Input / Output Address Decoding

1. What is the different between memory address decoding and memory mapped I/O address decoding?

Memory mapped I/O address decoding is the same as memory address decoding (except that the $\overline{IOWC}$ and $\overline{IORC}$ are not used because there is no IN or OUT or INS or OUTS instruction). The I/O is treated as normal memory location.

2. What is the different between memory address decoding and Isolated I/O address decoding?
   - The number of address pins connect to the decoder is different:
     - In memory: decode $A_0$–$A_{31}$, $A_0$–$A_{23}$, or $A_0$–$A_{19}$ for memory,
     - In I/O if fixed address is used it decode $A_0$–$A_7$ and from $A_8$–$A_{15}$ are zeros , and from $A_{16}$–$A_{19}$ are undefined. If variable address are used it decode $A_0$–$A_{15}$ and from $A_{16}$–$A_{19}$ are undefined.
   - $\overline{M/IO}$ and $\overline{RD}$ control are used to activate I/O device for reading ($\overline{IORC}$) and $\overline{M/IO}$ and $\overline{WR}$ control are used to activate I/O device for writing operation($\overline{IOWC}$).

3. How to Decode 8-Bit I/O Port Addresses using Fixed address?
   - Only address connections $A_0$–$A_7$ will be used and the other will be ignored.
   - Figure Q3 illustrates a 74ALS138 decoder that decodes 8-bit I/O ports F0H through F7H.

![Figure Q3](image-url)

The address at Y4 will be F4H according to the following table.

<table>
<thead>
<tr>
<th>A7</th>
<th>A6</th>
<th>A5</th>
<th>A4(G1)</th>
<th>A3(G2A)</th>
<th>A2(C)</th>
<th>A1(B)</th>
<th>A0(A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
</tr>
</tbody>
</table>
The address at Y1 will be F1H according to the following table.

<table>
<thead>
<tr>
<th>A7</th>
<th>A6</th>
<th>A5</th>
<th>A4(G1)</th>
<th>A3(G2A)</th>
<th>A2(C)</th>
<th>A1(B)</th>
<th>A0(A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4. Write a VHDL code to design the PLD version, using a GAL22V10 (a low-cost device) for decoding 8-Bit I/O Port Addresses using Fixed address?

- The VHDL program for the PLD is

```
library ieee;
use ieee.std_logic_1164.all;
entity DECODER_Q4 is
port (
    A7, A6, A5, A4, A3, A2, A1, A0: in STD_LOGIC; ---List Here the inputs
    D0, D1, D2, D3, D4, D5, D6, D7: out STD_LOGIC ); ---List Here the outputs
end;
architecture V1 of DECODER_Q4 is
begin
    D0 <= not( A7 and A6 and A5 and A4 and not A3 and not A2 and not A1 and not A0 ); -- ≈ \overline{F_0} 
    D1 <= not( A7 and A6 and A5 and A4 and not A3 and not A2 and not A1 and A0 ); -- ≈ \overline{F_1} 
    D2 <= not( A7 and A6 and A5 and A4 and not A3 and not A2 and A1 and A0 ); 
    D3 <= not( A7 and A6 and A5 and A4 and not A3 and not A2 and A1 and A0 ); 
    D4 <= not( A7 and A6 and A5 and A4 and not A3 and A2 and not A1 and not A0 ); 
    D5 <= not( A7 and A6 and A5 and A4 and not A3 and A2 and not A1 and A0 ); 
    D6 <= not( A7 and A6 and A5 and A4 and not A3 and A2 and A1 and not A0 ); 
    D7 <= not( A7 and A6 and A5 and A4 and not A3 and A2 and A1 and A0 ); 
end V1;
```

FIGURE Q4: A PLD that generates part selection signals F0H–F0H.
5. How to Decode 16-Bit I/O Port Addresses using variable address using a GAL22V10 (write VHDL code)?

- Figure Q5 illustrates a circuit that contains a PLD and a 4-input NAND gate used to decode I/O ports EFF8H–EFFFH. The NAND gate decodes part of the address (A15, A14, A13, and A11) because the PLD does not have enough address inputs. The output of the NAND gate connects to the Z input of the PLD and is decoded as a part of the I/O port address. The PLD generates address strobes for I/O ports – .

