the control word. When Port A is programmed as an output port, PC_3, PC_6 and PC_7 are used for control and PC_4 and PC_5 can be Input or Output, as programmed by bit D_3 of the control word. In Group B, PC_0, PC_1, and PC_2 are all used for control.

Mode 1 Input

The figure 20 shows Port A as in input port (when it operates in Mode 1) along with the control word and control signals (for handshaking with a peripheral). When the control word is loaded into the control register, Group A is configured in Mode 1 with Port A as an input port. Port A can accept parallel data from a peripheral (like a keyboard) and this data can be read by the CPU. The peripheral first loads data into



Figre 20A: 8255 PORT A IN MODE 1 (INPUT)

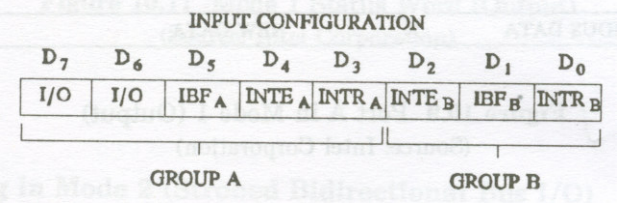


Figure20 B: MODE 1 STATUS WORD (INPUT)

Port A by making the $\overline{STB_A}$ input low. This latches the data placed by the peripheral on the common data bus into Port A. Port A acknowledges reception of data by making IBF_A (Input Buffer Full) high. IBF_A is set when the $\overline{STB_A}$ input is made low.

$INTR_A$ is an active high output signal which can be used to interrupt the CPU so that the CPU can suspend its current operation and read the data written into Port A by the peripheral $INTR_A$ can be enabled or disabled by the $INTE_A$ flip-flop which is controlled by Bit Set-Reset operation of PC_4. $INTR_A$ is set (if enabled by setting the $INTE_A$ flip-flop) after the $\overline{STB_A}$ has gone high again, and if IBF_A is high.

On receipt of the interrupt, the CPU can be made to read Port A. the falling edge of the $\overline{RD}$ input resets IBF_A and it goes low. This can be used to indicate to the peripheral that the input buffer is empty and that can again be loaded into it. The timing diagram and operation of Port B is

similar to that of Port A except that it uses different bits of Port C for control. $INTE_B$ is controlled by Bit Set/Reset of PC_2.

If the CPU is busy with other system operations, it can read data from the input port when it is interrupted. This is often called Interrupt Controlled I/O. However, if the CPU is otherwise not busy with other jobs, it can continuously poll (read) the status word to check for an IBF_A. This is often called Program Controlled I/O. The status word(figure20 B) is accessed by reading Port C (A_1 A_0 must be 10, $\overline{RD}$ and $\overline{CS}$ must be low). The status word format as assumed by the bits of Port C when Ports A and B are input ports in Mode 1, is shown in the figure 20B.

Mode 1 Output

Figure21A shows Port A configured as an output port (when in Mode 1) along with the control word and control signals (for handshaking with a peripheral). When the control word as shown in the figure21A, is loaded into the control register. Group A is configured in Mode 1 with Port A as an output port. The CPU can send out data to a peripheral (like a display device) through Port A of the 8255.

The $\overline{OBF_A}$ output (Output Buffer Full) goes low on the rising edge of the $\overline{WR}$ signal (when the CPU writes data into the 8255). The $\overline{OBF_A}$ output from 8255 can be used as a strobe input to the peripheral to latch the contents of Port A. The peripheral responds to the receipt of data by

making the $\overline{ACK_A}$ input of the 8255 low, thus acknowledging that it has received the data sent out by the CPU through Port A. The $\overline{ACK_A}$ low reset the $\overline{OBF_A}$ signal, which can be polled by the CPU through $\overline{OBF_A}$ of the status word to load the next data when it is high again.

$INTR_A$ is an active high output of the 8255 which is made high (is the associated INTE flip-flop is set) when $\overline{ACK_A}$ is made again by the

