Using a Programmable Logic Device As a System Controller In an I/O Bus Based System(1)

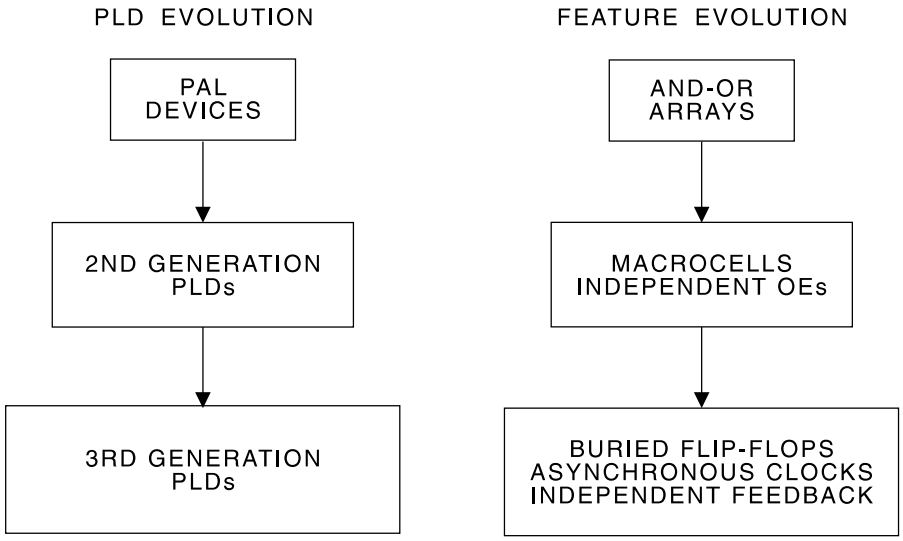
Summary

As Programmable Logic Devices (PLDs) become more complex, the amount of logic that can be placed in one device is rapidly increasing. Complete controllers and subsystems now fit into one or two PLDs. As a result, the PLD may now be connected directly to the system bus as an independent peripheral. First generation PAL® devices are difficult to use in these applications. However, recent innovations in PLD architecture enable them to be easily designed into bus-based systems.

PLD Evolution

The driving force behind PLD usage has been to integrate as much of the Small Scale Integration (SSI) logic on a packed PC board as possible. The first level of integration was made possible by the invention of the PAL device. First generation products were usually in 20-pin packages with a typical device having nine dedicated inputs and eight dedicated outputs. One input pin was a dedicated output enable, and one pin a dedicated, common clock for up to eight flip-flops. Making one of these devices work on an I/O bus was difficult and typically was used as little more than a simple latch.

In the mid-eighties, second generation devices appeared. These PLDs are generally in 24 or more pins, have independent output enable controls and *output macrocells*.



Note: 1. This article originally appeared in Northcon '86 ®PAL is the registered trademark of Monolithic

Erasable Programmable Logic Device

Application Note

Rev. 0483C–09/99

The macrocells allow the designer to configure each output independently as registered or combinatorial. However, there are still too few registers in these devices to allow the design of complex state machines. Also, these circuits lack independent feedback paths, which further reduce the usable number of registers. This also complicates the use of the output pins as true I/O structures.

Recently several third generation devices (such as the Atmel ATV750) have appeared. These devices are differentiated by the following features.

Extra Registers

Up to twice the usual number. The ATV750 has 20 flip-flops.

Independent Feedbacks

Feedback paths for the registers are independent of the output configuration. In addition, there are separate input paths from the I/O pads.

Asynchronous Clocks

Product terms for each flip-flop's clock allows the designer to break up the registers into different functional blocks.

Control function outputs that have no other use than to manage the other resources inside the PLD need not be brought outside the device, allowing implementation of complex state machines internally.

As PLDs have evolved, so have the applications for them. Initially, PLDs could only integrate a few SSI functions. A typical application was a special-purpose decoder or encoder. With the introduction of more flip-flops, Medium Scale Integration (MSI) functions such as state machines could be designed. Third generation devices are the first true Large Scale Integration (LSI) devices, and are capable of integrating several of the previous generation devices into one package. Now state machines can be combined with an output decoder to control peripheral functions, and still have adequate resources to interface directly to the microprocessor.