- The program for the PLD is listed in Example 11–3.

```vhdl
library ieee;
use ieee.std_logic_1164.all;
entity DECODER_Q5 is
port ( Z, A12, A10, A9, A8, A7, A6, A5, A4, A3, A2, A1, A0: in STD_LOGIC;
      D0, D1, D2, D3, D4, D5, D6, D7: out STD_LOGIC);
end;
architecture V1 of DECODER_Q5 is
begin
D0 <= not ( not Z and not A12 and A10 and A9 and A8 and A7 and A6 and A5 and A4 and A3
            and not A2 and not A1 and not A0 );
D1 <= not ( not Z and not A12 and A10 and A9 and A8 and A7 and A6 and A5 and A4 and A3
            and not A2 and not A1 and A0 );
D2 <= not ( not Z and not A12 and A10 and A9 and A8 and A7 and A6 and A5 and A4 and A3
            and not A2 and not A1 and A0 );
D3 <= not ( not Z and not A12 and A10 and A9 and A8 and A7 and A6 and A5 and A4 and A3
            and not A2 and not A1 and A0 );
D4 <= not ( not Z and not A12 and A10 and A9 and A8 and A7 and A6 and A5 and A4 and A3
            and not A2 and not A1 and A0 );
D5 <= not ( not Z and not A12 and A10 and A9 and A8 and A7 and A6 and A5 and A4 and A3
            and not A2 and not A1 and A0 );
D6 <= not ( not Z and not A12 and A10 and A9 and A8 and A7 and A6 and A5 and A4 and A3
            and not A2 and not A1 and A0 );
D7 <= not ( not Z and not A12 and A10 and A9 and A8 and A7 and A6 and A5 and A4 and A3
            and not A2 and not A1 and A0 );
end V1;
```

To find the address of the output D4 for example

1. get the code for D4
   
   ```vhdl
   D4 <= not ( not Z and not A12 and A10 and A9 and A8 and A7 and A6 and A5 and A4 and A3
            and not A2 and not A1 and A0 );
   ```

2. remove the first NOT
D4 <= ( not Z and not A12 and A10 and A9 and A8 and A7 and A6 and A5 and A4 and A3 and A2 and not A1 and not A0 );

not Z means (inputs to the NAND A11=1,A13=1,A14=1 A15=1) and not A12 means (A12=0) , and A10 means (A10=1) and A9 means (A9=1) , and A8 means (A8=1) , and A7 means (A7=1) and A6 means (A6=1), and A5 means (A5=1), and A4 means (A4=1) and A3 means (A3=1), and A2 means (A2=1), and not A1 means (A1=0) and not A0 means (A0=0)

3. fill in the following table

<table>
<thead>
<tr>
<th>A15</th>
<th>A14</th>
<th>A13</th>
<th>A12</th>
<th>A11</th>
<th>A10</th>
<th>A9</th>
<th>A8</th>
<th>A7</th>
<th>A6</th>
<th>A5</th>
<th>A4</th>
<th>A3</th>
<th>A2</th>
<th>A1</th>
<th>A0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>E</td>
<td>F</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

From the table the address that decode the I/O device connected to the output pin D4 will be **EFFE H**

By the same method find the address that decode the I/O device connected to the output pins D1, D2, D3, D4, D5, D6 (It will be as the following figure)

![Diagram](image)

**FIGURE Q5:** A PLD that decodes 16-bit I/O ports EFF8H through EFFFH.

**Important Note:** any change in the VHDL code will be change the address at the output of the PLD device.
6. **What is the different of I/O Port address and I/O port wide?**
   - **I/O Port address** are the address the decode the I/O device to enable it for read or write operation
   - **I/O port wide** is the wide of data read or written from the I/O device and it may be 8-bit wide or 16-bit wide or 32-bit wide or 64-bit wide.

7. **How the data can be transferred between microprocessor and I/O device?**
   - **In case of 8-bit wide I/O devices**: Data transferred to an 8-bit I/O device exist in one of the I/O banks in a 16-bit microprocessor such as the 80386SX. There are 64K different 8-bit ports, because a 16-bit port uses only one 8-bit ports (one of the I/O banks).
   - **16-bit wide I/O devices**: The I/O system on such a microprocessor contains two 8-bit memory banks, just as memory does. This is illustrated in Figure 11–13, which shows the separate I/O banks for a 16-bit system such as the 80386SX.
   - In a 16-bit microprocessor such as the 80386SX. There are only 32K different 16-bit ports because a 16-bit port uses two 8-bit ports (the two I/O banks).

8. **How to connect two different 8-bit (port-wide) I/O devices using fixed address method (8-bit I/O address) located at 40H and 41H for write operation?**
   - Because these are 8-bit output devices and because they appear in different I/O banks, separate I/O write signals are generated to clock a pair of latches that capture port data.
   - Note that all I/O ports use 8-bit addresses.
   - Thus, ports 40H and 41H can each be addressed as separate 8-bit ports, or together as one 16-bit port.