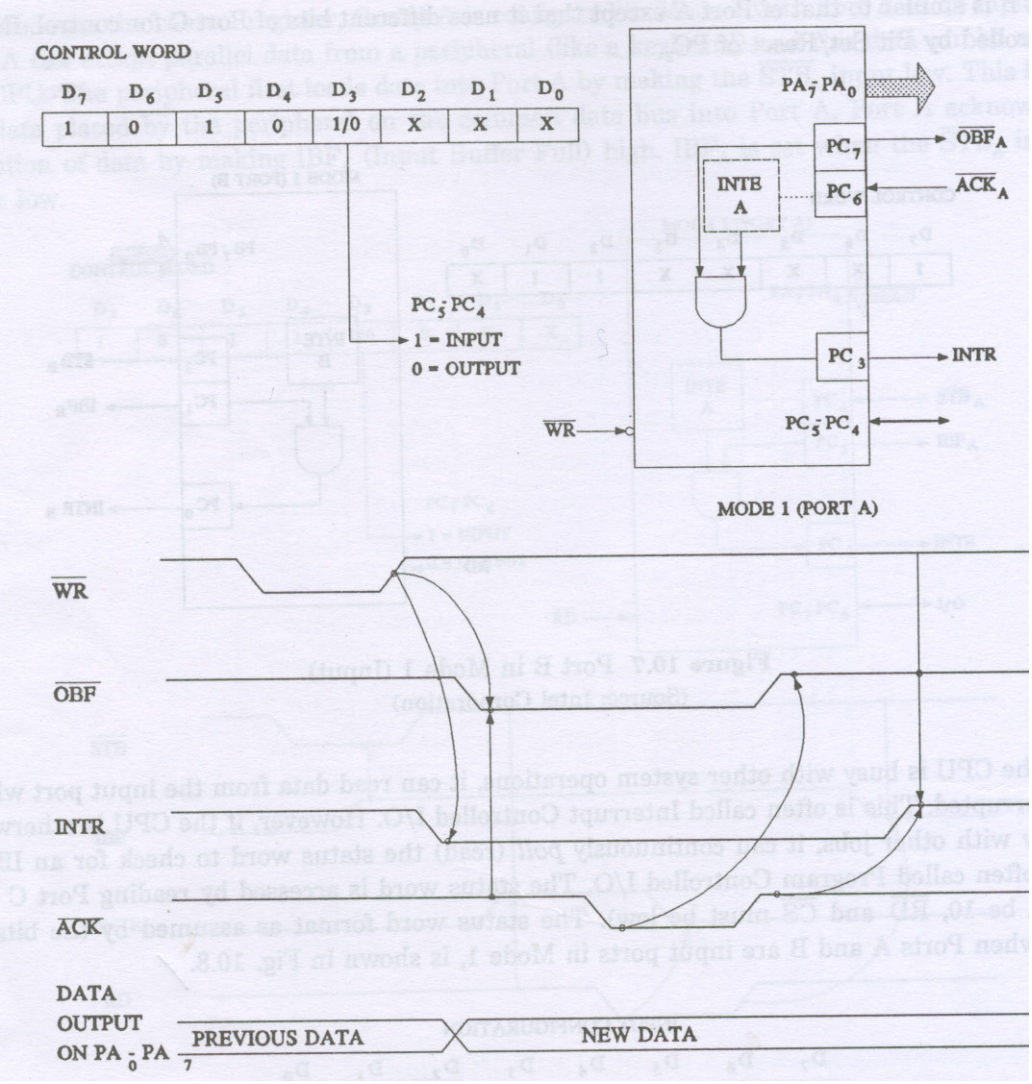


Figure 21A: 8255 PORT A IN MODE 1 (OUTPUT)

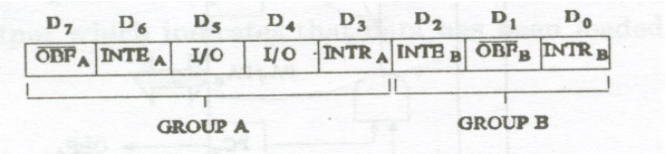


Figure 21B: MODE 1 STATUS WORD (OUTPUT)

peripheral, and when $\overline{OBF_A}$ goes high again. It can be used to interrupt the CPU whenever the output buffer is empty. It is reset by the falling edge of $\overline{WR}$ when the CPU writes data onto Port A. It can be enabled or disabled a '1' or a '0' respectively to PC_6 in the BSR mode. The operation of Port B is similar to the of Port A. $INTE_B$ is controlled by writing a '1' or a '0' to PC_2 in the BSR mode. The status word format as assumed by the bits of Port C when Ports A and B are output ports in Mode 1, is shown in the figure 21B.

7.4 Programming in Mode 2 (Strobed Bidirectional Bus I/O)

When the 8255 is operated in Mode 2 (by loading the appropriate control word), Port A can be used as a bidirectional 8-bit I/O bus using PC_3-PC_7 for handshaking. Port B can be programmed only in Mode 0 (PC_0-PC_2 as Input or Output), or in Mode 1 (with PC_0-PC_2 for handshaking).

Figure 22A shows the control word that would have to be loaded into the control word register to configure 8255 in Mode 2.

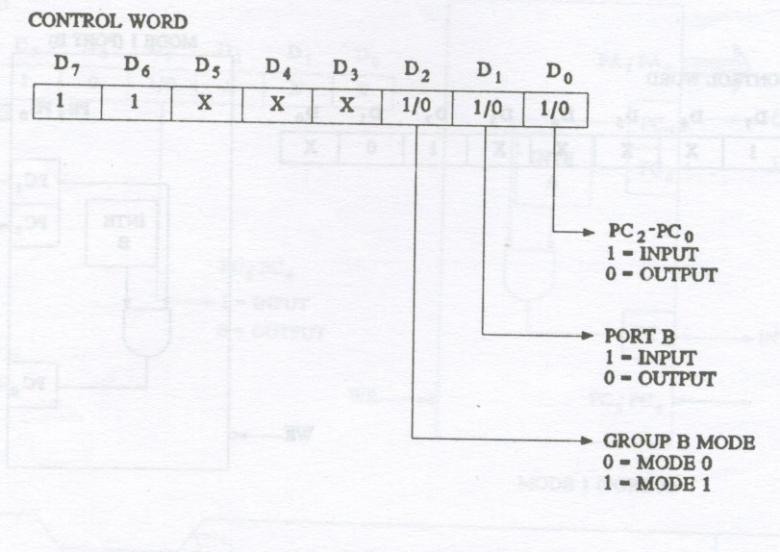


Figure 22A: 8255 MODE 2 CONTROL WORD

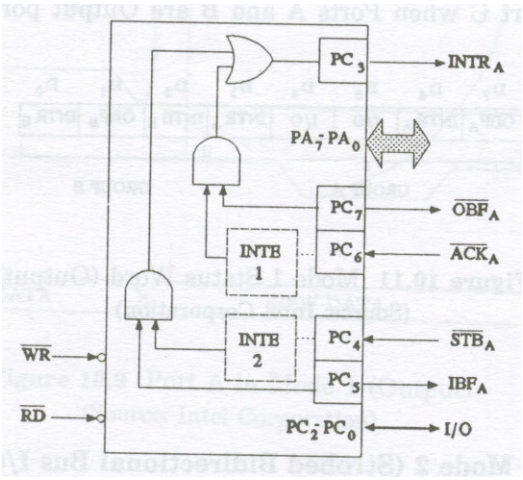


Figure22B: MODE 2 OPERATION

$\overline{\text{OBF}}$ (OUTPUT BUFFER FULL)

