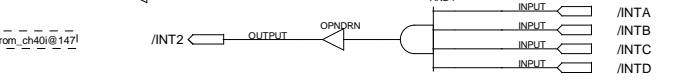
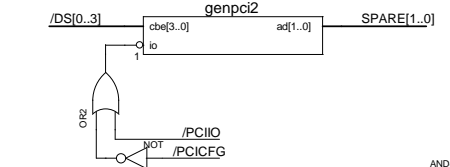
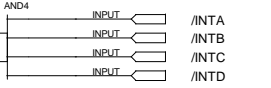
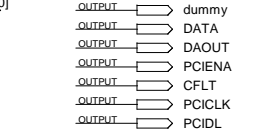
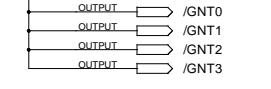
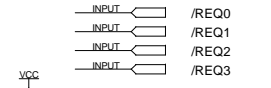
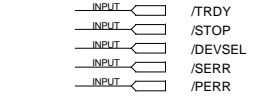
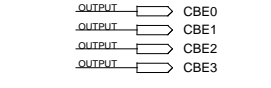
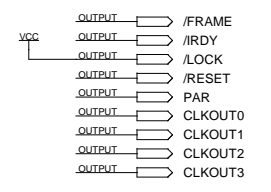
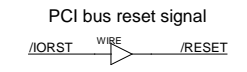
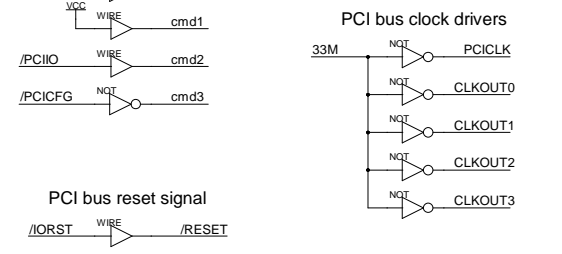
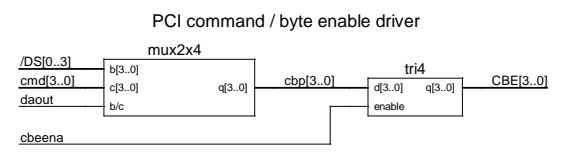
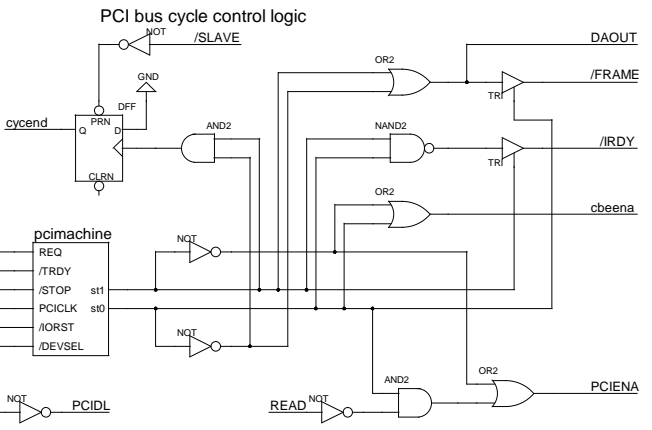
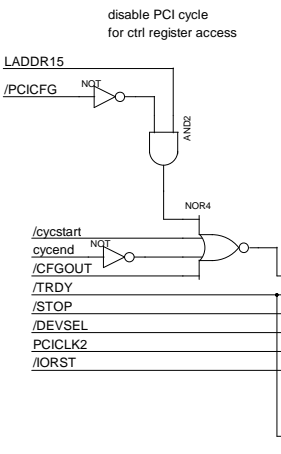
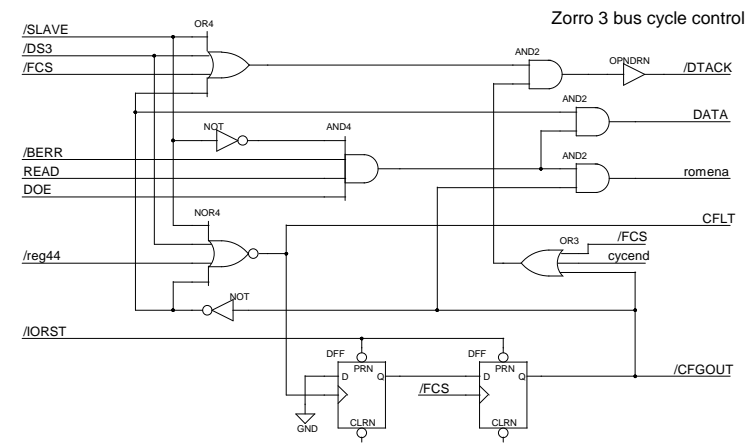
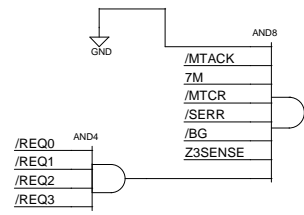
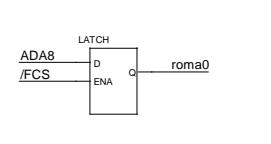
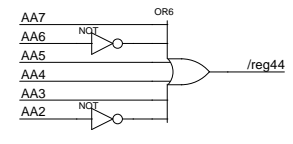


