

AN1214

MC88110 64-Bit External Bus Interface to 16-Bit EPROM

Introduction

This document describes the design and operation of a 50-MHz interface between the Motorola MC88110 RISC microprocessor and a 16-bit erasable programmable read-only memory (EPROM).

This design has been simulated successfully but has not been built in hardware. The EPROM interface is implemented using two 7-ns programmable array logic (PAL) devices, eight fast logic 8-bit data latches, and a 64K x 16 bit 170-ns EPROM.

This document is organized as follows: 1) the introduction, 2) an overview of the EPROM design with a block diagram of the interface, 3) timing and signal descriptions of all the devices in the interface, 4) a description of the interface hardware design—this section provides a detailed description of the interface logic control diagrams and the detailed schematics of the EPROM interface, and 5) the timing traces of the interface design. Appendix A contains the control and counter state machine state diagrams, the reduced PAL logic equations, and the series of test vectors used to test the operation of the PALs.

Overview

The EPROM interface is designed to be part of a larger system that has numerous peripheral devices connected to the MC88110 bus. The larger system contains the necessary decode logic to select each of the peripheral devices. This logic decodes the address bits and sends an EPROM chip enable ($\overline{\text{EPROM_CE}}$) signal whenever the EPROM is accessed. A functional block diagram of the EPROM interface is shown in Figure 1. The dotted region delineates the interface logic presented in this document.

The MC88110 provides a 32-bit address on its address signals along with the transfer start ($\overline{\text{TS}}$), data bus busy ($\overline{\text{DBB}}$), and read/write ($\text{R}/\overline{\text{W}}$) signals. The EPROM interface presented in this design always responds with a full 64bit data operand.

An additional feature of this design allows for other memory devices (such as a DRAM or SRAM) to be re-mapped to the EPROM's original location in the memory map. An external signal named REMAP facilitates this change. Whenever the EPROM interface detects an EPROM chip select, it checks the REMAP signal to determine if the EPROM has been re-mapped, and thereby decides whether to respond to the access. The EPROM interface quantifies the EPROM chip select with REMAP. If REMAP is not asserted, the EPROM responds to the access. If REMAP is asserted the EPROM does not respond to the access.

This document contains information on a new product. Specifications and information herein are subject to change without notice.



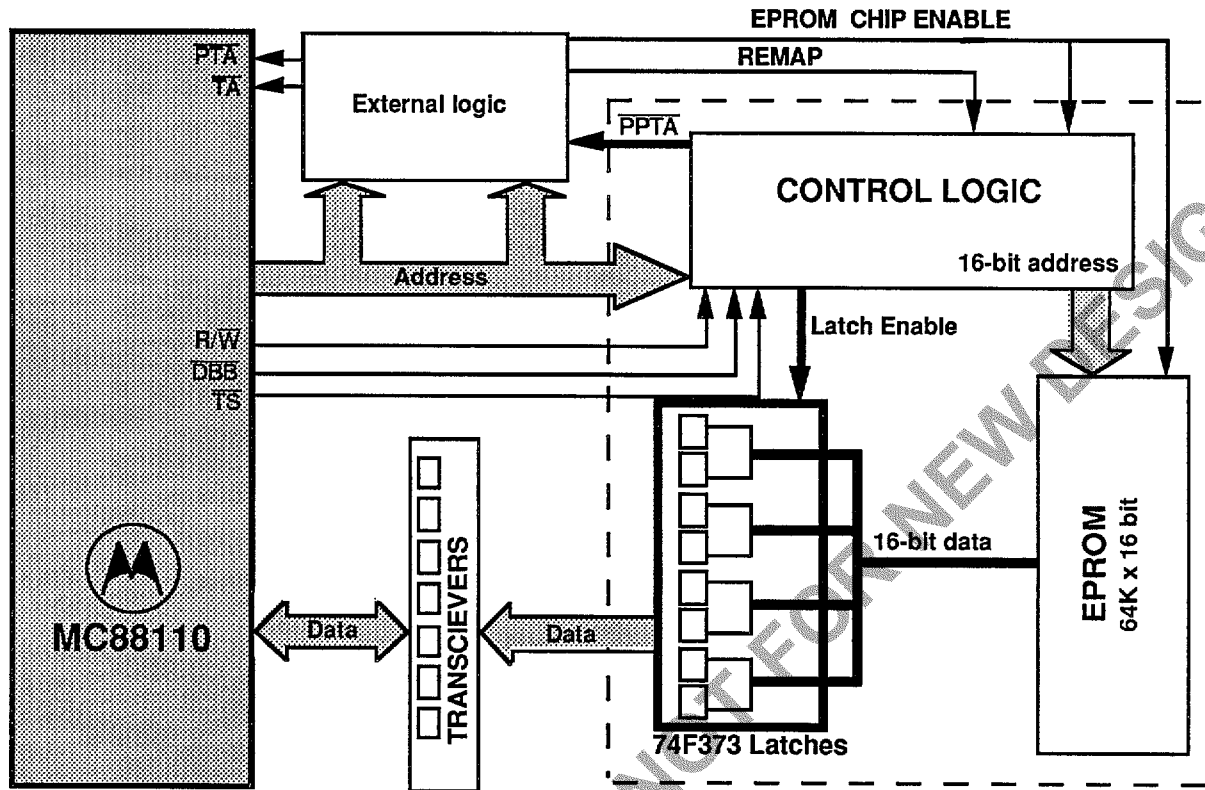


Figure 1. Functional Block Diagram of the EPROM Interface

One reason for using a REMAP signal would be to allow a faster memory to be used in place of the relatively slow EPROM. Users can first copy the EPROM contents into faster memory and then activate the REMAP signal. The REMAP signal prevents the EPROM from responding to future accesses, and allows those accesses to be serviced by the faster memory. The REMAP signal can also be used to execute a user-routine in SRAM during warm reset. Warm reset forces the MC88110 processor to initiate execution at address zero, which usually maps to the EPROM. By re-mapping an SRAM or DRAM to this address after power-up, it is possible to place a modified reset routine at address zero.

This design utilizes a 64K x 16 bit EPROM device that provides 16 bits of data for each 16-bit address applied to its address signals. Thus four EPROM accesses are required to obtain the 64 bits of data requested by the MC88110. The interface logic receives the address and the appropriate transfer attributes from the MC88110. It uses 14 bits (A16-A3) of this address and two explicitly generated bits to provide the necessary 16-bit EPROM address. The two explicitly generated address bits are toggled during each EPROM cycle in order to access four consecutive 16-bit half words from the EPROM. Each 16-bit half-word is latched into a pair of 8-bit latches. After four EPROM cycles, the four pairs of 8-bit latches contain the requested 64-bit double word.

Upon completion of the data transfer, the interface logic sends a pre-transfer acknowledge (\overline{PTA}) and transfer acknowledge (\overline{TA}) signal to the MC88110 to indicate that the transfer is complete. If a write to EPROM is attempted, or if the REMAP signal indicates that the EPROM has been re-mapped, the interface logic does not initiate an EPROM access. It is assumed that an external logic block will issue an error signal to the rest of the system if a write access is attempted to the EPROM. The following sections of this document describe the timing and signal descriptions for both the MC88110 and the EPROM.

Timing and Signal Descriptions

NOTE

In this document, assert and negate are used to specify the setting of a signal to a particular state. Specifically, the terms **assert** and **assertion** refer to a signal that is active or true; **negate** and **negation** refers to a signal that is inactive or false. These terms are used independently of the voltage level (high or low) that they represent. Throughout the remainder of this document, active high signals are denoted by the signal name (for example, SIGNAL), and active low signals are denoted by the signal name with an overbar (for example, $\overline{\text{SIGNAL}}$). All timing delays are given respective to the rising edge of the clock.

As shown in Figure 1, the interface logic uses as input the address bits, $\overline{\text{TS}}$, $\overline{\text{DBB}}$, and $\text{R}/\overline{\text{W}}$, a REMAP signal, and an EPROM_CE, and responds back with 64 bits of data and an acknowledgement signal. The address bits, $\overline{\text{TS}}$, $\overline{\text{DBB}}$, and $\text{R}/\overline{\text{W}}$ are outputs of the MC88110, while the REMAP and EPROM_CE signals are generated by external logic. This EPROM interface has been designed assuming that REMAP is valid 7 ns after address is valid, and that EPROM_CE is valid 17 ns after the address is valid.

Figure 2 is a functional timing diagram of the signals sent from and received by the MC88110. The following subsection describes the timing of the MC88110 as applicable to this design.

MC88110 TIMING

At the start of a bus transaction, the MC88110 asserts the $\overline{\text{TS}}$ signal for one clock cycle. This signal is asserted 0 to 10 ns after the first rising clock edge of the data transfer. The 32 address bits are valid 4 to 15 ns after the rising edge of the first clock. $\overline{\text{DBB}}$, the signal that indicates there is a data bus master, is asserted by the MC88110 0 to 10 ns after the second clock. The $\overline{\text{DBB}}$ signal remains asserted until the data has been latched by the MC88110, and the current data tenure ends.