This is an active low output which indicates that the CPU has written data into Port A.

$\overline{\text{ACK}}$ (ACKNOWLEDGE)

This is an active low input signal (generated by the peripheral) which enables the tri-state output buffer or Port A and makes Port A data available to the peripheral. In Mode 2, Port A outputs e in tri-state until enabled.

INTE 1

This is the flip-flop associated with Output Buffer Full. INTE 1 can be used to enable or disable the interrupt by setting resetting PC_6 in the BSR Mode.

INPUT CONTROL SIGNALS

$\overline{\text{STB}}$ (STORBE INPUT)

This is an active low input signal which enables Port A to latch the data available at its input.

IBF (Input Buffer Full Flip-Flop)

This is an active high output which indicates that data has been loaded into the input latch of Port A.

INTE 2

This is in Interrupt enable flip-flop associated with Input Buffer Full. It can be controlled by setting or resetting PC_4 in the BSR Mode.

STATUS WORD IN MODE 2

The status word for Mode 2 (accessed by reading Port C) is shown in the figure22C .The status word carry information about $\overline{OBF_A}$, $INTE_1$, IBF_A, $INTE_2$, and $INTR_A$. The status of the bits D_2-D_0 depend on the mode setting of Group B. If B is programmed in Mode 0, D_2 –D_0 are the same as PC_2-PC_0 (simple I/O); however is B is in Mode 1, D_2 –D_0 carry information about the control signals for B depending upon whether B is an Input port or Output port respectively.

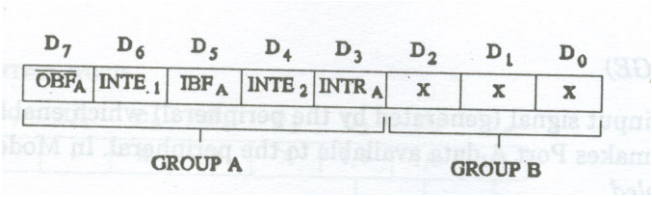
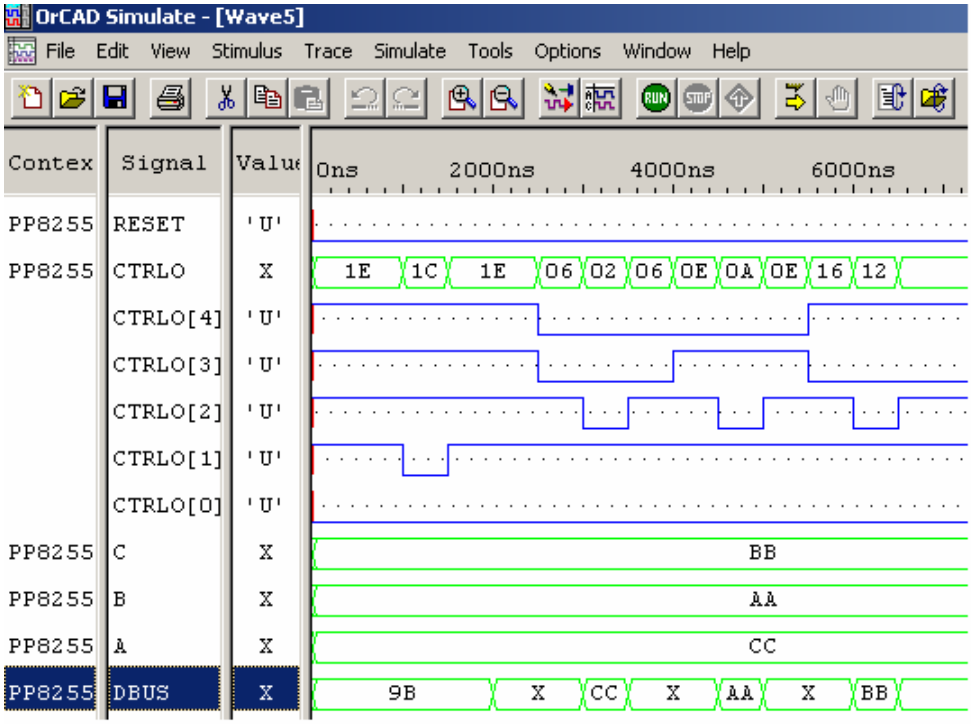
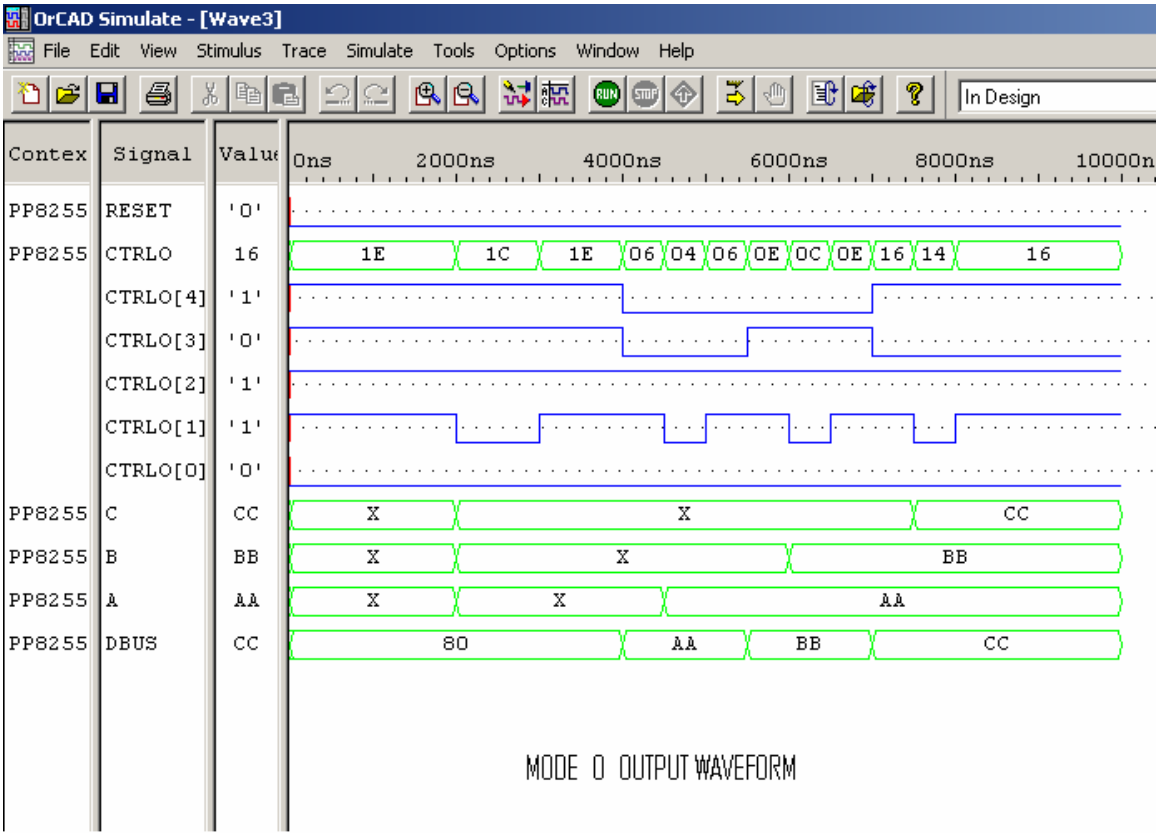


