INTRODUCTION

The PLUS153-10 is available in a 20-pin DIP or 20-pin PLCC package. The PLUS173-10 is available in a 24-pin DIP or 28-pin PLCC package. Both parts have t_{PD} no greater than 10ns.

Both parts provide 32 wide input product terms, whose outputs may be tied to the inputs of the sum terms (OR gates) below. There are no restrictions on this interconnect — any or all product terms may feed any or all sum terms. Thus each OR term can accept from 1 to 32 inputs without leaving the chip for a signal "wrap around". All ten outputs are bi-directional, so they may be traded off as inputs are used. Finally, each output may be polarity configured (exclusive-OR fused) and each is independently 3-Stateable from a separate product term (each) which is identical to the rest.

Although slightly slower (from pin to pin) than 7.5ns 20L8 structures, the following example demonstrates a simple case of how a 10ns PLA can be faster than a 7.5ns PAL.

Example 1: Glue Collection

This first example is an illustration of compressing glue logic. Figure 1 shows a piece of logic which performs one of two operations on two 8-bit numbers. These may come from different registers in a system, or be from two halves of a 16-bit bus. The goal is to perform the input operations (compare the bytes in one mode or multiplex one bit out in the other) in 10ns. Using MSI parts, this could have been done except there is no 16 to 1 MUX available in the 74F device series. There is a 74150 device available, but it has a propagation delay of 17ns. So this will not work. Figure 1 shows the solution using the 7.5ns PAL devices. Unfortunately, because architecture provides only seven product terms per sum term (16L8-7) multiple signal passes are required. This results in a solution needing over 20ns. It might be conjectured that a 15ns 22V10 could make it with 16 product terms on some outputs, but doing the MUX would only provide the output at "point 1" in 15ns. Additional time is needed to make the final out signal. A 10ns 22V10 could not make spec, with an additional 74F32 adding 4ns. Figure 3 shows the preferred solution — a single PLUS173 generating the final function in 10ns. Figures 4 and 5 show the pinout and SNAP equations for this solution.

Example 2: Cache Update Inhibit

Key to modern microsystem design has been simple, fast RISC processors with quick cache and single cycle high performance operation. Unfortunately even using the new cache control chips, exception handling results in clumsy designs. This may be one of the reasons simple, direct-mapped caches have also become popular. Exception handling is often resolving transactions which occur with data items that are non-cacheable. This occurs in a number of ways — first, EPROMs, I/O devices and special state registers are not cacheable items, so they will never be put into a cache memory. What happens when a non-cacheable item is referenced? The cache controller will miss and begin to update the cache. The transaction must be terminated before it overlays an I/O device onto the least recently used cache address.

The way to deal with the transaction overlay problem is straightforward — recognize all non-cacheable transactions and intercept them before the controller cleans house. How big of a problem is this? Figure 6 shows what might be an average engineering workstation. Each device (disk controller, LAN controller, keyboard, printer, etc.) usually has several internal registers each occupying a unique address. With two disks, a LAN, modem and printer, a system could instantly exceed 16 distinct I/O registers. It is best to assume a large number. Enter the PLA — the PLUS173 — for such a system. Each product term can be scattered all over memory if needed and decodes summed into a single output signal generating a composite inhibit. This process takes less than 10ns for up to 32 devices. Using a 20L8-7 requires trading off resolution (number of address bits resolved) and feeding through the chip multiple times, expanding to 13 devices in two passes (at 15ns for a 7.5ns device). Using a PLA keeps the RISC design very clean and fast.

Example 3: Interfacing Mixed Memory Types

Other sections of a microprocessor system can use the summation of a large number of decoded terms. For instance, the interrupt request, DMA request and the cycle extension WAIT line are contenders for a large number of decoded and summed inputs. Some are asserted low and some high so polarity control is vital. Some require a 3-State or open collector resistive pull-up, so the PLA enable fits well. These are situations where attention signals come into the processor.

It is not always necessary for the CPU to operate at full speed. Operating the CPU at a slower speed brings about a more economical and compact system. This is due to higher costs associated with fast memory and greater board area for wide memory configurations.