To signal completion of the data transfer, the MC88110 expects to receive two acknowledge signals—pre-transfer acknowledge ($\overline{\text{PTA}}$) and $\overline{\text{TA}}$. The $\overline{\text{PTA}}$ signal is asserted the clock before the data becomes valid (that is, clock 3 in Figure 2). It has a set-up time of 9 ns and a hold time of -3 ns. The $\overline{\text{TA}}$ signal is asserted during the clock cycle that data is valid (that is, clock 4 in Figure 2). It also has a set-up time of 9 ns and a hold time of -3 ns. The $\overline{\text{DBB}}$ signal negates during the clock after the final $\overline{\text{TA}}$ of the transfer is asserted. As shown in Figure 2, $\overline{\text{DBB}}$ negates 0 to 10 ns after the rising edge of clock 4. The MC88110 requires that input data have a setup time of 9 ns and a hold time of -3 ns.

After receiving the $\overline{\text{TA}}$, the MC88110 negates $\overline{\text{DBB}}$ 0 to 10 ns into the next clock (that is, clock 4 in Figure 2). During back-to-back data accesses, the next data tenure can start immediately if the address bus is parked. The new bus tenure starts with the assertion of $\overline{\text{TS}}$ in the clock cycle after the one in which the $\overline{\text{TA}}$ for the previous tenure has been received (clock 4 in Figure 2).

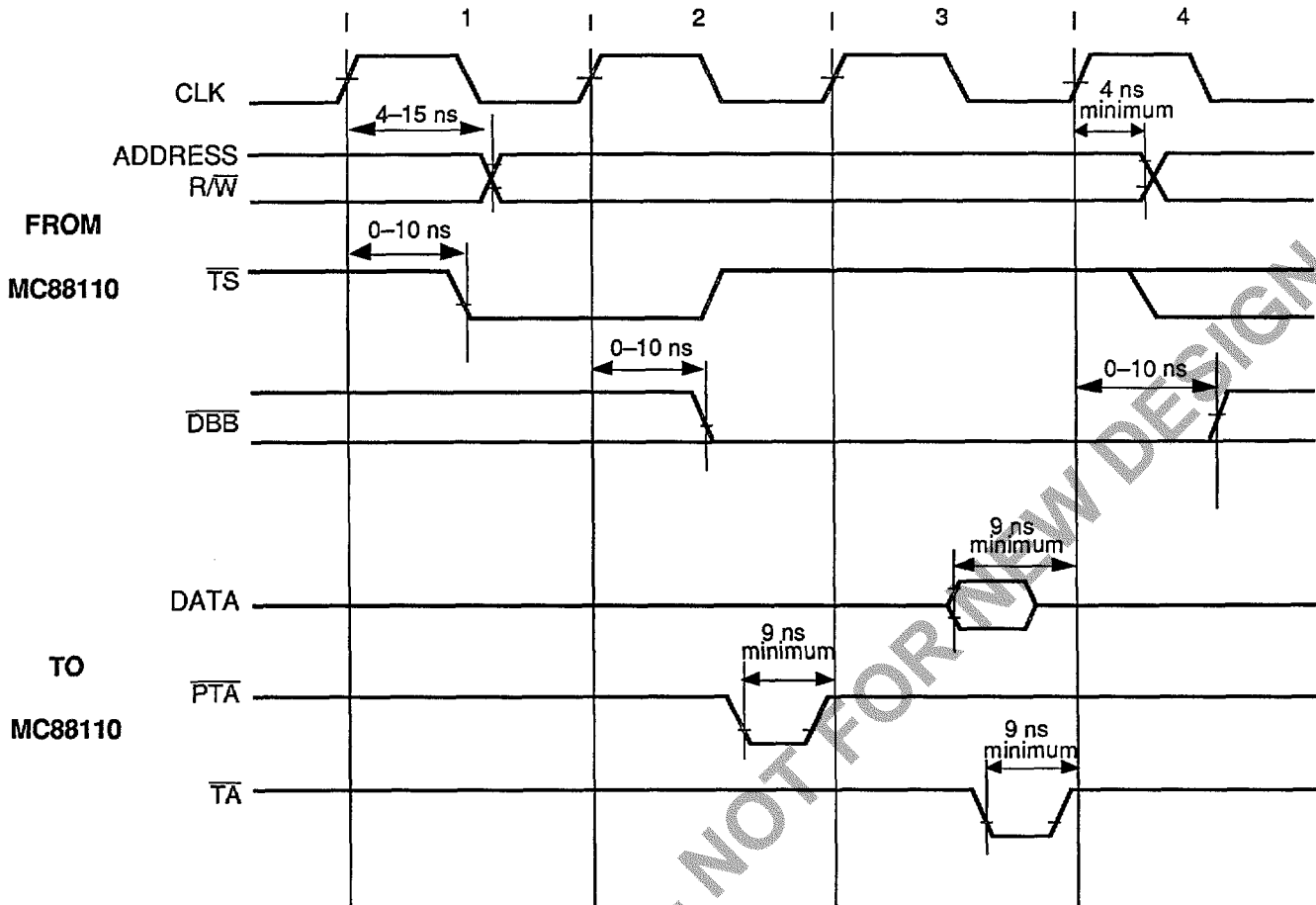


Figure 2. Functional Timing Diagram of Selected MC88110 Signals

For burst accesses, where the MC88110 expects four sets of 64 data bits, the MC88110 retains data bus tenure by asserting \overline{DBB} until four \overline{TA} signals are received. During burst accesses, the MC88110 places the next address on the bus, the clock cycle after it receives the \overline{TA} for the previous word access. The next address becomes valid 4 to 15 ns into this clock cycle.

To signal completion of a transfer, the EPROM interface logic generates a pre-pretransfer acknowledge (\overline{PPTA}) signal. The \overline{PPTA} signal is asserted two clock cycles before the data is valid. Control logic external to the EPROM interface generates the required \overline{PTA} and \overline{TA} signals from the \overline{PPTA} signal.

EPROM TIMING

The EPROM requires a 16-bit address, a chip enable, and an output enable as inputs. The 16-bit address is provided by the interface logic. The chip enable and output enable signals are connected together and obtained from external decode logic. The EPROM guarantees that within 170 ns after an asserted EPROM chip enable and a valid address are received by the EPROM, data will be valid on its data signals.

PAL TIMING

The state machines for the interface logic are implemented using two 16-input PAL devices. Each PAL generates both registered and combinational data, and requires a minimum data setup time of 7 ns and a data hold time of zero nanoseconds. These devices have a maximum delay of 6.5 ns from input to registered output, and 7 ns from input to combinational output. For simplicity, we assume a delay of 7 ns for

both cases. The PALs also have a maximum delay of 3 ns for feedback input. Therefore a combinatorial output equation that contains a feedback input will be output in, at most, 10 ns (that is, 7 + 3).

LATCH TIMING

Four pairs of 8-bit Fast 74F373 latches are used to hold the 64 data bits obtained from the EPROM. In order for the 74F373 to latch in data properly, it is necessary for the data to meet a minimum set-up time of 1 ns before and a minimum hold time of 3 ns after the latch is turned on. These latches have a maximum propagation delay of 8 ns.

Hardware Design

The interface logic was implemented with a control state machine and a counter state machine. The control state machine generates the signals initiating four 16-bit EPROM accesses, while the counter state machine generates signals indicating completion of each EPROM access.

Both state machines are configured to reset to S0 if \overline{DBB} negates anytime during the data transfer operation. It is unnecessary to have an explicit reset signal being sent to the interface logic since the MC88110 negates \overline{DBB} whenever a reset occurs.