<table>
<thead>
<tr>
<th>Port</th>
<th>A7</th>
<th>A6</th>
<th>A5</th>
<th>A4</th>
<th>A3</th>
<th>A2</th>
<th>A1</th>
<th>A0</th>
</tr>
</thead>
</table>

Figure Q7: The I/O banks found in the 8086, 80186, 80286, and 80386SX.
According to the above tables, the program for the PLD decoder used will be:

```
-- VHDL code for the decoder of Figure Q8
library ieee;
use ieee.std_logic_1164.all;
entity DECODER_Q8 is
   port (BHE, IOWC, A7, A6, A5, A4, A3, A2, A1, A0: in STD_LOGIC;
         D0, D1: out STD_LOGIC);
end;
architecture V1 of DECODER_Q8 is
begin
   D0 <= not BHE and not IOWC and A7 and not A6 and A5 and A4 and A3 and A2 and A1 and A0;
   D1 <= BHE and not IOWC and A7 and not A6 and A5 and A4 and A3 and A2 and A1 and not A0;
end V1;
```

- The design will be as shown in figure Q8
9. How to connect 16-bit I/O device using fixed address method (8-bit I/O address) located at 64H and 65H for read operation?
   - The PLD decoder does not have a connection for address bits BHE because these signals do not apply to 16-bit-wide I/O devices.
   - Only port number 64 will be decoded and 65 will be generated automatically.

<table>
<thead>
<tr>
<th>Port</th>
<th>A7</th>
<th>A6</th>
<th>A5</th>
<th>A4</th>
<th>A3</th>
<th>A2</th>
<th>A1</th>
<th>A0</th>
</tr>
</thead>
<tbody>
<tr>
<td>64 H</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>VHDL</td>
<td>Not A7</td>
<td>And A6</td>
<td>And A5</td>
<td>And Not A4</td>
<td>And not A3</td>
<td>And A2</td>
<td>And Not A1</td>
<td>And Not A0</td>
</tr>
</tbody>
</table>

- The program for the PLD

```vhdl
-- VHDL code for the decoder of Figure Q9
library ieee;
use ieee.std_logic_1164.all;
entity DECODER_Q9 is
port ( IORC, A7, A6, A5, A4, A3, A2, A1: in STD_LOGIC; D0: out STD_LOGIC );
end;
```
architecture V1 of DECODER_Q9 is
begin
D0 <= not IORC and not A7 and A6 and A5 and not A4 and not A3 and not A2 and not A1 and not A0;
end V1;

- As illustrated in Example Q9, shows how the enable signals are generated for the three-state buffers (74HCT244) used as input devices.

FIGURE Q8: A 16-bit-wide port decoded at I/O addresses 64H and 65H.

10. What is the programmable Peripheral Interface (PPI)?

The 82C55 programmable peripheral interface (PPI) is:
- A very popular, low-cost interfacing component found in many applications.
- The PPI, which has 24 pins for I/O that are programmable in groups of 12 pins,
- Have groups that operate in three distinct modes of operation.
- The 82C55 can interface any TTL-compatible I/O device to the microprocessor.
- The 82C55 (CMOS version) requires the insertion of wait states if operated with MP using higher than 8 MHz
- The 82C55 is used for interface to the keyboard and the parallel printer port in PC.

11. The 82C55 PPI has how many programmable I/O PIN CONNECTION. List the pins that belong to group A & group B in the 82C55. Also mention two pins which accomplish internal I/O port section.
The 82C55 has 24 programmable I/O.

- **Group A consist:**
  1. port A
  2. upper half of port C (PC7-PC4)
  3. A1,A0,WR

- **Group b consist from :**
  1. port B (PB7-PB0)
  2. lower half of port c(PC3-PC0)
  3. RD
  4. RESET
  5. CK
  6. VCC
  7. GND

  Two pins accomplish internal I/O port address is A0 and A1

12. What three modes of operation are available to the 82C55 PPI? Mention the purpose of STB signal is strobes input operation of the 82C55

Mode 0: Latched I/O
Mode 1: strobes I/O
Mode2: bi-directional I/O.

<table>
<thead>
<tr>
<th>$A_1$</th>
<th>$A_0$</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Port A</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Port B</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Port C</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Command register</td>
</tr>
</tbody>
</table>

13. How to connect 82C55 PPI to a microprocessor
Figure Q13: The 82C55 interfaced to the low bank of the 80386SX microprocessor. It functions at 8-bit I/O port addresses C0H (port A), C2H (port B), C4H (port C), and C6H (command register).
14. How to program 82C55 PPI?

Command byte A

7 6 5 4 3 2 1 0

Group B

Port C (PC3 → PC0)
1 = input
0 = output

Port B
1 = input
0 = output

Mode
00 = mode 0
01 = mode 1

Command byte B

7 6 5 4 3 2 1 0

Group A

Port C (PC7 → PC4)
1 = input
0 = output

Port A
1 = input
0 = output

Mode
00 = mode 0
01 = mode 1
1X = mode 2