Some software routines where slower performance may be acceptable include power up initialization, diagnostic routines, or some exception routines. When speed is critical, an 8-bit bus is the most economical and compact because of readily available byte wide PROMs and RAMs. The 68030 is easily interfaced to 8-, 16- or 32-bit ports because it dynamically interprets the port size during each bus cycle. Figure 7 shows an example of interfacing both a slow 200ns 8-bit EPROM and a fast 35ns 32-bit RAM to a 68030. A PLUS173-10 was chosen for its high speed and large number of inputs and outputs. The EPROM occupies memory space 0–32K while the RAM occupies addresses 64–128K. Note that because not all of the upper memory address bits were decoded, the memory arrays will also appear at other addresses.

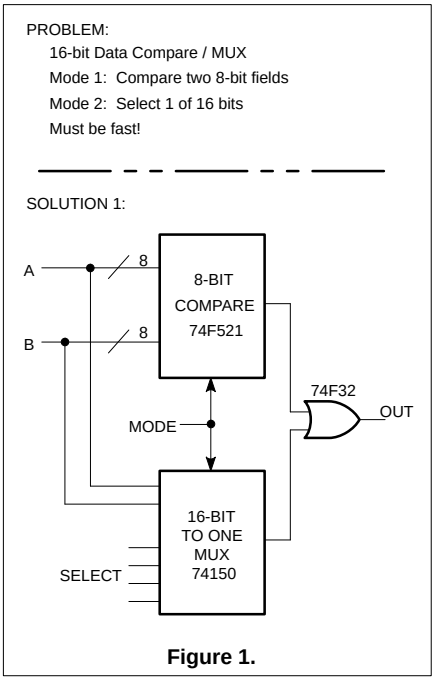
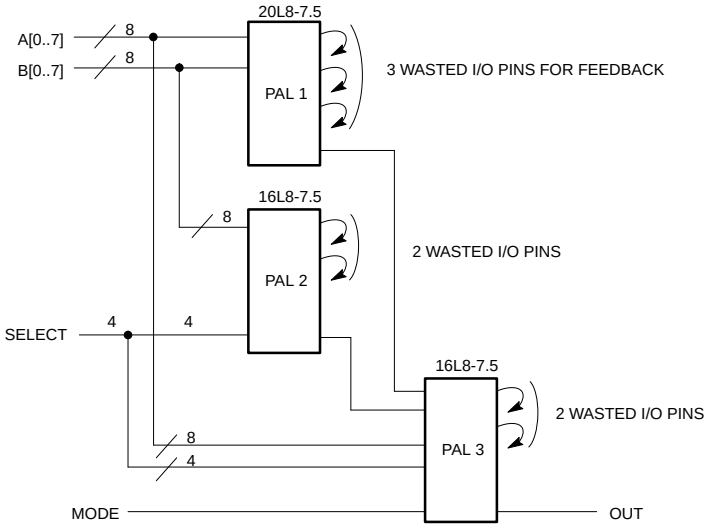


Figure 1.

Quick PLA

SOLUTION 2: MAKE IT OUT OF PALs



3) 7.5 PALs
1) 20L8-7.5
2) 16L8-7.5

COMPARE FUNCTION:

t_{PD} = 7.5 (1st PASS — #1)
= 7.5 (2nd PASS — #1)
= 7.5 (3rd PASS — #3)
= 22.5ns

MUX FUNCTION:

t_{PD} = 7.5 (1st PASS — #2 or 3)
= 7.5 (2nd PASS — #2 or 3)
= 7.5 (3rd PASS — #3)
= 22.5ns

GATE EFFICIENCY:

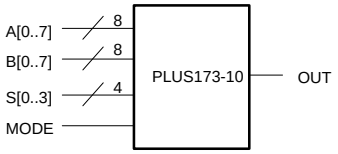
30 PALs
40 GATES USED
152 GATES WASTED

RESULT:

NOT FAST ENOUGH; SLOWER THAN 74F SOLUTION

Figure 2.

SOLUTION 3: MAKE IT OUT OF A PLUS173D



t_{PD} = 10ns
1 PACKAGE

COMPARE FUNCTION:
16 PRODUCT TERMS

MUX FUNCTION:
16 PRODUCT TERMS

INVERTERS:
INCLUDED

ORS:
INCLUDED

GATE EFFICIENCY:
1 PLA
32 GATES USED
NO GATES WASTED

RESULT:
THE BEST SOLUTION; ONE PACKAGE

Figure 3.