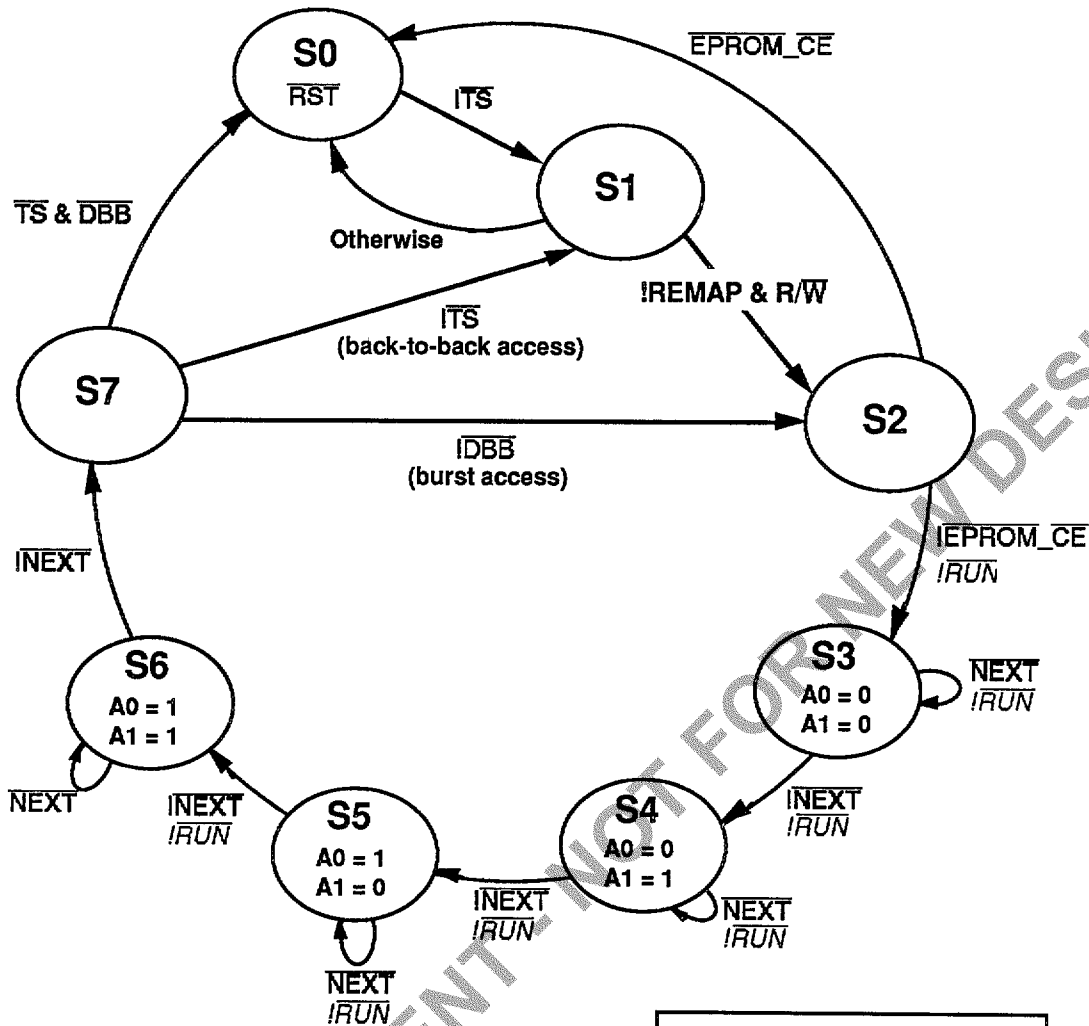
CONTROL STATE MACHINE DESIGN

The control state machine, shown in Figure 3, controls the operations of the interface logic. This state machine checks all data transfers to determine if an EPROM access has been requested. When a valid EPROM access is identified, the control state machine initiates four 16-data bit accesses in order to obtain 64 data bits. This state machine informs the counter state machine to begin counting by sending out an asserted \overline{RUN} signal. The counter state machine returns the completion signal \overline{NEXT} to indicate that one EPROM read access has completed.

The control state machine consists of eight states. The first three states recognize a valid EPROM data transfer. State S0 is the reset state. The state machine remains in this state until a data transaction is signaled by means of the \overline{TS} signal. When \overline{TS} asserts, the state machine transitions to S1. The state graph proceeds to state S2 if the EPROM has not been re-mapped and a read data transfer is occurring. The next transition, from S2 to S3, occurs only if the external address decode logic generates an $\overline{EPROM_CE}$. If $\overline{EPROM_CE}$ is not asserted, then the control state machine resets to state S0. The four states S3 to S6 (referred to in this section as the DATA_READ states) are each used to read 16-bit words into the data latches. The control state machine remains in each DATA_READ state until the counter state machine (shown in Figure 4) indicates that an EPROM read has completed by asserting the \overline{NEXT} signal.

Upon completion of all four accesses, the state machine reaches state S7—the decision state. Depending on the type of the data transfer, the state machine may go to one of three states. It will reset and return to S0 if both \overline{TS} and \overline{DBB} are negated, since this indicates that the data transfer has completed. It will go to state S2 to initiate a new data transfer if \overline{TS} is asserted. This occurs during back-to-back data accesses. It will go to state S1 and continue the current data transfer if \overline{DBB} is still asserted, since this indicates a burst access by the MC88110.

To access different 16-bit words in each DATA_READ state, the least significant bits of the 16-bit EPROM address (that is, bits A1 & A0) are toggled. The two low-order bits of the state variable assignments for the DATA_READ states correspond to the A1 and A0 bits of the EPROM. These low order state bits drive the two least significant address lines to the EPROM.



NOTE:

1. All of the states reset to S0 if $\overline{\text{DBB}}$ negates
2. *Italics* denote outputs
3. Exclamation point (!) indicates the logical NOT

Figure 3. Control State Machine for EPROM Interface

The control state machine was implemented with a TIBPAL16R4. This PAL has 8 inputs, 4 registered outputs, and 4 combinatorial outputs. This PAL receives 7 input signals ($\overline{\text{TS}}$, $\overline{\text{EPROM_CE}}$, $\overline{\text{REMAP}}$, $\overline{\text{R/W}}$, $\overline{\text{DBB}}$) and drives four combinatorial outputs (E1, E2, E3, E4). These combinatorial outputs are used to enable and disable the four pairs of latches. Three of the registered outputs are used for the three state bits needed for the eight states.

COUNTER STATE MACHINE DESIGN

The counter state machine, shown in Figure 4, implements a simple counter that is used to indicate when the EPROM sends valid data. This state machine receives the $\overline{\text{RUN}}$ input from the control state machine, and returns a completion signal, $\overline{\text{NEXT}}$. The assertion of $\overline{\text{NEXT}}$ indicates that one EPROM read access has completed, and that the data has been latched onto the appropriate data latch pair.

The counter state machine consists of 11 states. This state machine remains in state S0 until it receives a $\overline{\text{RUN}}$ signal from the control state machine. The control state machine asserts the $\overline{\text{RUN}}$ signal each time it

transitions to a new DATA_READ state. Upon receiving the $\overline{\text{RUN}}$ signal, the counter state machine proceeds through the remaining 10 states, one clock cycle per state. At the S9 to S10 transition, the state machine sends the completion signal, $\overline{\text{NEXT}}$. This signal is detected by the control state machine when the counter state machine transitions from S10 to S0. Thus, when the counter state machine returns to S0, the control state machine transitions to a new DATA_READ state. The $\overline{\text{NEXT}}$ signal indicates that sufficient time has elapsed for valid data to be latched into the appropriate set of latches.

The counter state machine is implemented with a TIBPAL16R8. This PAL has 8 inputs and 8 registered outputs. Four of these registered outputs are used for the state bits needed to represent the 11 states. Three of the remaining four registered outputs are used to generate $\overline{\text{ENABLE_ON}}$, $\overline{\text{NEXT}}$, and $\overline{\text{PPTA}}$. This PAL receives three inputs ($\overline{\text{RUN}}$, A1, and A0) from the control state machine PAL and one input ($\overline{\text{DBB}}$) from the MC88110.

Operation of the Counter State Machine

The four 16-bit words read from the EPROM are latched into four pairs of 8-bit 74F373 latches. The control state machine asserts and negates the latch enables of each latch pair. The counter state machine issues an $\overline{\text{ENABLE_ON}}$ signal that informs the control state machine when to turn the latch enables on. The control state graph qualifies the $\overline{\text{ENABLE_ON}}$ input with the A1 and A0 address bits to generate the E1, E2, E3 and E4 enable signals for the four pairs of latches. The specific logic equations for these signals are given in Appendix A.

As explained previously, the interface logic handles acknowledgement by generating a $\overline{\text{PPTA}}$ signal two cycles before the data transfer is complete. The $\overline{\text{PPTA}}$ signal is asserted by the counter state machine during its transition from S7 to S8. This signal is only asserted during the last 16-bit word access (that is, the control state machine is in state S6 and A1, A0 equals 11).

The $\overline{\text{EPROM_CE}}$ signal is used to disable the outputs of the 74F373 latches after the EPROM data transfer is complete. This prevents EPROM data from being driven through the transceivers onto the MC88110 bus after the EPROM data transfer is over, thus avoiding the problem of bus contention with the MC88110 writing to other peripherals on the bus at the same time. The final schematics of the EPROM Interface is shown in Figure 5.