System Application

The following example is an application of the Atmel ATV750 as a peripheral resource controller. The design required a state machine, a bus interface unit and a peripheral control interface. All ten outputs of the ATV750 are used, most in the combinatorial mode. However, the 17 required flip-flops were still available to latch the address and data buses, provide a status register, and a two-bit counter. This design would require three second generation, 24-pin PLDs, or five first generation 20-pin devices and at least two other discrete devices. In all, more than 80% of the ATV750 is utilized. The number of gates alone integrated into the ATV750 in this application is more than other 24-pin PLD's have to offer.

The System

The system described is a peripheral controller/bus interface for connecting a special-purpose, custom encryption/exponentiation chip to an 80186 microprocessor (Figure 1). The custom chip has a serial interface, and only one bi-directional pin to indicate its "busy" status. All chip functions are controlled with a set of single-purpose input pins. While simple, this interface is not directly compatible with a modern microprocessor, such as the 80186. The PLD system described not only combines the required glue logic, but also off-loads the parallel-to-serial conversion from the processor. This application note will only touch on the salient features of the design, and why a third generation device is so useful.

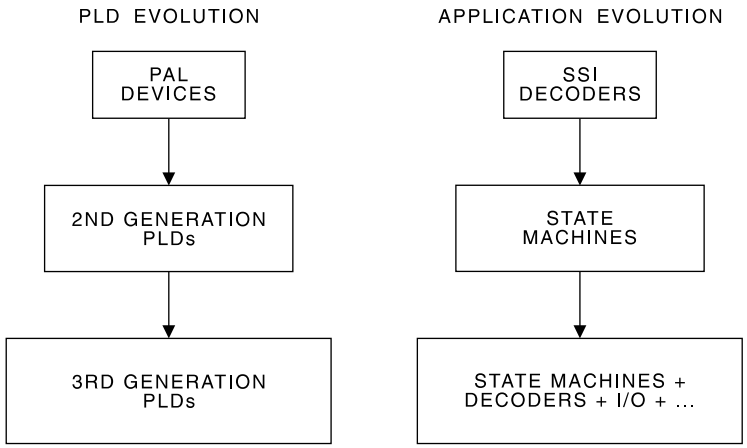
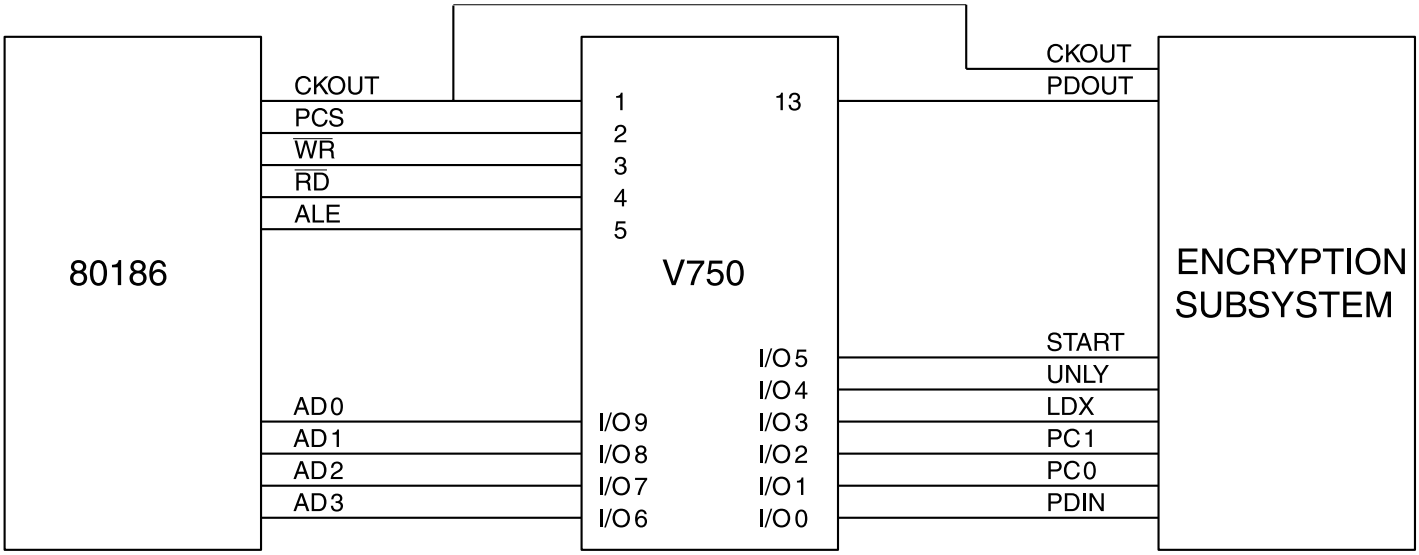