- prom\_ch40@80 I /FCS INPUT
- prom\_ch40@41 I /SLAVE INPUT
- prom\_ch40@111 I /DTACK OUTPUT
- prom\_ch40@43 I /CFGOUT OUTPUT
- prom\_ch40@102 I DOE INPUT
- prom\_ch40@149 I /MTCR INPUT
- prom\_ch40@130 I /MTACK INPUT
- prom\_ch40@131 I /BERR INPUT
- prom\_ch40@128 I /BR OUTPUT
- prom\_ch40@123 I /BG INPUT
- prom\_ch40@109 I READ INPUT
- prom\_ch40@105 I Z3SENSE INPUT
- prom\_ch40@101 I /IORST INPUT
- prom\_ch40@145 I /INT6 INPUT
- prom\_ch40@103 I 7M INPUT
- prom\_ch40@129 I /DS0 INPUT
- prom\_ch40@100 I /DS1 INPUT
- prom\_ch40@107 I /DS2 INPUT
- prom\_ch40@106 I /DS3 INPUT
- prom\_ch40@88 I 33M INPUT
- prom\_ch40@50 I /PCICFG INPUT
- prom\_ch40@51 I /PCIO INPUT
- prom\_ch40@49 I DATPAR0 INPUT
- prom\_ch40@48 I DATPAR1 INPUT
- prom\_ch40@139 I PCICLK2 INPUT
- prom\_ch40@62 I SPARE0 OUTPUT
- prom\_ch40@63 I SPARE1 OUTPUT
- prom\_ch40@64 I LADDR15 INPUT
- prom\_ch40@65 I SPARE3 INPUT
- prom\_ch40@135 I AA2 INPUT
- prom\_ch40@136 I AA3 INPUT
- prom\_ch40@137 I AA4 INPUT
- prom\_ch40@146 I AA5 INPUT
- prom\_ch40@144 I AA6 INPUT
- prom\_ch40@134 I AA7 INPUT
- prom\_ch40@132 I ADA8 INPUT
- prom\_ch40@122 I ADA31 OUTPUT
- prom\_ch40@121 I ADA30 OUTPUT
- prom\_ch40@110 I ADA29 OUTPUT
- prom\_ch40@108 I ADA28 OUTPUT



Zorro config register 44 decoder



- prom\_ch40@29 I
- prom\_ch40@28 I
- prom\_ch40@27 I
- prom\_ch40@25 I
- prom\_ch40@24 I
- prom\_ch40@30 I
- prom\_ch40@31 I
- prom\_ch40@32 I
- prom\_ch40@33 I

- prom\_ch40@23 I
- prom\_ch40@21 I
- prom\_ch40@20 I
- prom\_ch40@19 I

- prom\_ch40@18 I
- prom\_ch40@16 I
- prom\_ch40@15 I
- prom\_ch40@14 I
- prom\_ch40@13 I