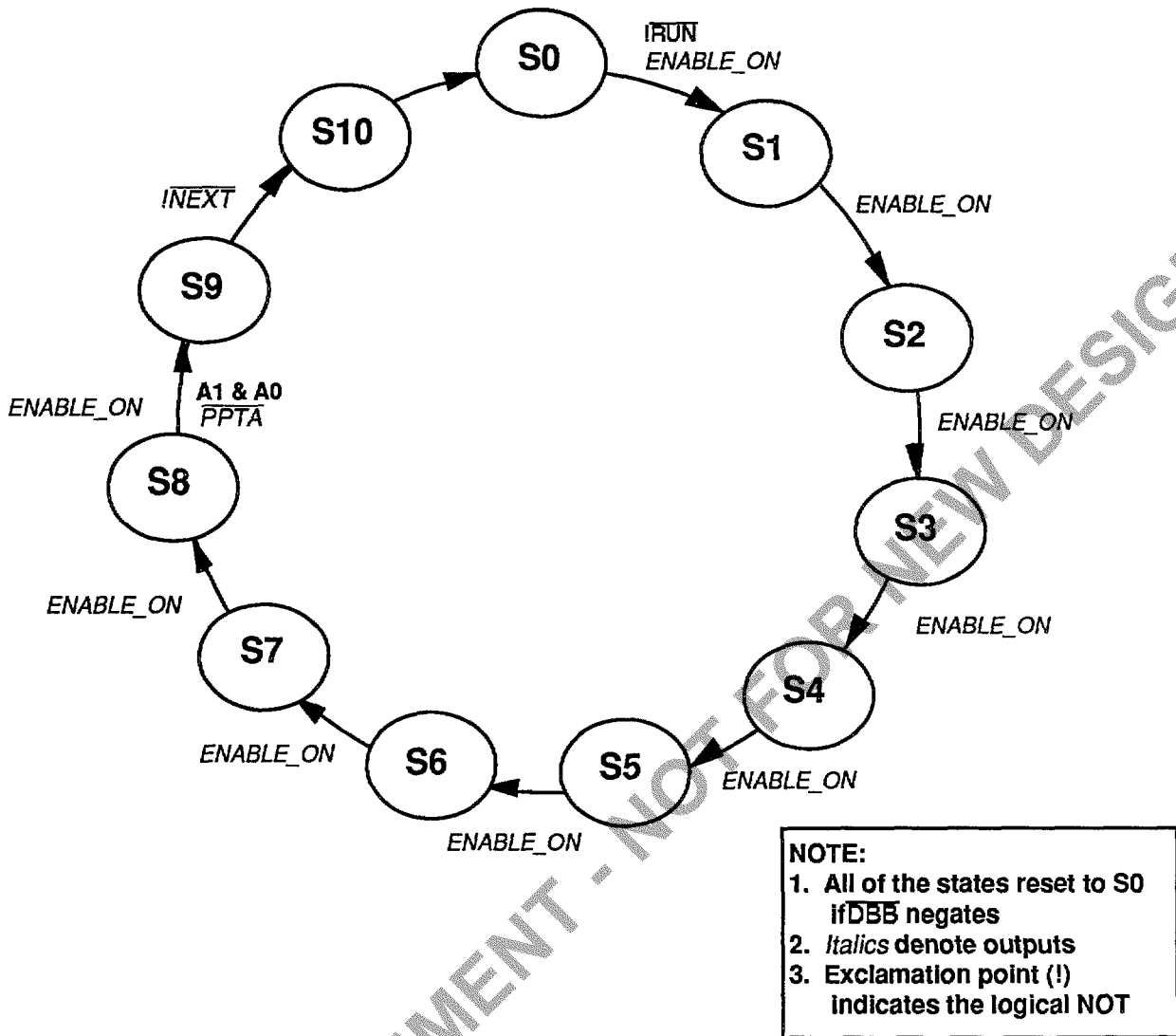


Figure 4. Counter State Machine for EPROM Interface

Timing Analysis

This section discusses some of the timing restrictions encountered in the 50-MHz EPROM interface design, how some of these constraints were met, and what timing margins are provided, taking into account the impact of clock period variations in the MC88915 clock driver that generates the clock signal for MC88110. The EPROM interface presented in this document is designed to work with clocks whose periods may vary up to 1 ns.

In the timing diagrams described in this section, maximum propagation delays are shown for all signals. In some cases, minimum delays are also shown. Additionally, all timing delays are rounded to whole numbers. The following paragraphs explain the operational timing for the EPROM interface. Refer to Figure 6 and Figure 7 for the timing diagrams of the interface. Figure 6 shows the timing for the first of the four EPROM accesses, and Figure 7 shows the last EPROM access.

Both the control state machine (CTL_{SM}), and the counter state machine (CNT_{SM}) remain in state S0 until a data transfer is initiated by the MC88110 by asserting \overline{TS} . The MC88110 asserts \overline{TS} 10 ns after clock 1,

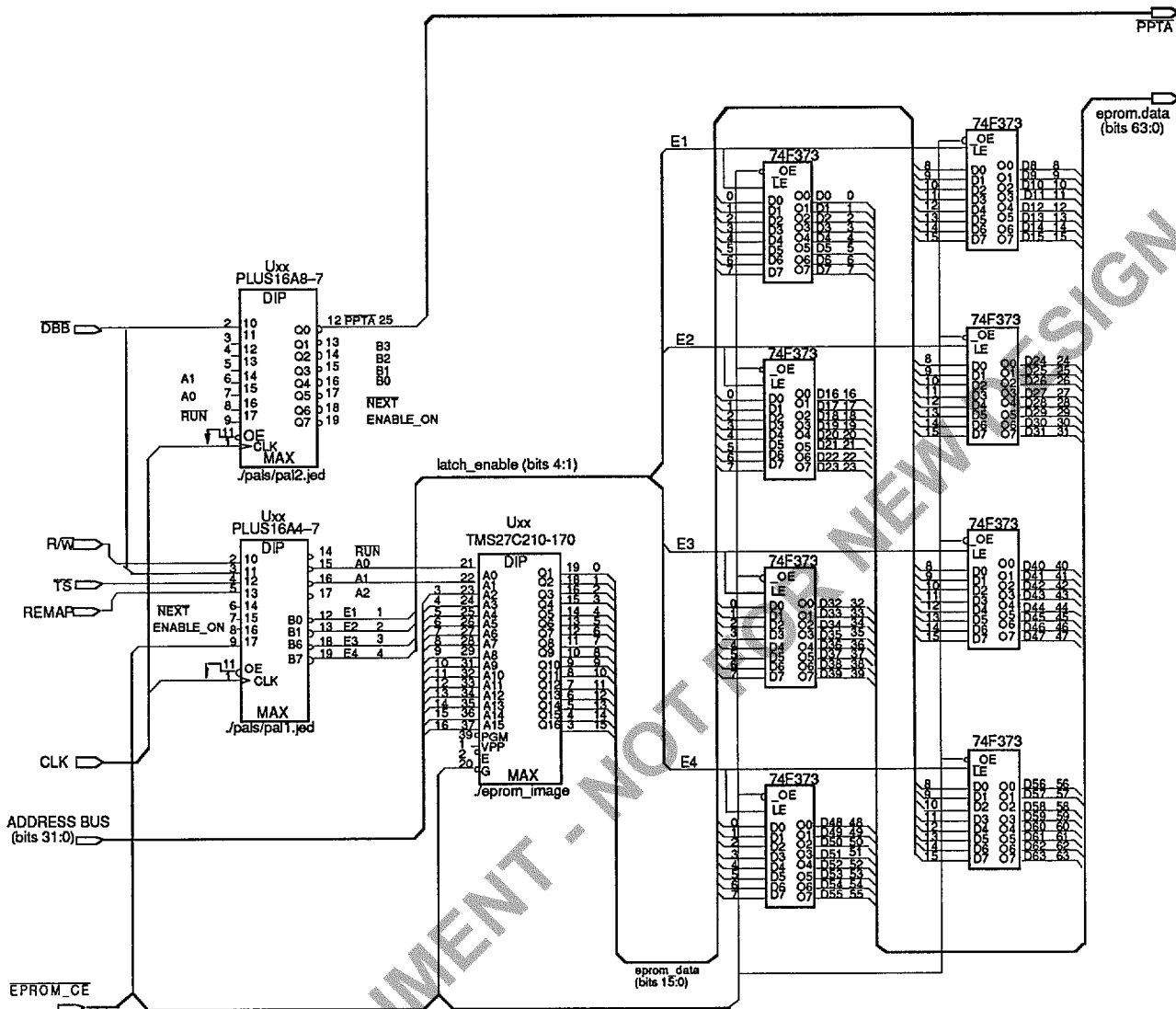


Figure 5. Final Schematic of EPROM Interface Design

causing the CTLSM to transition from state S0 to S1. This leaves a 10 ns set-up time for the CTLSM PAL, which only requires a 7 ns set-up.

The R/\bar{W} signal (not shown in the timing diagram) is guaranteed to be valid at the same time as the address lines (ADDR). Since the R/\bar{W} signal appears 4 to 15 ns after clock 1, it cannot be used for the first state transition of CTLSM as the PAL set-up time might not be met. Therefore the R/\bar{W} signal is checked in clock 2. It is assumed that the external REMAP signal is valid and also can be checked in clock 2. If REMAP is negated and R/\bar{W} indicates a read, the CTLSM transitions to S2.

The $\overline{EPROM_CE}$ is decoded from the address and \overline{DBB} and thus becomes valid 17 ns into clock 2. (that is, 3 ns before the S1–S2 state transition). The $\overline{EPROM_CE}$ cannot be checked at the S1–S2 transition since the 3-ns margin before the active edge of clock 3 does not satisfy the PAL set-up time requirements. $\overline{EPROM_CE}$ is checked in clock 4, which provides a 23 ns set-up time margin before the rising edge of the clock. If $\overline{EPROM_CE}$ is asserted, a transition to S3 occurs. The transition to S3 indicates that an EPROM read data access has been initiated. Note that if the $\overline{EPROM_CE}$ is based solely upon the address, it will meet the set-up time requirements for the S1–S2 transition one clock cycle earlier. Then the $\overline{EPROM_CE}$ can be checked at the same time as REMAP and R/\bar{W} , and thus reduce the CTLSM by one state.

Upon reaching S3, the CTLSM sends the registered signal, \overline{RUN} . The registered \overline{RUN} signal is valid 7 ns into clock 4. The A1 and A0 outputs from the CTLSM are also valid 7 ns into clock 4. These bits are two of the state bits of CTLSM. Thus, 7 ns into clock 4, a valid address is available at the EPROMs address signals.

The \overline{RUN} signal is the trigger for CNTSM to transition to state S1. CNTSM detects this signal in clock 5 and switches from state S0 to S1. In state S1, CNTSM outputs the ENABLE_ON signal, which appears 7 ns into clock 5. The ENABLE_ON signal is received by the CTLSM, which generates an enable for one pair of 74F373 latches 7 ns later. This signal allows any data that appears on the EPROMs data lines to flow into the selected latch. The enable signals remain asserted from clock 5 through clock 12, while the CNTSM transitions from S1 to S8. During this time, CTLSM remains in S3.