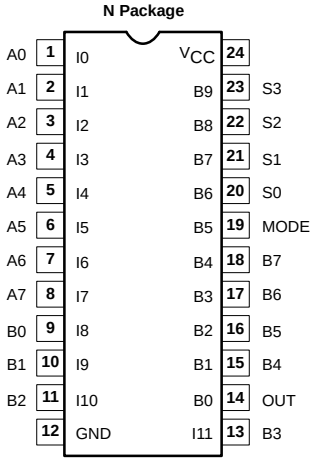


Figure 4. Comparator/MUX Pin Program

Quick PLA

```
@PINLIST
a[0..7]   i;
b[0..7]   i;
s[0..3]   i;
MODE  i;
OUT     o;

@GROUPS
sel       =   s[0..3];

@TRUTHTABLE
@LOGIC EQUATIONS
 comp    =   a0 * b0 /mode
         +   /a0 * b0 /mode
         +   a1 * b1 /mode
         +   /a1 * b1 /mode
         +   a2 * b2 /mode
         +   /a2 * b2 /mode
         +   a3 * b3 /mode
         +   /a3 * b3 /mode
         =   a4 * b4 /mode
         +   /a4 * b4 /mode
         +   a5 * b5 /mode
         +   /a5 * b5 /mode
         +   a6 * b6 /mode
         +   /a6 * b6 /mode
         +   a7 * b7 /mode
         +   /a7 * b7 /mode;

 mux     =   a0 * (sel == 0h) * mode
         +   a1 * (sel == 1h) *
 mode    +   a2 * (sel == 2h) * mode
         +   a3 * (sel == 3h) * mode
         +   a4 * (sel == 4h) * mode
         +   a5 * (sel == 5h) * mode
         +   a6 * (sel == 6h) * mode
         +   a7 * (sel == 7h) * mode
         =   b0 * (sel == 8h) * mode
         +   b1 * (sel == 9h) * mode
         +   b2 * (sel == Ah) * mode
         +   b3 * (sel == Bh) * mode
         +   b4 * (sel == Ch) * mode
         +   b5 * (sel == Dh) * mode
         +   b6 * (sel == Eh) * mode
         +   b7 * (sel == Fh) * mode;

 out = mux + comp;

@INPUT VECTORS
@OUTPUT VECTORS
@STATE VECTORS
@TRANSITIONS
```