- prom\_ch40@12 I
- prom\_ch40@11 I
- prom\_ch40@10 I
- prom\_ch40@160 I

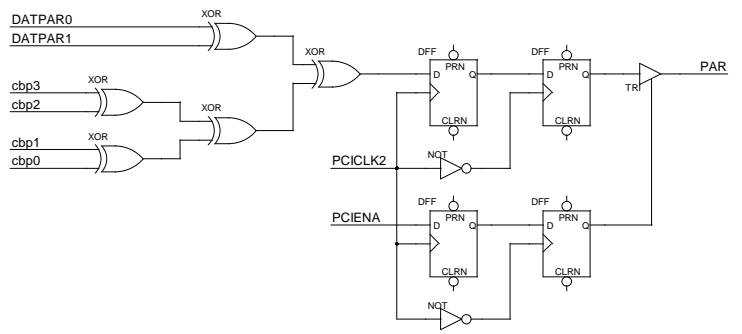
- prom\_ch40@159 I
- prom\_ch40@158 I
- prom\_ch40@153 I
- prom\_ch40@152 I

- prom\_ch40@57 I
- prom\_ch40@54 I
- prom\_ch40@52 I
- prom\_ch40@58 I
- prom\_ch40@56 I
- prom\_ch40@53 I

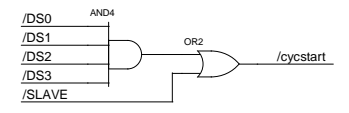
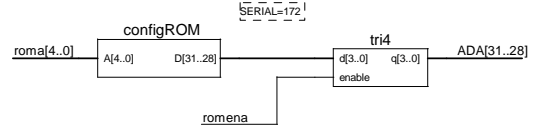
- prom\_ch40@90 I
- prom\_ch40@91 I
- prom\_ch40@92 I
- prom\_ch40@93 I

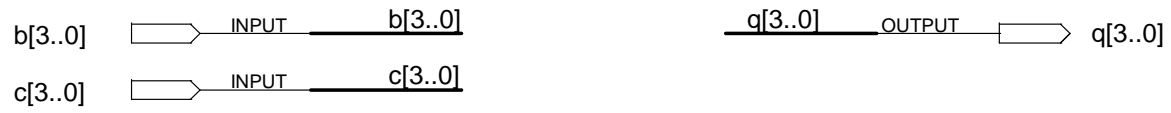
- roma4 WIRE
- roma3 WIRE
- roma2 WIRE
- roma1 WIRE