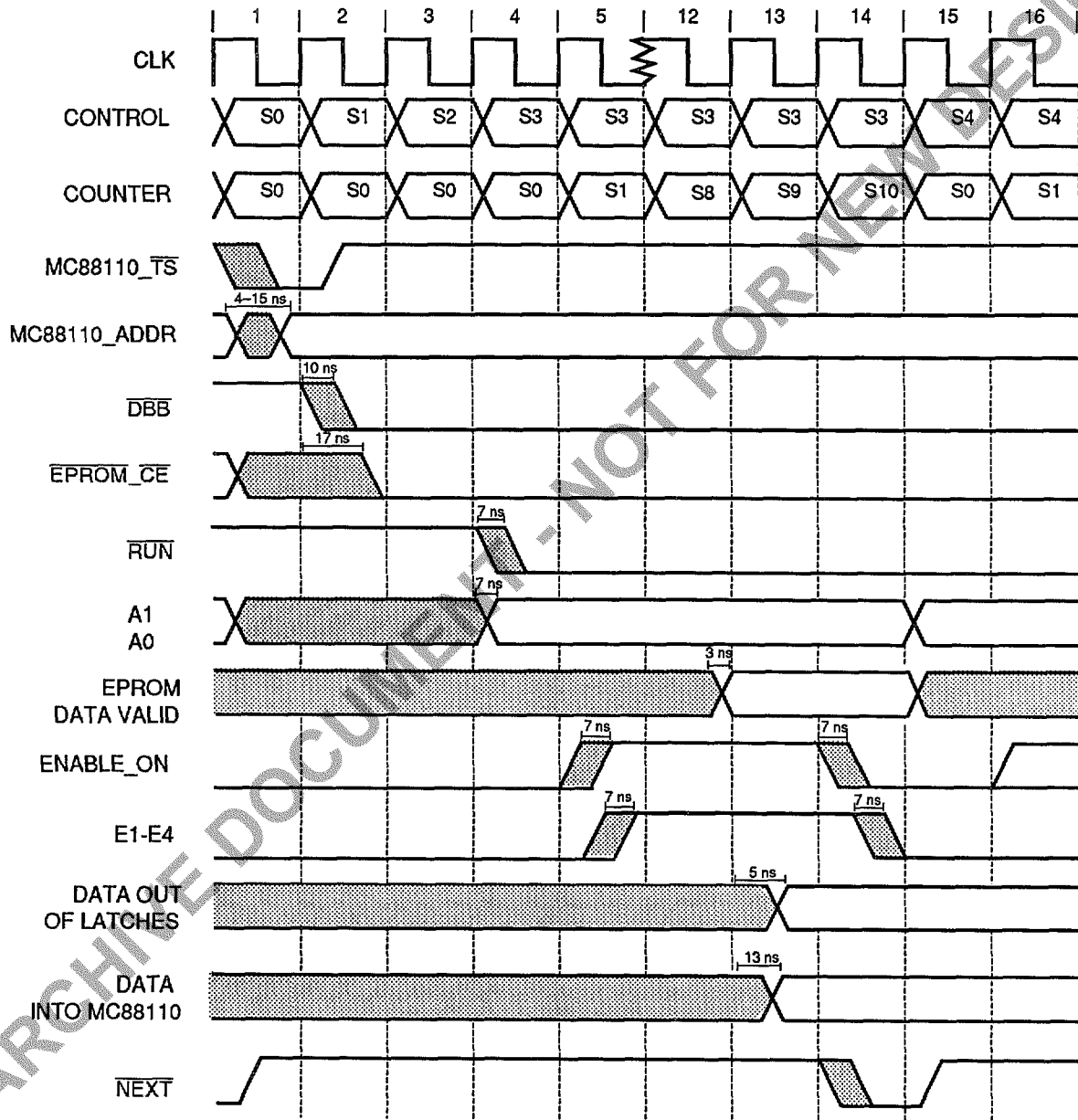


Figure 6. Timing Analysis Diagram (first EPROM Access)

When the CNTSM reaches S8 in clock 12, the EPROM data is valid. The EPROM timing specifications state that valid data is guaranteed to have appeared on the EPROM data signals 170 ns after a valid address is placed on its address signals. Since the address becomes valid 13 ns before clock 5, valid data is guaranteed to have appeared on the EPROM data signals 17 ns into clock 12. This data flows through the

selected pair of 74F373 latches 8 ns later (that is, 5 ns into clock 13). In clock 14, when the CNTSM reaches S10, the ENABLE_ON signal is negated. This forces the CTLSM to latch data into the first pair of latches (that is, negate E1) causing data to be captured in these latches.

The design presented in this application note, meets the 3 ns set-up time and the 1 ns hold time requirements of the 74F373 easily. Since the data is latched into the 74F373 5 ns into clock 13, this design provides a 15 ns set-up time. Additionally, EPROM data remains valid until clock 15, thus providing at least 6 ns of data hold time for the 74F373 latch pair.

If we assume a clock period variation of 1 ns, then EPROM data is valid 6 ns into clock 13. This data then flows through the latch in 8 ns. Thus 14 ns into clock 13, the latch is guaranteed to contain valid data. This leaves a margin of 5 ns before the latch turns off, latching the data. Thus even with a 1 ns clock period variation, this design meets the appropriate hold time for the 74F373.

During clock 14, CNTSM sends the $\overline{\text{NEXT}}$ signal, which informs the CTLSM that the first 16-bit word has been latched. The $\overline{\text{NEXT}}$ signal is a registered output that is asserted 7 ns into clock 14. CTLSM detects the asserted $\overline{\text{NEXT}}$ signal at clock 15 and transitions from S3 to S4. At the same time, CNTSM returns to state S0 from state S10. The CTLSM PAL keeps $\overline{\text{RUN}}$ asserted in S4. CNTSM checks this signal at clock 15 before it transitions back to S1, initiating the start of the second 16-bit word EPROM access.

The second and third EPROM accesses are identical to the first EPROM access except that the output data is latched onto two different pairs of latches. The second and third EPROM accesses occur between clock 15 and clock 37.

The timing for the fourth and last EPROM access is shown in Figure 7. The last EPROM access is initiated in clock 37. Once the last access starts, $\overline{\text{RUN}}$ is negated by the CTLSM to ensure that a fifth EPROM access does not occur. The fourth access is similar to the previous three accesses except that $\overline{\text{PPTA}}$ is generated to indicate that the 64-bit data transfer is near completion. The $\overline{\text{PPTA}}$ signal is a registered output from CNTSM, and is valid 7 ns into clock 45. Therefore, $\overline{\text{PPTA}}$ is valid 13 ns before clock 46 and therefore meets the setup time of 9 ns needed for the generation of $\overline{\text{PTA}}$ and $\overline{\text{TA}}$. It is assumed that external hardware generates the $\overline{\text{PTA}}$ and $\overline{\text{TA}}$ signals required by the MC88110. Since $\overline{\text{PPTA}}$ is valid at clock 45, $\overline{\text{PTA}}$ needs to be valid at clock 46 and $\overline{\text{TA}}$ needs to be generated in clock 47.

EPROM data becomes valid 17 ns into clock 45, and flows into the last 74F373 latch pair 5 ns into clock 46. It takes 8 ns for this data to flow through the transceivers to the MC88110 data signals. The MC88110 receives the last 16 bits of the 64-bit double word 13 ns into clock 46. Thus the data is valid 13 ns into clock 46. This data remains valid at the MC88110's data signals until clock 48.

Assuming a clock period variation of 1 ns, EPROM data is valid 6 ns into clock 46; data will be presented at the 74F373 latch pair 14 ns into clock 46, and reaches the MC88110 8 ns later (that is, 3 ns into clock 48).

The timing specifications of the MC88110 require that data be valid at least 9 ns before the last clock of the data transfer (that is, clock 48). Thus in this design, even with a 19 ns clock period, the data is valid at the MC88110 data bus 16 ns before the end of the final clock. This provides a 7 ns timing margin.

The CNTSM asserts the $\overline{\text{NEXT}}$ signal during clock 47. This signals CTLSM to switch to S7 in clock 48. Upon detecting the $\overline{\text{TA}}$ signal at the start of clock 48, the MC88110 negates $\overline{\text{DBB}}$ 0 ns to 10 ns into clock 48 to indicate the end of the data transfer. The MC88110 then ceases driving an address causing the address bus to be placed in a high-Z state. This forces the $\overline{\text{EPROM_CE}}$ to negate and disable the outputs of all the 74F373 latch pairs, thus preventing them from driving data onto the MC88110's data bus. The CTLSM detects the negated $\overline{\text{DBB}}$ and resets to state S0 if a back-to-back access is not in progress.

If the MC88110 sends out back-to-back data requests, $\overline{\text{DBB}}$ still negates in clock 48 to indicate the end of the first 64-bit data transfer. However, $\overline{\text{TS}}$ also asserts in clock 48 to indicate the start of a new data transfer. A new address is placed on the address bus 4 to 15 ns into clock 48. Thus if CTLSM detects an asserted $\overline{\text{TS}}$ while in state S7 it transitions to state S1 in order to process the back-to-back data request. This case is illustrated in Figure 7.

If the MC88110 data request is for a burst transfer, the \overline{DBB} signal does not negate upon receipt of the first \overline{TA} signal, but remains asserted until four \overline{TA} signals are detected. Thus if CTLISM detects that \overline{DBB} is still asserted in clock 48, it transitions to state S2 to initiate the read of the next 64-bit word in the burst transfer.