Figure 22C: STATUS WORD FOR MODE 2

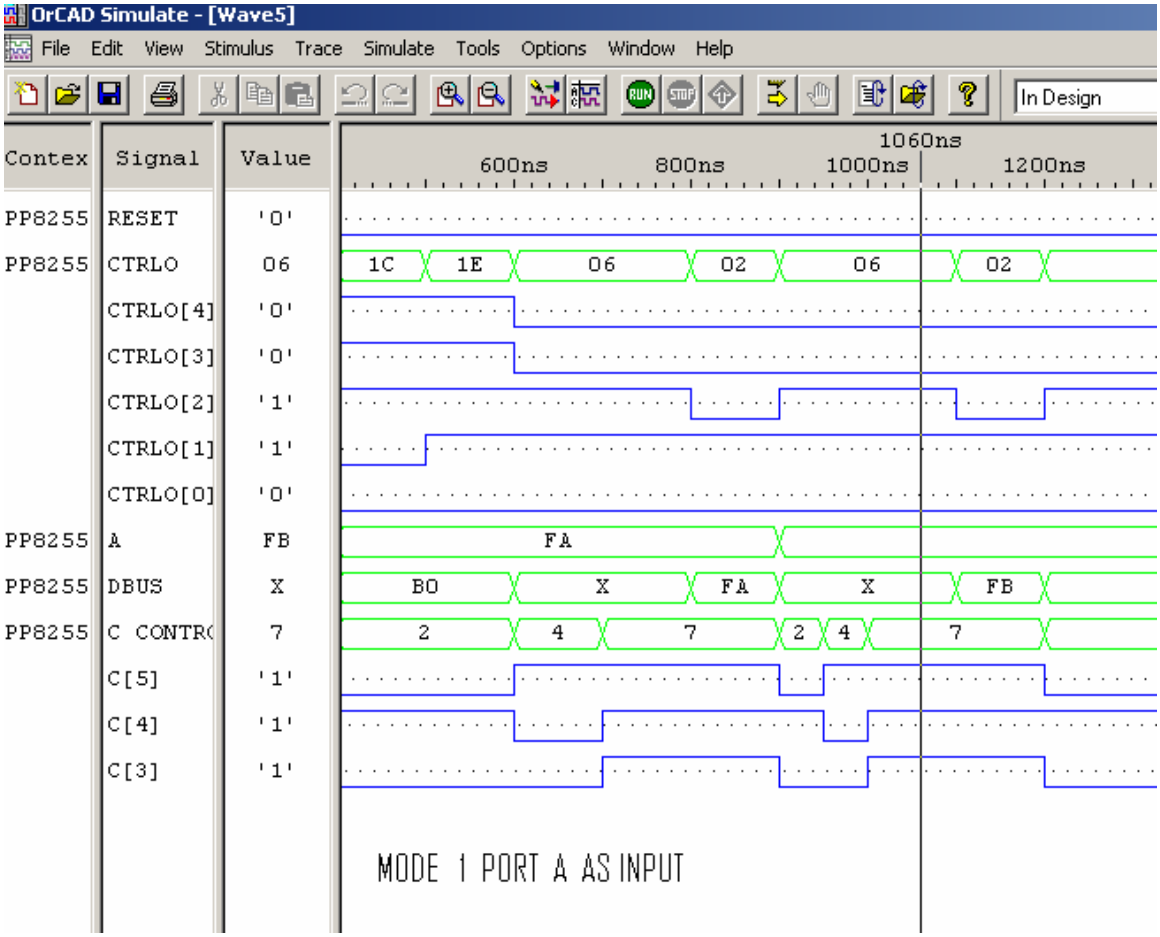
8 SIMULATION RESULTS

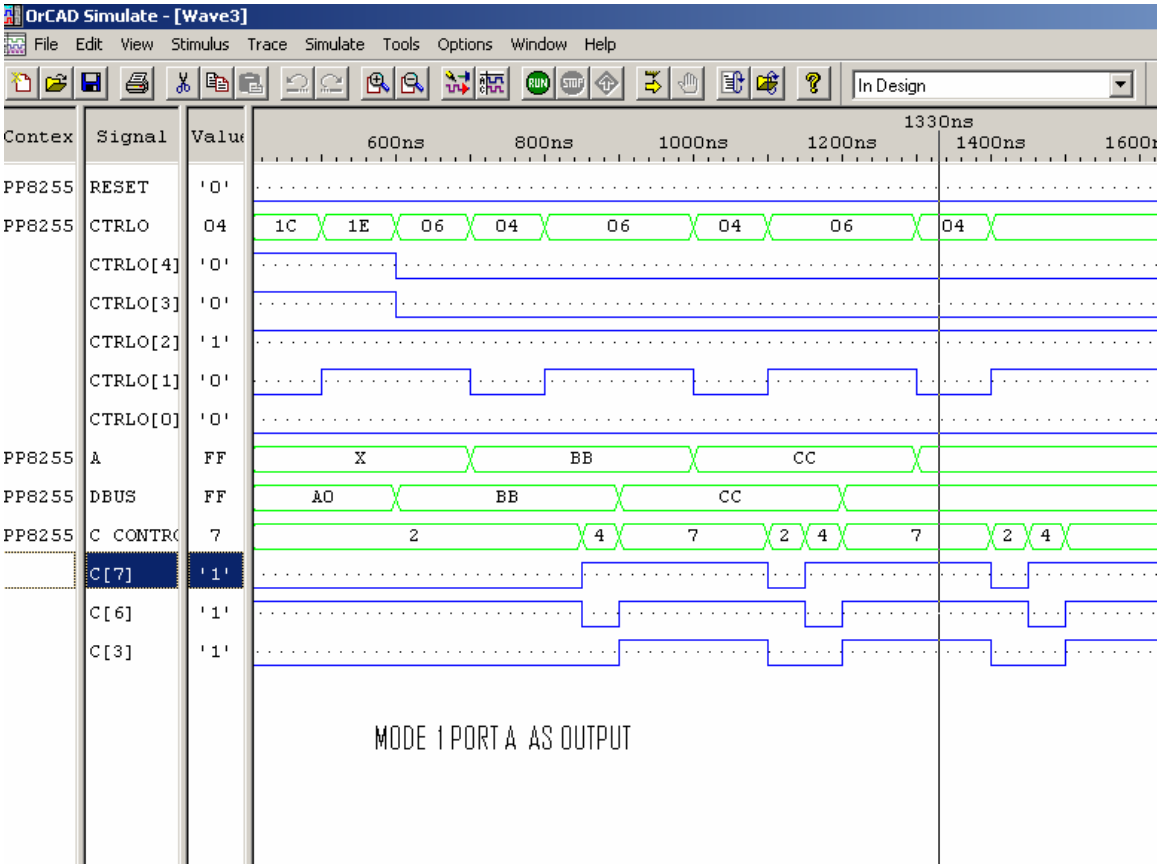


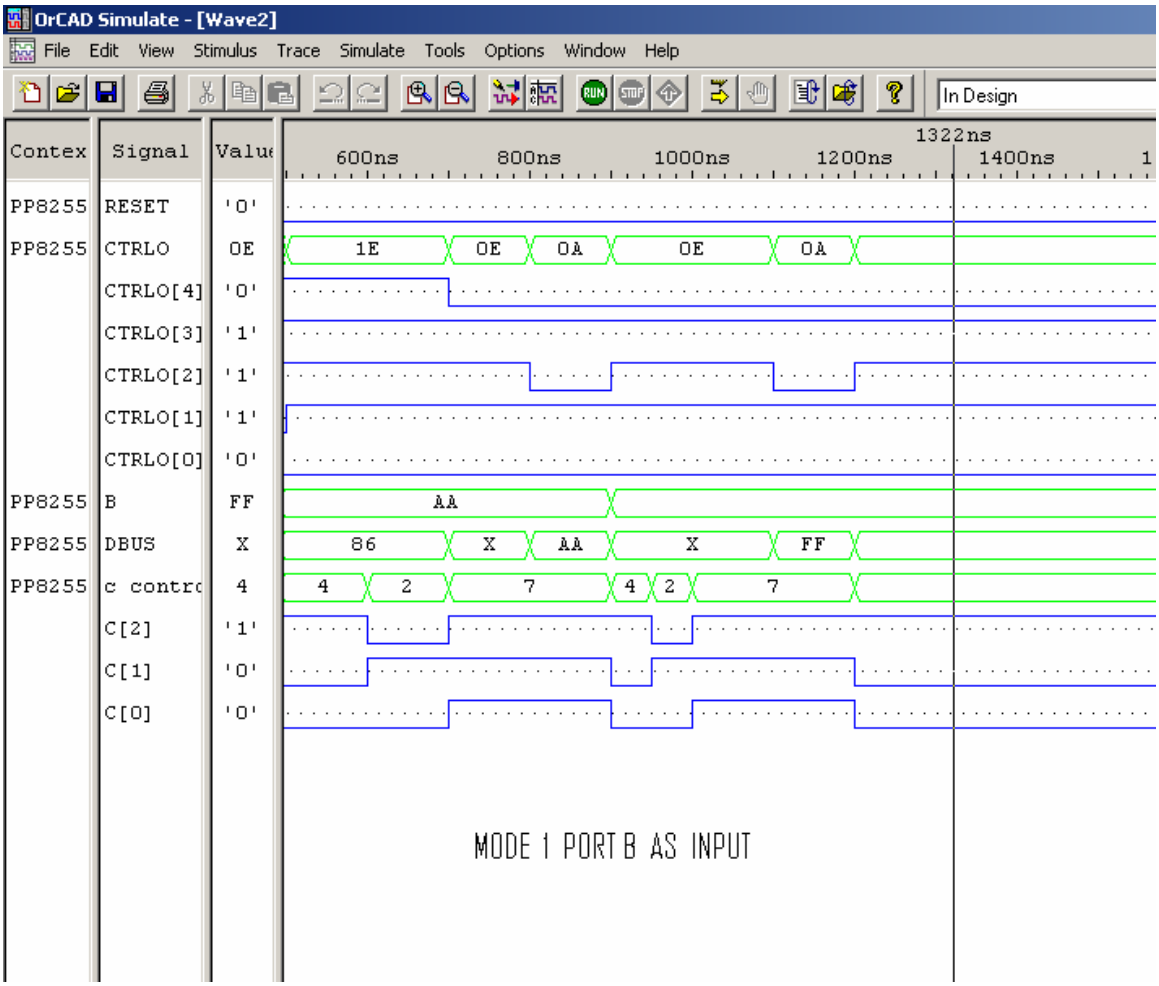