Figure 1. System Diagram



The Microprocessor Bus

The 80186 uses a multiplexed address and data bus. Several control signals, such as ALE, $\overline{RD}$ and $\overline{WR}$ tell the system when to get what type of data from the bus. The 80186 also has some internally decoded chip selects, and one is used here for convenience. The system clock is an 8 MHz signal, which is appropriate for the encryption chip and well within this PLD's timing specification. The lower four bits of the address are latched into the PLD to define the upcoming operation, which then allows the PLD to output the requested data in one read cycle of the microprocessor. These address bits are decoded to define the instruction to be executed by the PLD subsystem.

Tackling the I/O Bus

Using first and second generation PLDs, the equations for the I/O bus interface are shown in Figure 2. These equations consume twelve sum terms, eight flip-flops, and twelve output pins. Since this requires two PLDs, another ten input pins are required as well. When rewritten for the ATV750, only four macrocells are required, and eight sum terms and flip-flops. No extra inputs are required, as the ATV750's I/Os are true input/output pins.

The equations for the ATV750 are in Figure 3. This compaction is possible for three reasons:

1. The individual product terms for OE permit the pin to be used as both an input and output.
2. The three feedback paths allow both registers to be used while the pin status is still available to the array.
3. The product term for the flip-flop clock means that the sum term for one of the flip-flops can be shared between the D input and the output pin. **A single ATV750 macrocell can incorporate logic which would require up to three output pins and one input pin in other PLDs.**

Figure 2.

```
ad0=adp0 & !pcs & !rd & ai0 & !ai2              " output data
  # !pc0 & !pcs & !rd & !ai0 & !ai2             " status "
  # yst & !pcs & !rd & !ai0 & !ai2;             " status unly

ad1=adp1 & !pcs & !rd & ai0 & !ai2              " output data
  # !pc1 & !pcs & !rd & !ai0 & !ai2;            " status "

ad2=adp2 & !pcs & !rd & ai0 & !ai2              " output data
  # xst & !pcs & !rd & !ai0 & !ai2              " status "
  # !pc0 & !pcs & !rd & !ai0 & !ai2             " status "
  # !pc1 & !pcs & !rd & !ai0 & !ai2;            " status  "

ad3=adp3    & !pcs & !rd & ai0 & !ai2           " output data
  # startqb & !pcs & !rd & !ai0 & !ai2;         " status "

adp0:=ad0 & !pcs & !wr                          " load data
  # pdout & !ystb                               " circulate y"
  # pdout & !xstb                               " circulate x"
  # pdout & !pcs0                               " circulate load"
  # pdout & !pcs1                               " circulate load"
  # adp0 & ystb & xstb & pcs0 & pcs1 & pcs      " hold data"
  # adp0 & ystb & xstb & pcs0 & pcs1 & wr;      " hold data"

  adp1:=ad1 & !pcs & !wr                        " load data
  # adp0 & !ystb                                " circulate y"
  # adp0 & !xstb                                " circulate x"
  # adp0 & !pcs0                                " circulate load"
  # adp0 & !pcs1                                " circulate load"
  # adp1 & ystb & xstb & pcs0 & pcs1 & pcs      " hold data"
  # adp1 & ystb & xstb & pcs0 & pcs1 & wr;      " hold data"

adp2:=ad2 & !pcs & !wr                          " load data
  # adp1 & !ystb                                " circulate y"
  # adp1 & !xstb                                " circulate x"
  # adp1 & !pcs0                                " circulate load"
  # adp1 & !pcs1                                " circulate load"
  # adp2 & ystb & xstb & pcs0 & pcs1 & pcs      " hold data"
  # adp2 & ystb & xstb & pcs0 & pcs1 & wr;      " hold data"

adp3:=ad3 & !pcs & !wr                          " load data
  # adp2 & !ystb                                " circulate y"
  # adp2 & !xstb                                " circulate x"
  # adp2 & !pcs0                                " circulate load"
  # adp2 & !pcs1                                " circulate load"
  # adp3 & ystb & xstb & pcs0 & pcs1 & pcs      " hold data"
  # adp3 & ystb & xstb & pcs0 & pcs1 & wr;      " hold data"

ai0:=ad0 & pcs                                  " idle state "
  # ad0 & ale                                   " idle state "
  # ai0 & !pcs & !ale;                          " hold instruction"

ai1:=ad1 & pcs                                  " idle state "
  # ad1 & ale                                   " idle state "
  # ai1 & !pcs & !ale;                          " hold instruction"

ai2:=ad2 & pcs                                  " idle state "
  # ad2 & ale                                   " idle state "
  # ai2 & !pcs & !ale;                          " hold instruction"

ai3:=ad3 & pcs                                  " idle state "
  # ad3 & ale                                   " idle state "
  # ai3 & !pcs & !ale;                          " hold instruction"
```