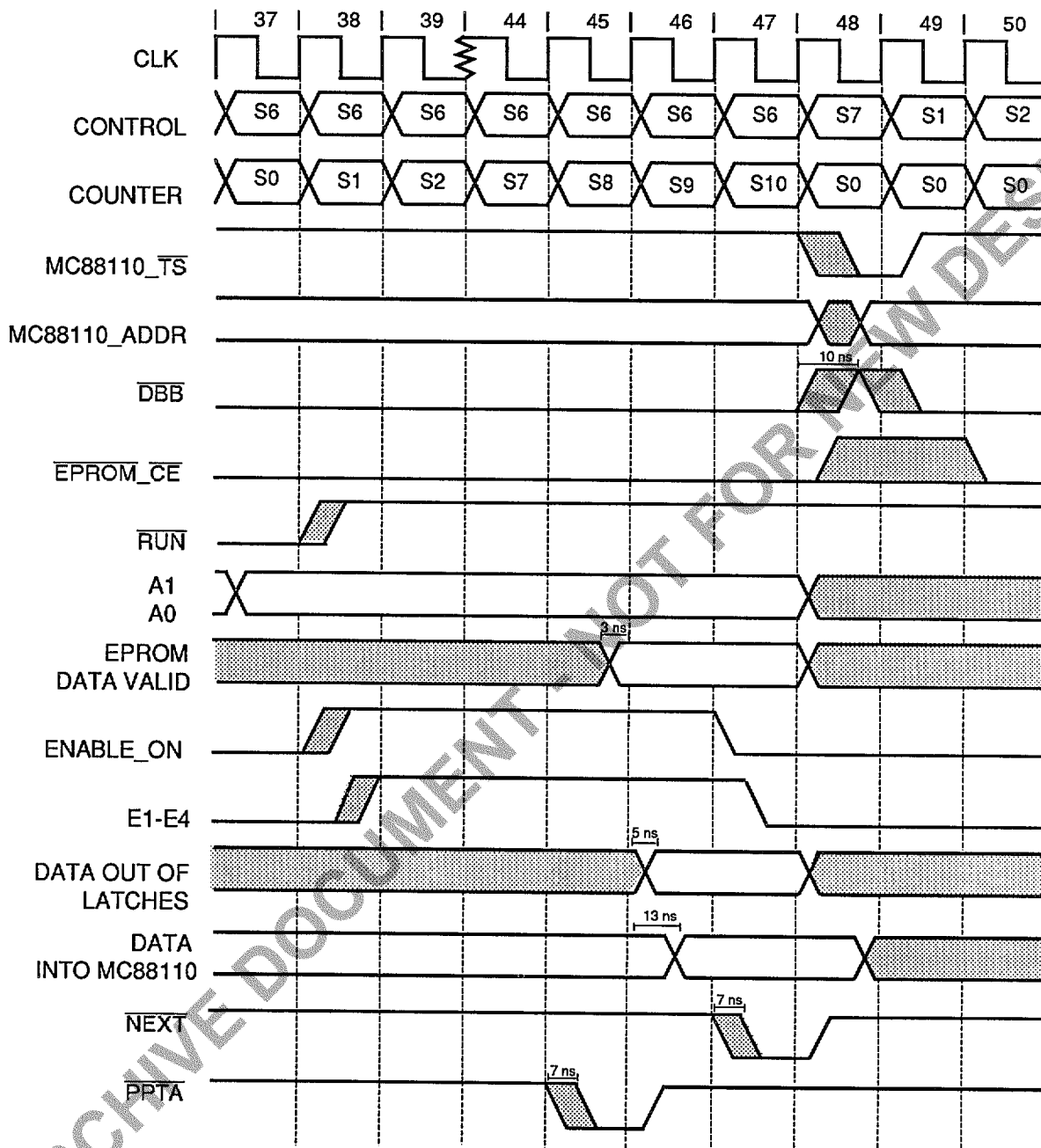


Figure 7. Timing Analysis Diagram (last EPROM Access)

Conclusion

This document describes a 64-bit to 16-bit EPROM interface for the MC88110. The design allows 64-bit word transfers to be completed in 48 cycles. The design presented in this application note satisfies all timing

constraints of a 50-MHz design with minimum but sufficient timing margins. The EPROM interface presented was designed to have sufficient timing margins to allow for board propagation delays.

ARCHIVE DOCUMENT - NOT FOR NEW DESIGN

State Diagram and PAL Equations for Control State Machine

```
module EPROM_Interface_Pal1;  
title '88110 Bus to 16-bit EPROM interface PAL1'
```

```
" This PAL implements State Machine number 1  
" This PAL also generates all the enable signals  
" to the 74F373 latches
```

```
pal1 device 'p16r4'; " 7 ns
```

```
" Inputs
```

```
CLK          pin 1;  
!TS          pin 4;  
ENABLE_ON   pin 8;  
!EPROM_CE   pin 9;  
REMAP       pin 5;  
!DBB        pin 3;  
R_W         pin 2;  
!NEXT       pin 7;  
!OE         pin 11;
```

```
" Outputs
```

```
!a2,!a1,!a0  pin 17,16,15;  
!RUN         pin 14;  
E1,E2,E3,E4 pin 12,13,18,19;
```

```
" Combinatorial
```

```
c,x = .c.,.x.;
```

```
" Set assignments for state machine
```

```
state_bits = [a2,a1,a0];
```

```
" Master State Values
```

```
S0 = (state_bits == ^b000); s0=^b000;  
S1 = (state_bits == ^b001); s1=^b001;  
S2 = (state_bits == ^b010); s2=^b010;  
S3 = (state_bits == ^b111); s3=^b111;  
S4 = (state_bits == ^b110); s4=^b110;  
S5 = (state_bits == ^b101); s5=^b101;  
S6 = (state_bits == ^b100); s6=^b100;  
S7 = (state_bits == ^b011); s7=^b011;
```

```
state_diagram state_bits
```

State s0:
if (TS) then s1
else s0;

State s1:
if (!REMAP & R_W) then s2
else s0;

State s2:
if (EPROM_CE) then s3
else s0;

State s3:
if (!IDBB) then s0 "IDBB negates if RESET
else if (!INEXT) then s3
else if (NEXT) then s4;

State s4:
if (!IDBB) then s0
else if (!INEXT) then s4
else if (NEXT) then s5;

State s5:
if (!IDBB) then s0
else if (!INEXT) then s5
else if (NEXT) then s6;

State s6:
if (!IDBB) then s0
else if (!INEXT) then s6
else if (NEXT) then s7;

State s7:
if (!TS & !IDBB) then s0
else if (TS) then s1
else if (IDBB) then s2;

equations

E1 = a1 & a0 & ENABLE_ON;
E2 = a1 & !a0 & ENABLE_ON;
E3 = !a1 & a0 & ENABLE_ON;
E4 = !a1 & !a0 & ENABLE_ON;

RUN := IS0 & !S1 & !S6 & !S7 & EPROM_CE;

test_vectors

([!IOE,CLK,REMAP,R_W,!EPROM_CE,ITS,!DBB,INEXT,ENABLE_ON] -> [state_bits,!RUN,E1,E2,E3,E4]);

" testing state graph

[0, c, x, x, x, 1, 1, x, x] -> [s0, 1, 0, 0, 0, 0];
[0, c, x, x, 0, 1, 0, x, x] -> [s0, 1, 0, 0, 0, 0]; "TS not asserted
[0, c, x, x, 0, 1, x, x] -> [s1, 1, 0, 0, 0, 0];
[0, c, 0, 0, x, 1, 0, x, x] -> [s0, 1, 0, 0, 0, 0]; "write and not remap
[0, c, 1, 1, x, x, 1, x, x] -> [s1, 1, 0, 0, 0, 0];
[0, c, 1, 1, x, 1, x, x, x] -> [s0, 1, 0, 0, 0, 0]; "read and remap
[0, c, 0, 1, x, 0, x, x, x] -> [s1, 1, 0, 0, 0, 0];
[0, c, 0, 1, x, 1, 0, x, x] -> [s2, 1, 0, 0, 0, 0];
[0, c, 0, 1, 1, 1, 0, x, 0] -> [s0, 1, 0, 0, 0, 0]; "eprom not selected
[0, c, x, x, x, 0, 1, x, 0] -> [s1, 1, 0, 0, 0, 0];
[0, c, 0, 1, 0, 1, 0, x, 0] -> [s2, 1, 0, 0, 0, 0];
[0, c, 0, 1, 0, 1, 0, 1, 0] -> [s3, 0, 0, 0, 0, 0]; "start asserts
[0, c, 0, 1, 0, 1, 0, 1, 1] -> [s3, 0, 1, 0, 0, 0]; "first latch enabled
[0, c, 0, 1, 0, 1, 0, 1, 1] -> [s3, 0, 1, 0, 0, 0];
[0, c, 0, 1, 0, 1, 0, 1, 1] -> [s3, 0, 1, 0, 0, 0]; "assume we are in s9 of pal2
[0, c, 0, 1, 0, 1, 0, 1, 1] -> [s3, 0, 1, 0, 0, 0]; "enable off at this transition
[0, 0, 0, 1, 0, 1, 0, 1, 0] -> [s3, 0, 0, 0, 0, 0]; "first latch disabled
[0, c, 0, 1, 0, 1, 0, 0, 0] -> [s4, 0, 0, 0, 0, 0]; "next state s4