MODE 0 INPUT WAVEFORM

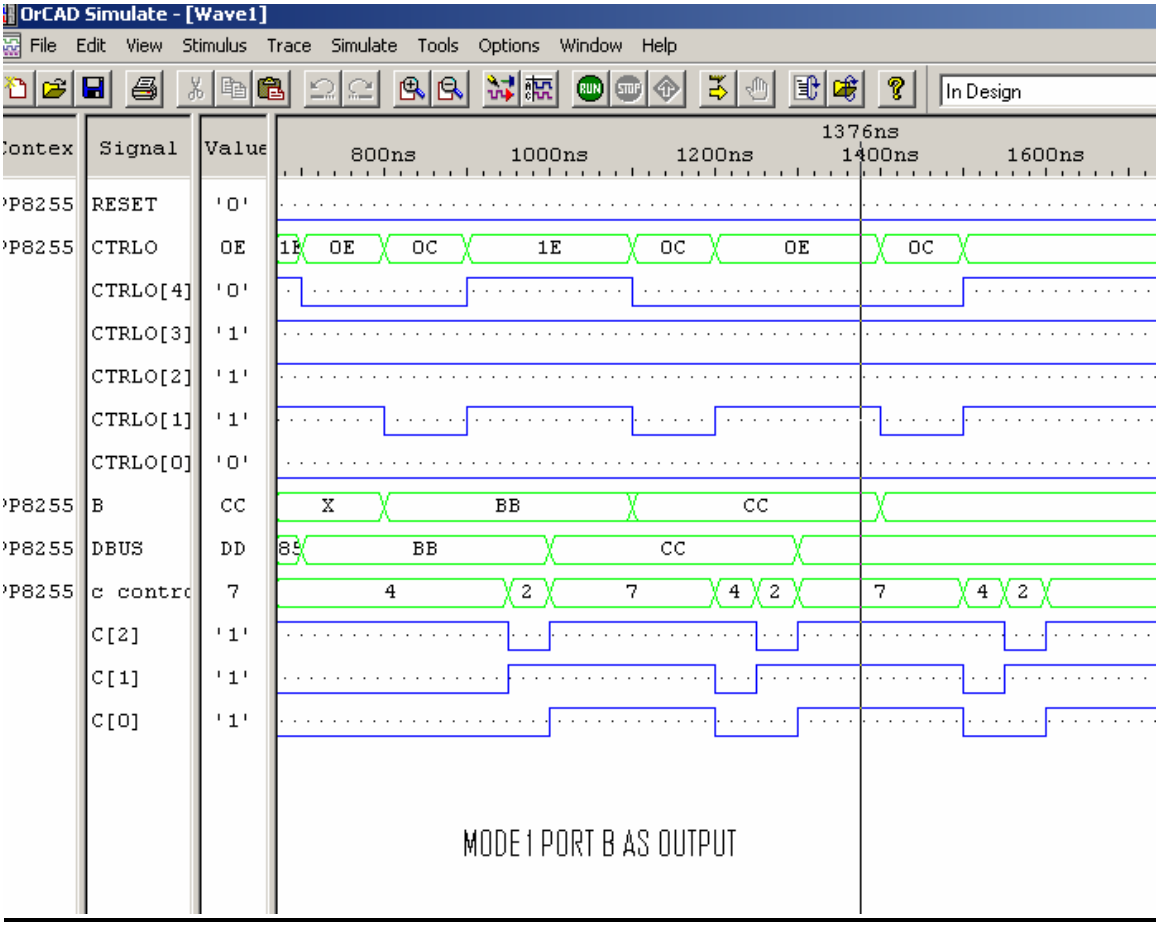


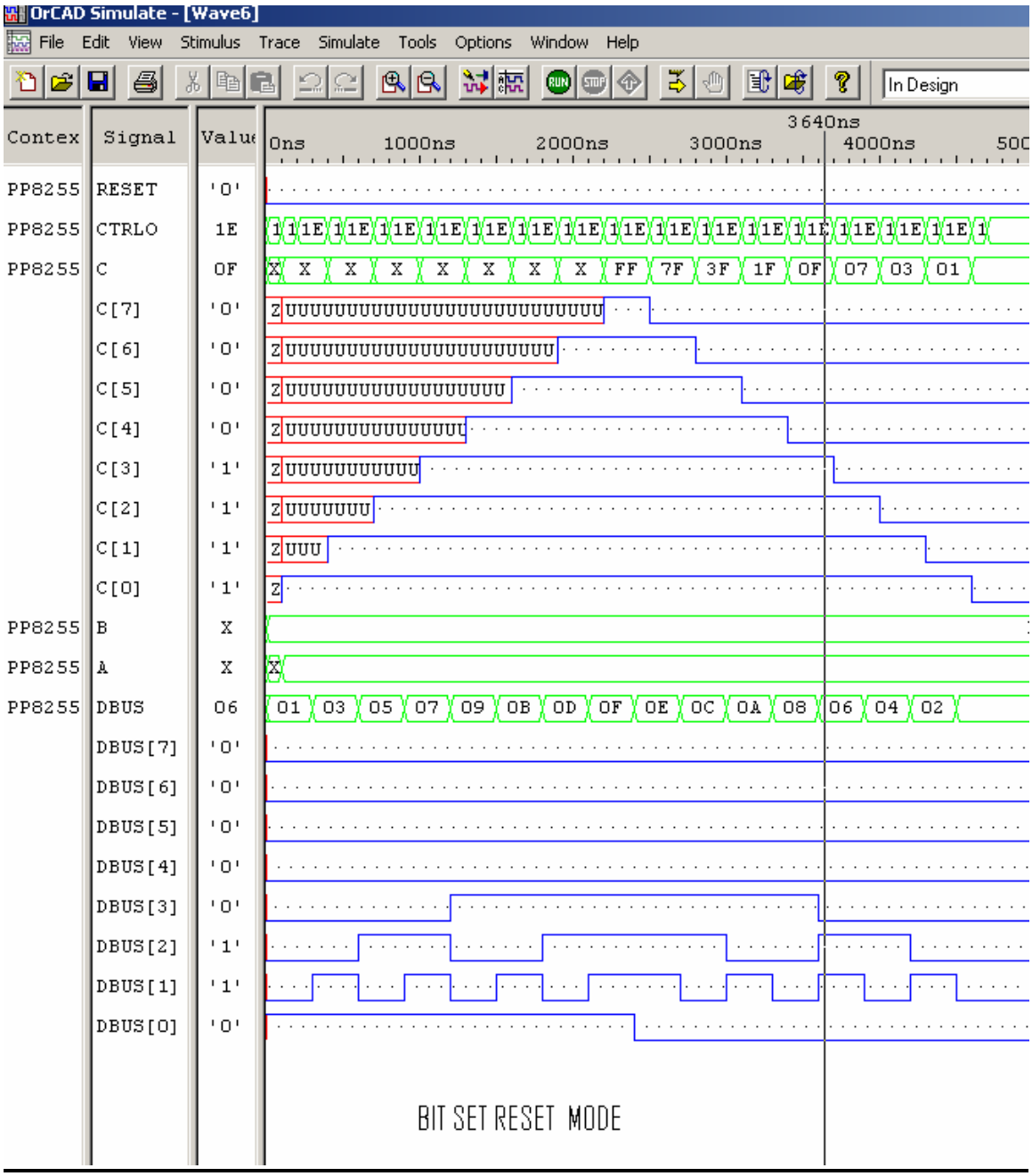
MODE 0 OUTPUT WAVEFORM


OrCAD Simulate - [Wave5]
File Edit View Stimulus Trace Simulate Tools Options Window Help
In Design
Contex
Signal
Value
1060ns
600ns
800ns
1000ns
1200ns
PP8255 RESET '0'
PP8255 CTRLO 06
CTRLO[4] '0'
CTRLO[3] '0'
CTRLO[2] '1'
CTRLO[1] '1'
CTRLO[0] '0'
PP8255 A FB
PP8255 DBUS X
PP8255 C CONTR(7
C[5] '1'
C[4] '1'
C[3] '1'
1C 1E 06 02 06 02
FA
B0 X FA X FB
2 4 7 2 4 7
MODE 1 PORT A AS INPUT