PCI parity circuit

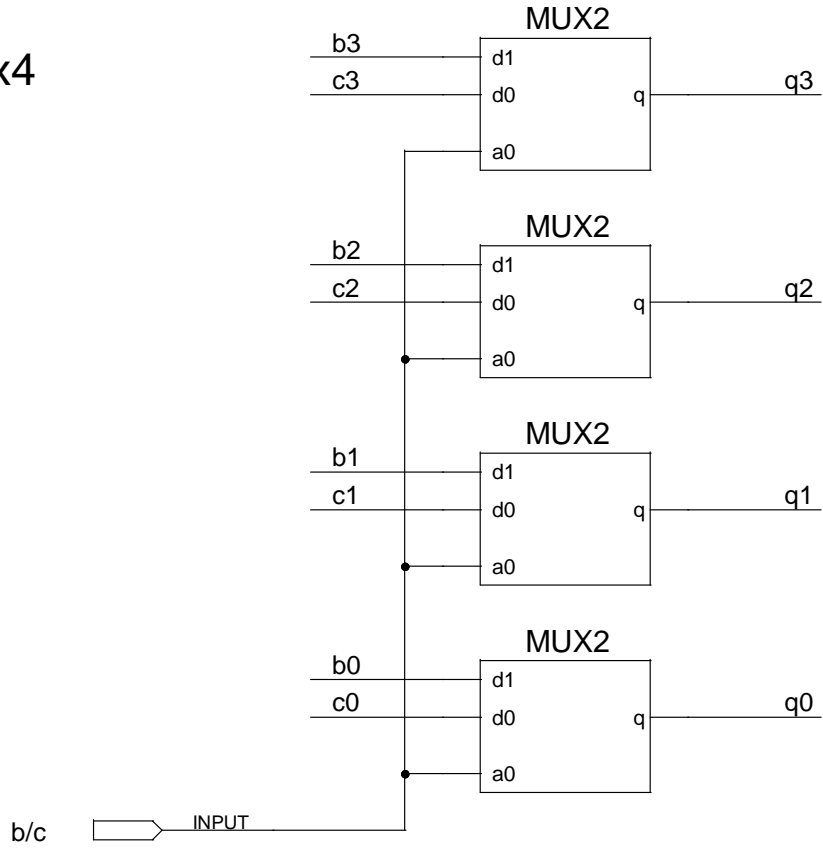


Zorro 3 configuration ROM



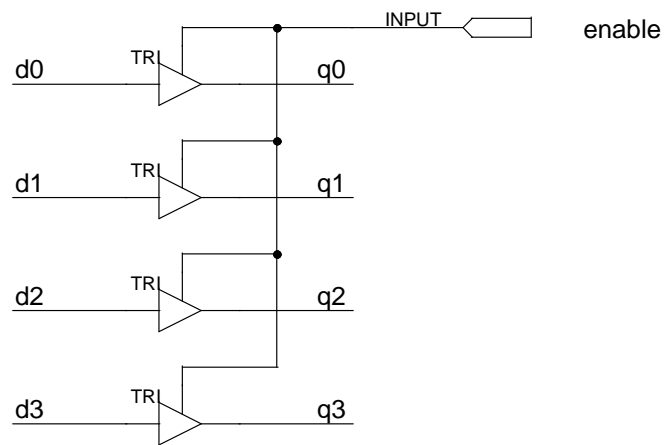


**mux2x4**





tri4



```
SUBDESIGN genpci2
```

```
(  
  cbe[3..0], io : INPUT;  
  ad[1..0] : OUTPUT;  
)
```

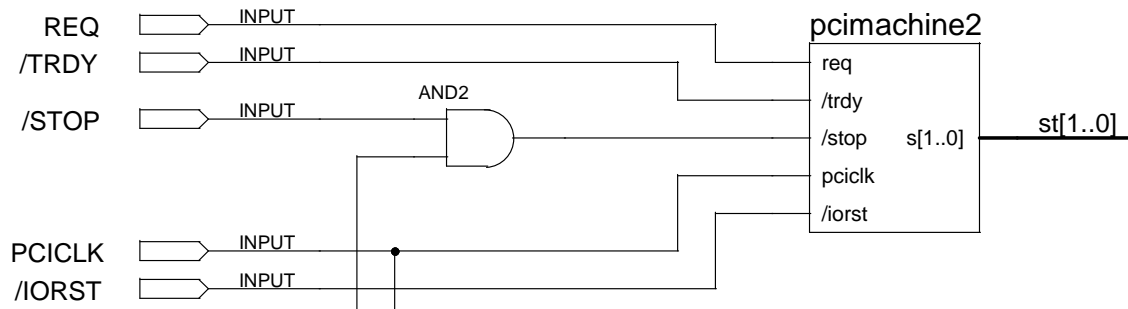
```
BEGIN
```

```
TABLE
```

```
  cbe[3..0], io => ad1, ad0;  
  B"0000" , 1 => 0, 0;  
  B"0001" , 1 => 0, 1;  
  B"0010" , 1 => 0, 0;  
  B"0011" , 1 => 1, 0;  
  B"0100" , 1 => 0, 0;  
  B"0101" , 1 => 0, 1;  
  B"0110" , 1 => 0, 0;  
  B"0111" , 1 => 1, 1;  
  B"1000" , 1 => 0, 0;  
  B"1001" , 1 => 0, 1;  
  B"1010" , 1 => 0, 0;  
  B"1011" , 1 => 1, 0;  
  B"1100" , 1 => 0, 0;  
  B"1101" , 1 => 0, 1;  
  B"1110" , 1 => 0, 0;  
  B"1111" , 1 => 0, 0;  
  B"xxxx" , 0 => 0, 0;
```