Figure 3.

```
ai0.ck  =   clk2 & !ale; ai2.ck  =   clk2 & !ale; " clock instruction
ai1.ck  =   clk2 & !ale; ai3.ck  =   clk2 & !ale; " clock instruction

ai0=ad0 & !pcs & ale                              " load instruction"
  # adp0 & !pcs & !rd & ai0 & !ai2                " output data
  # !pc0 & !pcs & !rd & !ai0 & !ai2               " status "
  # yst & !pcs & !rd & !ai0 & !ai2                " status unly

ai1=ad1 & !pcs & ale                              " load instruction"
  # adp1 & !pcs & !rd & ai0 & !ai2                " output data
  # !pc1 & !pcs & !rd & !ai0 & !ai2;              " status "

ai2=ad2 & !pcs & ale                              " load instruction"
  # adp2 & !pcs & !rd & ai0 & !ai2                " output data
  # xst & !pcs & !rd & !ai0 & !ai2                " status "
  # !pc0 & !pcs & !rd & !ai0 & !ai2               " status  "
  # !pc1 & !pcs & !rd & !ai0 & !ai2;              " status  "

ai3=ad3 & !pcs & ale                              " load instruction"
  # adp3    & !pcs & !rd & ai0 & !ai2             " output data
  # startqb & !pcs & !rd & !ai0 & !ai2;           " status "
enable ad0=!pcs & !rd;enable ad2  =               !pcs & !rd;

enable ad1=!pcs & !rd;enable ad3  =               !pcs & !rd;

(adp equations remain the same as before, but are now buried in the macrocell)
```

Figure 4.

```
!pc0=!clk22 & !pcs0;


!pc1=!clk22 & !pcs1;


!pcs0:=ai2 & ai0 & start
  # !cn0 & count & !pcs0
  # !cn1 & count & !pcs0


 !pcs1:=ai1 & ai2 & start
  # !cn0 & count & !pcs1
  # !cn1 & count & !pcs1;
  # !cn1 & count & !pcs1;
```

The Chip Interface

The encryption chip is loaded and unloaded serially, four bits at a time in this design. The equations for the interface logic are in Figure 4. Also in this figure is a simple state diagram for the two bit counter required for this design. This state machine is buried, and its decoded outputs are used to control the serial transfers.

Starting the Peripheral Chip

To begin execution in the peripheral chip, a bi-directional signal named *start* is asserted. This is an active low signal. The controller must assert this signal low for four clock cycles. Then the exponentiation chip will hold this line low until it has completed its operations. An external pull-up resistor is required. The internal flip-flop, whose output is named *stint*, contains the state of the peripheral. This is used to signal the microprocessor that the subsystem is busy when the processor reads the ATV750's status.

Multiplexing Flip-flop Inputs and I/O Pins

One I/O pin/flip-flop combination can be used to store the state of the encryption chip and to output this to the peripheral. This is accomplished by multiplexing the sum term output between the flip-flop's D input and the output buffer. The sum term and the OE product term are active to begin the encryption chip's exponentiation cycle. After the state machine counter finishes counting, the output is put into the high impedance state. If the external chip has begun its operation correctly, it will then hold the pin low. Now the state of the I/O pin is used as the D input to the flip-flop, but not output because the OE term is off. The multiplexed macrocell is in Figure 5. The following simple equations are all that is required to implement this logic:

```
enable start =!count;
stint:=!coun
# !start & count;
start.c=clk2;
```

Conclusion

The application of a third generation EPLD in an I/O bus based system demonstrates the usefulness of the following features:

- **Buried Registers**
- **Independent Feedback Paths**
- **Asynchronous Register Clocks**

This design consists of roughly 600 gates, which fit into a ATV750 gate complexity PLD with 80% utilization factor. Due to the usefulness of the new features and their implementation in the macrocell of the ATV750, this design, which would have required three second generation devices, could easily fit into one ATV750.

Figure 5. Multiplexed "D" Input