Figure 5. SNAP Equation Listing

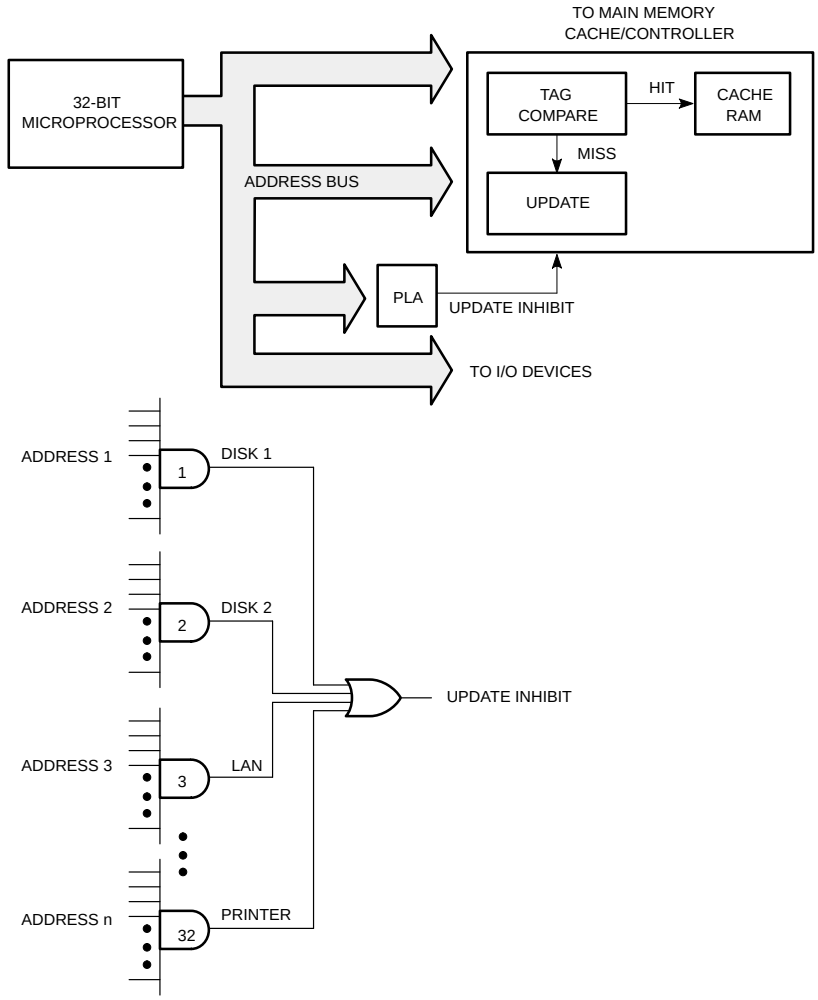


Figure 6. Cache Update Inhibit Using a PLA

Quick PLA

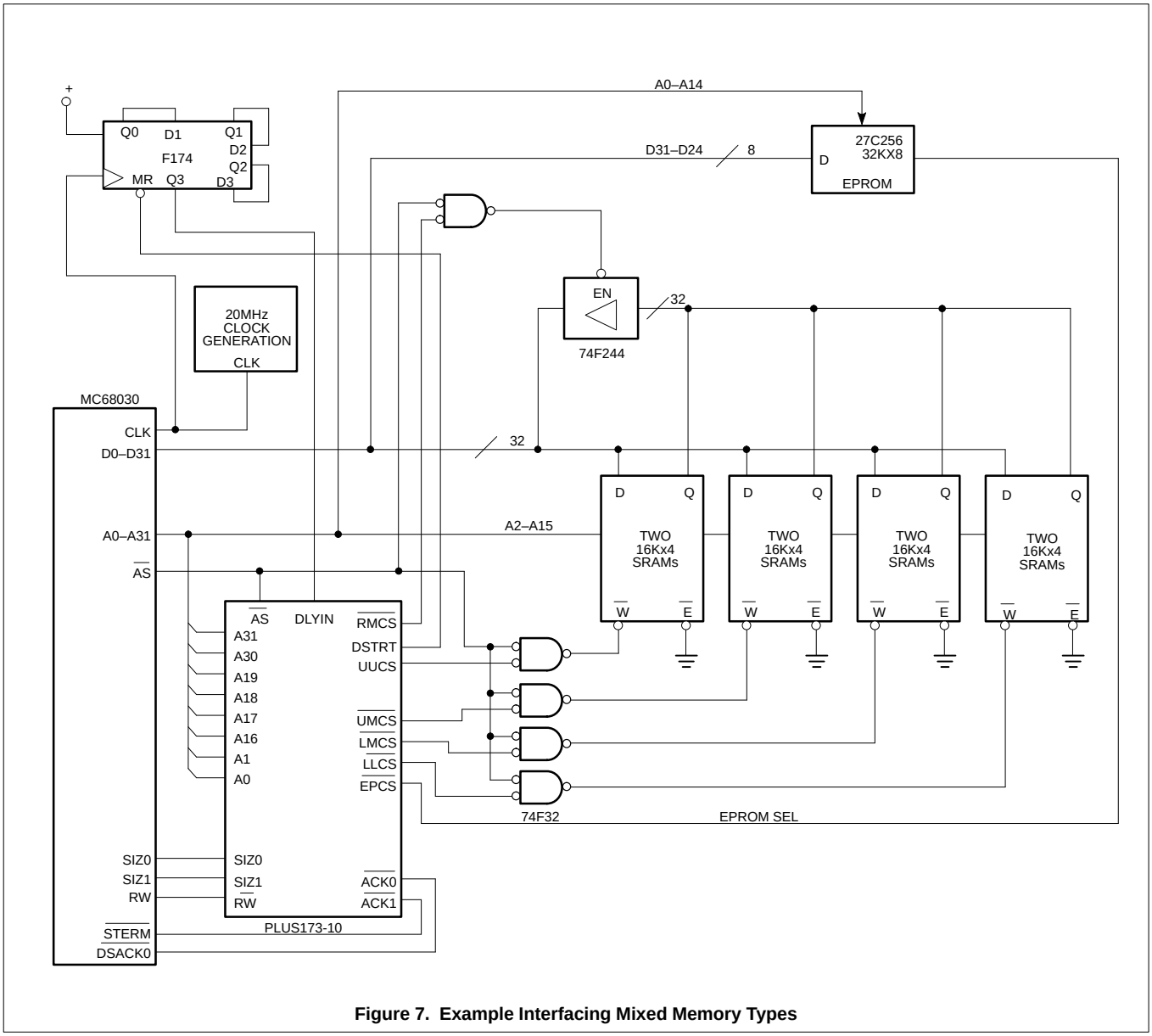


Figure 7. Example Interfacing Mixed Memory Types

Quick PLA

DECODER FOR INTERFACING SRAMa AND EPROMs TO AN MC68030. THIS DESIGN IS FOR A PLUS173 DEVICE

```
@PINLIST
dlyin     i;
nas       i;
a[31..30]    i;
a[19..16]    i;
a[1..0]   i;
siz0      i;
siz1      i;
rw        i;
nrmcs     o;
dstrt     o;
nuucs     o;
numcs     o;
nlmcs     o;
nllcs     o;
nepcs     o;
nack[1..0]   o
;
@LOGIC EQUATIONS
 "EPROM enable"
 nepcs = / ( /a31 * /a30 * /a19 * / a18 * /a17 * / a16 * / nas;

 "start shift register during EPROM access"
 dstrt = / ( /a31 * /a30 * /a19 * / a18 * /a17 * / a16 * / nas;

 "DSACKO after 4 clock cycles for EPROM access"
 nack0 = / (dlyin);

 "immediate STERM upon RAM access"
 nack1 = / ( /a31 * /a30 * /a19 * /a18 * /a17 * /a16);

 "Byte select signals for RAM writes"
 nuucs   =   / ( /a0 * /a1 *          /rw * a16 * /a17 * /18 * /a19 * /a30 * /a31);