[0, c, 0, 1, 0, 1, 0, 1, 1] -> [s4, 0, 0, 1, 0, 0]; "enable on latch2
[0, c, 0, 1, 0, 1, 0, 1, 1] -> [s4, 0, 0, 1, 0, 0]; "next is 0
[0, c, 0, 1, 0, 1, 0, 1, 0] -> [s4, 0, 0, 0, 0, 0]; "enable off latch2
[0, c, 0, 1, 0, 1, 0, 0, 0] -> [s5, 0, 0, 0, 0, 0]; "next state s5

[0, c, 0, 1, 0, 1, 0, 1, 1] -> [s5, 0, 0, 0, 1, 0]; "enable on latch3
[0, c, 0, 1, 0, 1, 0, 1, 1] -> [s5, 0, 0, 0, 1, 0]; "next is 0
[0, 0, 0, 1, 0, 1, 0, 1, 0] -> [s5, 0, 0, 0, 0, 0]; "enable off latch3
[0, c, 0, 1, 0, 1, 0, 0, 0] -> [s6, 0, 0, 0, 0, 0]; "next state s6

[0, c, 0, 1, 0, 1, 0, 1, 1] -> [s6, 1, 0, 0, 0, 1]; "enable on latch4
[0, c, 0, 1, 0, 1, 0, 1, 1] -> [s6, 1, 0, 0, 0, 1]; "next is 0
[0, c, 0, 1, 0, 1, 0, 1, 0] -> [s6, 1, 0, 0, 0, 0]; "enable off latch4
[0, c, 0, 1, 0, 1, 0, 0, 0] -> [s7, 1, 0, 0, 0, 0]; "next state s7
[0, c, 0, 1, 0, 1, 0, 1, 0] -> [s2, 1, 0, 0, 0, 0]; "burst access
"eprom read again
[0, c, 0, 1, 0, 1, 0, 1, 0] -> [s3, 0, 0, 0, 0, 0]; "back to s3

[0, c, 0, 1, 0, 1, 0, 1, 1] -> [s3, 0, 1, 0, 0, 0]; "first latch enabled
[0, 0, 0, 1, 0, 1, 0, 1, 0] -> [s3, 0, 0, 0, 0, 0]; "first latch disabled
[0, c, 0, 1, 0, 1, 0, 0, 0] -> [s4, 0, 0, 0, 0, 0]; "next state s4

[0, c, 0, 1, 0, 1, 0, 1, 1] -> [s4, 0, 0, 1, 0, 0]; "enable on latch2
[0, 0, 0, 1, 0, 1, 0, 1, 0] -> [s4, 0, 0, 0, 0, 0]; "enable off latch2
[0, c, 0, 1, 0, 1, 0, 0, 0] -> [s5, 0, 0, 0, 0, 0]; "next state s5

[0, c, 0, 1, 0, 1, 0, 1, 1] -> [s5, 0, 0, 0, 1, 0]; "enable on latch3
[0, 0, 0, 1, 0, 1, 0, 1, 0] -> [s5, 0, 0, 0, 0, 0]; "enable off latch3
[0, c, 0, 1, 0, 1, 0, 0, 0] -> [s6, 0, 0, 0, 0, 0]; "next state s6

[0, c, 0, 1, 0, 1, 0, 1, 1] -> [s6, 1, 0, 0, 0, 1]; "enable on latch4
[0, 0, 0, 1, 0, 1, 0, 1, 0] -> [s6, 1, 0, 0, 0, 0]; "enable off latch4
[0, c, 0, 1, 0, 1, 0, 0, 0] -> [s7, 1, 0, 0, 0, 0]; "next state s7

"eprom read over

[0, c, 0, 1, 0, 0, 1, 0, 0] -> [s1, 1, 0, 0, 0, 0]; "back to back access

[0, c, 0, 1, 0, 1, 0, 0, 0] -> [s2, 1, 0, 0, 0, 0];

"eprom read start

[0, c, 0, 1, 0, 1, 0, 1, 0] -> [s3, 0, 0, 0, 0, 0]; "back to s3

[0, c, 0, 1, 0, 1, 0, 0, 0] -> [s4, 0, 0, 0, 0, 0]; "next state s4

[0, c, 0, 1, 0, 1, 0, 0, 0] -> [s5, 0, 0, 0, 0, 0]; "next state s5

[0, c, 0, 1, 0, 1, 0, 0, 0] -> [s6, 0, 0, 0, 0, 0]; "next state s6

[0, c, 0, 1, 0, 1, 0, 0, 0] -> [s7, 1, 0, 0, 0, 0]; "next state s7

"eprom read over

[0, c, 0, 1, 0, 1, 1, 0, 0] -> [s0, 1, 0, 0, 0, 0]; "no TS and no DBB

" Next round of testing is with x's and only the inputs that cause transitions

"(!IOE,CLK,REMAP,R_W,IEPROM_CE,!TS,!DBB,NEXT,ENABLE_ON] -> [state_bits,!RUN,E1,E2,E3,E4]);

[0, c, x, x, x, 0, x, x, x] -> [s1, 1, 0, 0, 0, 0];

[0, c, 0, 1, x, x, x, x, x] -> [s2, 1, 0, 0, 0, 0];

[0, c, x, x, 0, x, x, x, x] -> [s3, 0, 0, 0, 0, 0];

[0, c, x, x, x, x, x, 1, x] -> [s3, 0, 0, 0, 0, 0];

[0, c, x, x, x, x, x, 0, x] -> [s4, 0, 0, 0, 0, 0];

[0, c, x, x, x, x, x, 1, x] -> [s4, 0, 0, 0, 0, 0];

[0, c, x, x, x, x, x, 0, x] -> [s5, 0, 0, 0, 0, 0];

[0, c, x, x, x, x, x, 1, x] -> [s5, 0, 0, 0, 0, 0];

[0, c, x, x, x, x, x, 0, x] -> [s6, 0, 0, 0, 0, 0];

[0, c, x, x, x, x, x, 1, x] -> [s6, 1, 0, 0, 0, 0];

[0, c, x, x, x, x, x, 0, x] -> [s7, 1, 0, 0, 0, 0];

```

[0, c, x, x, x, 1, 0, x, x] -> [s2, 1, 0, 0, 0, 0]; "burst read; TS negated
[0, c, x, x, 0, x, x, x, x] -> [s3, 0, 0, 0, 0, 0];
[0, c, x, x, x, x, x, 0, 0] -> [s4, 0, 0, 0, 0, 0];
[0, c, x, x, x, x, x, 0, 0] -> [s5, 0, 0, 0, 0, 0];
[0, c, x, x, x, x, x, 0, 0] -> [s6, 0, 0, 0, 0, 0];
[0, c, x, x, x, x, x, 0, 0] -> [s7, 1, 0, 0, 0, 0];
[0, c, x, x, x, 0, x, x, x] -> [s1, 1, 0, 0, 0, 0]; "new TS
[0, c, 0, 1, x, x, x, x, x] -> [s2, 1, 0, 0, 0, 0];
[0, c, x, x, x, x, x, 0, x] -> [s3, 0, 0, 0, 0, 0];
[0, c, x, x, x, x, x, 0, x] -> [s4, 0, 0, 0, 0, 0];
[0, c, x, x, x, x, x, 0, x] -> [s5, 0, 0, 0, 0, 0];
[0, c, x, x, x, x, x, 0, x] -> [s6, 0, 0, 0, 0, 0];
[0, c, x, x, x, x, x, 0, x] -> [s7, 1, 0, 0, 0, 0];
[0, c, x, x, x, 1, 1, x, x] -> [s0, 1, 0, 0, 0, 0]; "no TS no DBB

```

```
end EPROM_Interface_Pal1;
```

88110 Bus to 16-bit EPROM interface PAL1
Equations for Module EPROM_Interface_Pal1

Device pal

- Reduced Equations:

```

~a2 := !(~EPROM_CE & ~a0 & !~a1 & ~a2
# !~DBB & ~NEXT & !~a2
# !~DBB & !~a0 & !~a2
# !~DBB & !~a1 & !~a2);

```

```

~a1 := !(~EPROM_CE & ~a0 & !~a1 & ~a2
# !~DBB & ~TS & !~a0 & !~a1
# !~DBB & !~NEXT & ~a0 & ~a1 & !~a2
# !REMAP & R_W & !~a0 & ~a1 & ~a2
# !~DBB & !~a0 & !~a1 & !~a2
# !~DBB & ~NEXT & !~a1 & !~a2);

```

```

~a0 := !(~EPROM_CE & ~a0 & !~a1 & ~a2
# !~TS & !~a0 & !~a1 & ~a2
# !~TS & ~a0 & ~a1 & ~a2
# !~DBB & !~NEXT & ~a0 & !~a2
# !~DBB & ~NEXT & !~a0 & !~a2);

```

```
E1 = !(ENABLE_ON # ~a0 # ~a1);
```

E2 = !(ENABLE_ON # ~a1 # !~a0);

E3 = !(ENABLE_ON # ~a0 # !~a1);

E4 = !(ENABLE_ON # !~a0 # !~a1);

~RUN := !(~EPROM_CE & ~a0 & !~a1 # !~EPROM_CE & !~a0 & !~a2);