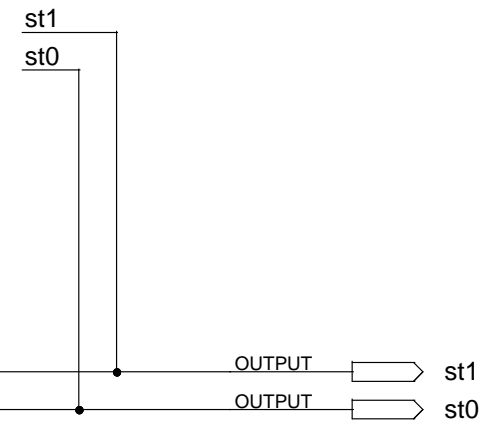
```
END TABLE;
```

```
END;
```

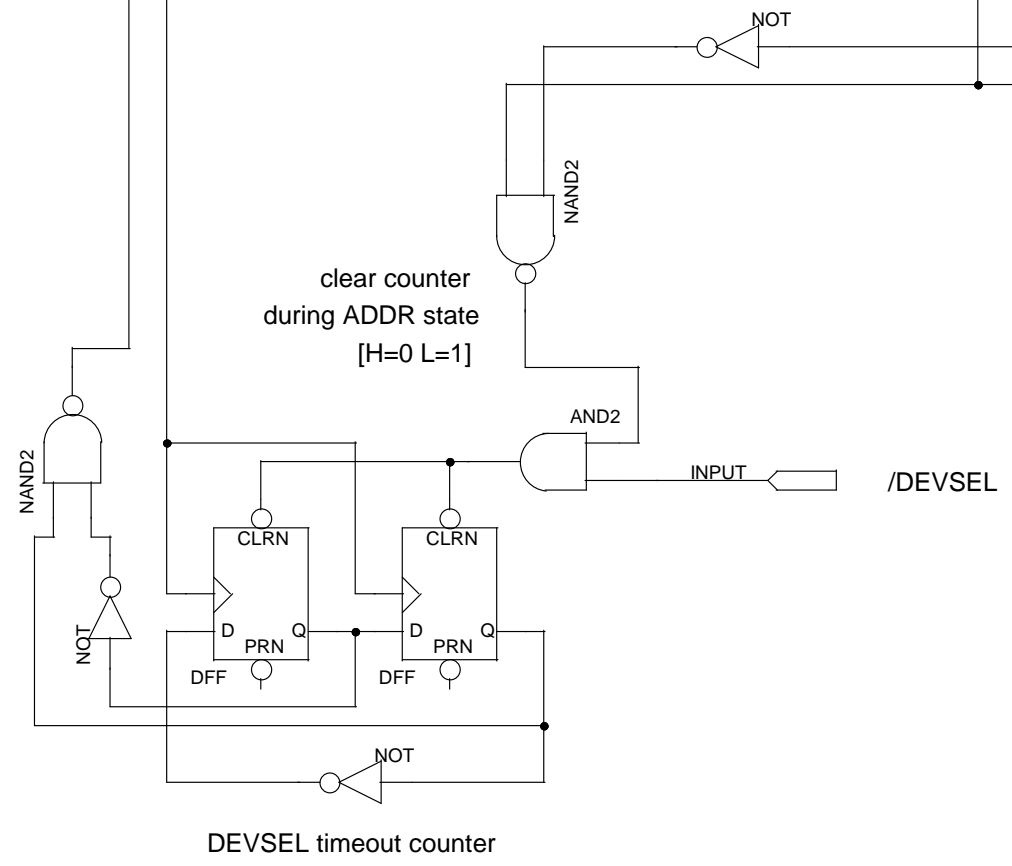


State encoding  
 H L  
 0 0 - drive bus (idle)  
 0 1 - address phase  
 1 1 - data phase  
 1 0 - turnaround cycle

low when  
 timeout  
 detected



### pcimachine



State machine flow:  
 STATE: drive bus  
 Go address phase if REQ=1 and STOP=1 and TRDY=1  
 Else go drive bus.  
 STATE: address phase:  
 Go data phase always.  
 STATE: data phase:  
 Go data phase if STOP=1 and TRDY=1  
 Else go turnaround.  
 STATE turnaround:  
 Go drive bus always.

```

SUBDESIGN pcimachine2
(
    req, /trdy, /stop, pciclk, /iorst : INPUT;
    s[1..0]: OUTPUT;
)
VARIABLE
    seq : MACHINE OF BITS (s[1..0])
        WITH STATES (drive = B"00",
                    addr  = B"01",
                    data  = B"11",
                    turn  = B"10");
BEGIN
    seq.clk = pciclk;
    seq.reset = !/iorst;
TABLE
seq,  req, /trdy, /stop => seq;

drive,  0,    X,    X => drive;
drive,  1,    0,    X => drive;
drive,  1,    1,    0 => drive;
drive,  1,    1,    1 => addr;

addr,   X,    X,    X => data;

data,   X,    0,    X => turn;
data,   X,    1,    0 => turn;
data,   X,    1,    1 => data;

turn,   X,    X,    X => drive;
END TABLE;
END;