OrCAD Simulate - [Wave3]
File Edit View Stimulus Trace Simulate Tools Options Window Help
In Design
Contex
Signal
Value
600ns
800ns
1000ns
1200ns
1330ns
1400ns
PP8255 RESET '0'
PP8255 CTRLO 04
1C 1E 06 04 06 04 06 04
CTRLO[4] '0'
CTRLO[3] '0'
CTRLO[2] '1'
CTRLO[1] '0'
CTRLO[0] '0'
PP8255 A FF
X BB CC
PP8255 DBUS FF
A0 BB CC
PP8255 C CONTR(7
2 4 7 2 4 7 2 4
C[7] '1'
C[6] '1'
C[3] '1'
MODE 1 PORT A AS OUTPUT


OrCAD Simulate - [Wave2]
File Edit View Stimulus Trace Simulate Tools Options Window Help
In Design
Contex
Signal
Value
600ns
800ns
1000ns
1200ns
1322ns
1400ns
PP8255 RESET '0'
PP8255 CTRLO 0E
CTRLO[4] '0'
CTRLO[3] '1'
CTRLO[2] '1'
CTRLO[1] '1'
CTRLO[0] '0'
PP8255 B FF
PP8255 DBUS X
PP8255 c contrc 4
C[2] '1'
C[1] '0'
C[0] '0'
1E
0E
0A
0E
0A
AA
86
X
AA
X
FF
4
2
7
4
2
7
MODE 1 PORT B AS INPUT


OrCAD Simulate - [Wave1]
File Edit View Stimulus Trace Simulate Tools Options Window Help
In Design
Signal
Value
800ns
1000ns
1200ns
1376ns
1400ns
1600ns
RESET
CTRLO
CTRLO[4]
CTRLO[3]
CTRLO[2]
CTRLO[1]
CTRLO[0]
B
DBUS
C[2]
C[1]
C[0]
OE
OC
1E
BB
CC
MODE 1 PORT B AS OUTPUT


OrCAD Simulate - [Wave6]
File Edit View Stimulus Trace Simulate Tools Options Window Help
In Design
Contex
Signal
Value
3640ns
0ns
1000ns
2000ns
3000ns
4000ns
PP8255 RESET '0'
PP8255 CTRLO 1E
PP8255 C OF
C[7] '0'
C[6] '0'
C[5] '0'
C[4] '0'
C[3] '1'
C[2] '1'
C[1] '1'
C[0] '1'
PP8255 B X
PP8255 A X
PP8255 DBUS 06
DBUS[7] '0'
DBUS[6] '0'
DBUS[5] '0'
DBUS[4] '0'
DBUS[3] '0'
DBUS[2] '1'
DBUS[1] '1'
DBUS[0] '0'
BIT SET RESET MODE

9 CONCLUSION

Simulation results confirm the successful implementation of the 8255 Programmable Peripheral Interface. With the help of VHDL programming coding and downloading on a CPLD integrated circuit, the usage of the semi-custom chip was enhanced to a high degree, making it suitable even for a 8255 PPI and its related applications.

10 DISCUSSION AND FURTHER WORK

CPLD' s provide designer flexibility to maintain, modify, and upgrade a design from time to time because of it's in system programmability. VHDL being technology independent provides the designer the ease to port his design to different technologies .These two features make the digital system design more simplified. Every implementation has its strengths and weakness. Full Custom ; Application specific integrated circuit (ASIC) implementation is very good in terms of chip area , speed and power consumption but it takes a lot of time to complete a design. On the other hand using the semi-custom approach (CPLD) the designer can tailor the general algorithm of the design according to the need on the fly ; the price paid is in terms of chip area , speed and power consumption. But this price is very less when when we see from the angle of time to market and flexibility of design. The current trends in the CPLDS design show shrinking area more logic density higher speeds and lower power consumption. So we can say that programmability feature of CPLDS weighs more in favour of the designer.

10.2 Further work

I have made only the ports available with 8255 With its respective functionality .It can be integrated with a signal processing unit. The

signal processing unit can be implemented on the same CPLD on which 8255PPI was implemented. The logic density of the CPLD are increasing with every new release. In time to come we can have CPLD with enough Logic density to accommodate Digital to Analog converter and Analog to Digital converter and signal processing unit on the same CPLD.

BIBLIOGRAPHY

1.Vhdl" by Douglas L.Perry , McGraw-Hill Publication

2. "Vhdl for Designers" by Lennart Lindh ,Prentice Hall Publication

3. "Vhdl Analysis and Modelling Of Digital Systems" by Z.Navabi , MacGraw Hill Publication

4. "Vhdl Coding Styles and Methodologies" – Ben Cohen , Springer Publication.

5. "A Designers guide to Vhdl" – Peter J Ashenden , Morgan Kaufmann Publication

6. "Vhdl Primer" - J.Bhaskar, Prentice Hall Publication

7. " A Vhdl Synthesis Primer" - J.Bhaskar, Star Galaxy Publication

8. "Vhdl for Programmable Logic" – Kevin Skahill , Addison-Wesley Publication

9. Intel Manual of 8255 PPI.

10. www.xilinx .com/product/coolpld/