State Diagram & PAL Equations for Counter State Machine

```
module EPROM_Interface_Pal2;  
title '88110 Bus to 16-bit EPROM interface PAL2'
```

```
" This PAL implements State Machine number 2
```

```
pal2 device 'p16r8';          "7 ns
```

```
" Inputs
```

```
CLK                pin 1;  
a0                 pin 7;  
a1                 pin 8;  
!RUN               pin 9;  
!DBB               pin 2;  
!OE                pin 11;
```

```
" Outputs
```

```
!b3,!b2,!b1,!b0   pin 13,14,15,16;  
ENABLE_ON          pin 19;  
!PPTA              pin 12;  
!NEXT              pin 18;
```

```
" Set assignments for state machine
```

```
state_bits = [b3,b2,b1,b0];
```

```
" Combinatorial
```

```
c,x = .c.,.x.;
```

```
" Master State Values
```

```
S0 = (state_bits == ^b0000); s0=^b0000;  
S1 = (state_bits == ^b0001); s1=^b0001;  
S2 = (state_bits == ^b0010); s2=^b0010;  
S3 = (state_bits == ^b0011); s3=^b0011;
```

```
S4 = (state_bits == ^b0100); s4=^b0100;  
S5 = (state_bits == ^b0101); s5=^b0101;  
S6 = (state_bits == ^b0110); s6=^b0110;  
S7 = (state_bits == ^b0111); s7=^b0111;  
S8 = (state_bits == ^b1000); s8=^b1000;  
S9 = (state_bits == ^b1001); s9=^b1001;  
S10 = (state_bits == ^b1010); s10=^b1010;
```

" States that are not used

```
s11=^b1011;  
s12=^b1100;  
s13=^b1101;  
s14=^b1110;  
s15=^b1111;
```

state_diagram state_bits

State s0:

```
if (DBB & RUN) then s1  
with ENABLE_ON := 1;  
else s0 with ENABLE_ON :=0;
```

State s1:

```
if (DBB) then s2 with ENABLE_ON := 1;  
else s0 with ENABLE_ON :=0;
```

State s2:

```
if (DBB) then s3 with ENABLE_ON := 1;  
else s0 with ENABLE_ON :=0;
```

State s3:

```
if (DBB) then s4 with ENABLE_ON :=1;  
else s0 with ENABLE_ON :=0;
```

State s4:

```
if (DBB) then s5 with ENABLE_ON := 1;  
else s0 with ENABLE_ON :=0;
```

State s5:

```
if (DBB) then s6 with ENABLE_ON := 1;
```

else s0 with ENABLE_ON :=0;

State s6:

if (DBB) then s7 with ENABLE_ON := 1;
else s0 with ENABLE_ON :=0;

State s7:

if (DBB) then s8 with ENABLE_ON := 1;
else s0 with ENABLE_ON :=0;

State s8:

if (DBB) then s9 with ENABLE_ON := 1;
else s0 with ENABLE_ON :=0;

State s9:

if (DBB) then s10 with ENABLE_ON := 0;
else s0 with ENABLE_ON :=0;

State s10:

goto s0 with ENABLE_ON :=0;

State s11:

goto s0 with ENABLE_ON := 0;

State s12:

goto s0 with ENABLE_ON := 0;

State s13:

goto s0 with ENABLE_ON := 0;

State s14:

goto s0 with ENABLE_ON := 0;

State s15:

goto s0 with ENABLE_ON := 0;

equations

PPTA := a0 & a1 & S7;

NEXT := S9;

test_vectors

([!IOE, CLK, !DBB,!RUN,a1,a0] -> [state_bits,ENABLE_ON,!NEXT,!PPTA]);
" testing what functional inputs will be

```

[0,c,0,0,1,0] -> [s8, 1,1,1];
[0,c,0,0,1,0] -> [s9, 1,1,1];
[0,c,0,0,1,0] -> [s10,0,0,1];
[0,c,0,0,1,0] -> [s0, 0,1,1];
" In state s6 of PAL1 state diagram
[0,c,0,0,1,1] -> [s1, 1,1,1];
[0,c,0,0,1,1] -> [s2, 1,1,1];
[0,c,0,0,1,1] -> [s3, 1,1,1];
[0,c,0,0,1,1] -> [s4, 1,1,1];
[0,c,0,0,1,1] -> [s5, 1,1,1];
[0,c,0,0,1,1] -> [s6, 1,1,1];
[0,c,0,0,1,1] -> [s7, 1,1,1];
[0,c,0,0,1,1] -> [s8, 1,1,0];
[0,c,0,0,1,1] -> [s9, 1,1,1];
[0,c,0,0,1,1] -> [s10,0,0,1];
[0,c,0,0,1,1] -> [s0, 0,1,1];

```

```
end EPROM_Interface_Pal2;
```

```
Device pal2
```

- Reduced Equations:

```

ENABLE_ON := !(~b2 & !~b3
# !~b1 & !~b3
# !~b0 & !~b3
# ~DBB & !~b3
# ~DBB & !~b2
# ~DBB & !~b1
# ~DBB & !~b0
# ~RUN & ~b0 & ~b1 & ~b2 & ~b3);

```

```
~b3 := !(~DBB & ~b1 & ~b2 & !~b3 # !~DBB & !~b0 & !~b1 & !~b2 & ~b3);
```

```

~b2 := !(~DBB & ~b0 & !~b2 & ~b3
# !~DBB & ~b1 & !~b2 & ~b3
# !~DBB & !~b0 & !~b1 & ~b2 & ~b3);

```

```

~b1 := !(~DBB & ~b0 & !~b1 & ~b3
# !~DBB & !~b0 & ~b1 & ~b2
# !~DBB & !~b0 & ~b1 & ~b3);

```

~b0 := !(~DBB & ~b0 & !~b2 & ~b3
!~DBB & ~b0 & !~b1 & ~b3
!~DBB & ~b0 & ~b1 & ~b2 & !~b3
!~RUN & ~b0 & ~b1 & ~b2 & ~b3);

~PPTA := !(a0 & a1 & !~b0 & !~b1 & !~b2 & ~b3);

~NEXT := !(~b0 & ~b1 & ~b2 & !~b3);

ARCHIVE DOCUMENT - NOT FOR NEW DESIGN

[0,c,1,1,x,x] -> [s0, 0,1,1];
[0,c,0,1,x,x] -> [s0, 0,1,1];
[0,c,0,0,x,x] -> [s1, 1,1,1];
[0,c,0,x,x,x] -> [s2, 1,1,1];
[0,c,0,x,x,x] -> [s3, 1,1,1];
[0,c,0,x,x,x] -> [s4, 1,1,1];
[0,c,0,x,x,x] -> [s5, 1,1,1];
[0,c,0,x,x,x] -> [s6, 1,1,1];
[0,c,0,x,x,x] -> [s7, 1,1,1];
[0,c,0,x,1,1] -> [s8, 1,1,0];
[0,c,0,x,x,x] -> [s9, 1,1,1];
[0,c,0,x,x,x] -> [s10,0,0,1];
[0,c,0,x,x,x] -> [s0, 0,1,1];

" testing what actual inputs will be

" In state s3 of PAL1 state diagram

[0,c,0,1,0,0] -> [s0, 0,1,1];
[0,c,0,0,0,0] -> [s1, 1,1,1];
[0,c,0,0,0,0] -> [s2, 1,1,1];
[0,c,0,0,0,0] -> [s3, 1,1,1];
[0,c,0,0,0,0] -> [s4, 1,1,1];
[0,c,0,0,0,0] -> [s5, 1,1,1];
[0,c,0,0,0,0] -> [s6, 1,1,1];
[0,c,0,0,0,0] -> [s7, 1,1,1];
[0,c,0,0,0,0] -> [s8, 1,1,1];
[0,c,0,0,0,0] -> [s9, 1,1,1];
[0,c,0,0,0,0] -> [s10,0,0,1];
[0,c,0,0,0,0] -> [s0, 0,1,1];


" In state s4 of PAL1 state diagram

[0,c,0,0,0,1] -> [s1, 1,1,1];
[0,c,0,0,0,1] -> [s2, 1,1,1];
[0,c,0,0,0,1] -> [s3, 1,1,1];
[0,c,0,0,0,1] -> [s4, 1,1,1];
[0,c,0,0,0,1] -> [s5, 1,1,1];
[0,c,0,0,0,1] -> [s6, 1,1,1];
[0,c,0,0,0,1] -> [s7, 1,1,1];
[0,c,0,0,0,1] -> [s8, 1,1,1];
[0,c,0,0,0,1] -> [s9, 1,1,1];
[0,c,0,0,0,1] -> [s10,0,0,1];
[0,c,0,0,0,1] -> [s0, 0,1,1];

" In state s5 of PAL1 state diagram

[0,c,0,0,1,0] -> [s1, 1,1,1];
[0,c,0,0,1,0] -> [s2, 1,1,1];
[0,c,0,0,1,0] -> [s3, 1,1,1];
[0,c,0,0,1,0] -> [s4, 1,1,1];
[0,c,0,0,1,0] -> [s5, 1,1,1];
[0,c,0,0,1,0] -> [s6, 1,1,1];
[0,c,0,0,1,0] -> [s7, 1,1,1];

PRELIMINARY DOCUMENT - NOT FOR NEW DESIGN

Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters can and do vary in different applications. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part. Motorola and  are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

Literature Distribution Centers:

USA: Motorola Literature Distribution; P.O. Box 20912; Phoenix, Arizona 85036.

EUROPE: Motorola Ltd.; European Literature Centre; 88 Tanners Drive, Blakelands, Milton Keynes, MK14 5BP, England.

JAPAN: Nippon Motorola Ltd.; 4-32-1, Nishi-Gotanda, Shinagawa-ku, Tokyo 141 Japan.

ASIA-PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbour Center, No. 2 Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong.