 nuucs   =   / ( a0 * /a1 *           /rw * a16 * /a17 * /18 * /a19 * /a30 * /a31
           +   / a1 * /siz0 *         /rw * a16 * /a17 * /18 * /a19 * /a30 * /a31
           +   / a1 * /siz1 *         /rw * a16 * /a17 * /18 * /a19 * /a30 * /a31);

 nlmcs   =   / ( /a0 * /a1 *          /rw * a16 * /a17 * /18 * /a19 * /a30 * /a31
           +   / a1 * /siz0 * /siz1 *  /rw * a16 * /a17 * /18 * /a19 * /a30 * /a31
           +   / a1 * siz0 * /siz1 *   /rw * a16 * /a17 * /18 * /a19 * /a30 * /a31
           +   / a1 * a0 * /siz0 *     /rw * a16 * /a17 * /18 * /a19 * /a30 * /a31);

 nllcs   =   / ( a0 * a1 *            /rw * a16 * /a17 * /18 * /a19 * /a30 * /a31
           +    a0 * siz0 * siz1 *    /rw * a16 * /a17 * /18 * /a19 * /a30 * /a31
           +       / siz0 * /siz1 *   /rw * a16 * /a17 * /18 * /a19 * /a30 * /a31
           +   / a1 *         siz1 *  /rw * a16 * /a17 * /18 * /a19 * /a30 * /a31);

 nrmcs   =   / (                      /rw * a16 * /a17 * /18 * /a19 * /a30 * /a31);
```

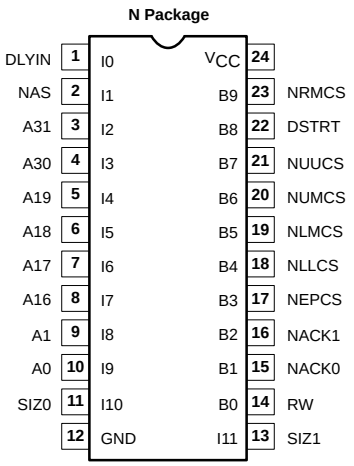


Figure 8. Equations for PLUS173 Shown in Figure 7