```

```

PARAMETERS
(
    SERIAL
);

CONSTANT S0R = ((SERIAL & H"F0000000") DIV H"10000000") XOR H"F";
CONSTANT S1R = ((SERIAL & H"0F000000") DIV H"01000000") XOR H"F";
CONSTANT S2R = ((SERIAL & H"00F00000") DIV H"00100000") XOR H"F";
CONSTANT S3R = ((SERIAL & H"000F0000") DIV H"00010000") XOR H"F";
CONSTANT S4R = ((SERIAL & H"0000F000") DIV H"00001000") XOR H"F";
CONSTANT S5R = ((SERIAL & H"00000F00") DIV H"00000100") XOR H"F";
CONSTANT S6R = ((SERIAL & H"000000F0") DIV H"00000010") XOR H"F";
CONSTANT S7R = ((SERIAL & H"0000000F") DIV H"00000001") XOR H"F";

CONSTANT MANUFACTURER = H"AD47";      % Amiga Inc. hardware developer number %

CONSTANT M0R = ((MANUFACTURER & H"F000") DIV H"1000") XOR H"F";
CONSTANT M1R = ((MANUFACTURER & H"0F00") DIV H"0100") XOR H"F";
CONSTANT M2R = ((MANUFACTURER & H"00F0") DIV H"0010") XOR H"F";
CONSTANT M3R = ((MANUFACTURER & H"000F") DIV H"0001") XOR H"F";

CONSTANT DEVTYPE = 1;                % device type %

CONSTANT D0R = ((DEVTYPE & H"F0") DIV H"10") XOR H"F";
CONSTANT D1R = ((DEVTYPE & H"0F") DIV H"01") XOR H"F";

CONSTANT ROMVEC = H"0000";           % ROM offset (currently unused) %

CONSTANT R0R = ((ROMVEC & H"F000") DIV H"1000") XOR H"F";
CONSTANT R1R = ((ROMVEC & H"0F00") DIV H"0100") XOR H"F";
CONSTANT R2R = ((ROMVEC & H"00F0") DIV H"0010") XOR H"F";
CONSTANT R3R = ((ROMVEC & H"000F") DIV H"0001") XOR H"F";

SUBDESIGN ConfigROM
(
    A[4..0]   : INPUT;
    D[31..28] : OUTPUT;
)
BEGIN
CASE A[] IS
    WHEN 0 => D[] = B"1000"; % $000 - Z3 type, no autolink RAM, no boot ROM %
    WHEN 1 => D[] = B"0101"; % $101 - single dev. board, 512 MB size %

    % device number (see constant above), register $004 %

    WHEN 2 => D[] = D0R;
    WHEN 3 => D[] = D1R;

    WHEN 4 => D[] = B"1100"; % $008 - I/O dev., not shutupable, Z3 size %
    WHEN 5 => D[] = B"1111"; % $108 - Logical size matches physical size %

```

```
WHEN 6 => D[] = B"1111"; % $00C - Reserved %
WHEN 7 => D[] = B"1111"; % $10C - Reserved %
```

```
% manufacturer number (see constant above), registers $010 and $014 %
```

```
WHEN 8 => D[] = M0R;
WHEN 9 => D[] = M1R;
WHEN 10 => D[] = M2R;
WHEN 11 => D[] = M3R;
```

```
% serial number (see constant above), registers $018 to $024 %
```

```
WHEN 12 => D[] = S0R;
WHEN 13 => D[] = S1R;
WHEN 14 => D[] = S2R;
WHEN 15 => D[] = S3R;
WHEN 16 => D[] = S4R;
WHEN 17 => D[] = S5R;
WHEN 18 => D[] = S6R;
WHEN 19 => D[] = S7R;
```

```
% ROM vector (see constant above), registers $028 and $02C %
```

```
WHEN 20 => D[] = R0R;
WHEN 21 => D[] = R1R;
WHEN 22 => D[] = R2R;
WHEN 23 => D[] = R3R;
```

```
% others reserved %
```

```
WHEN OTHERS => D[] = B"1111";
END CASE;
END;
```