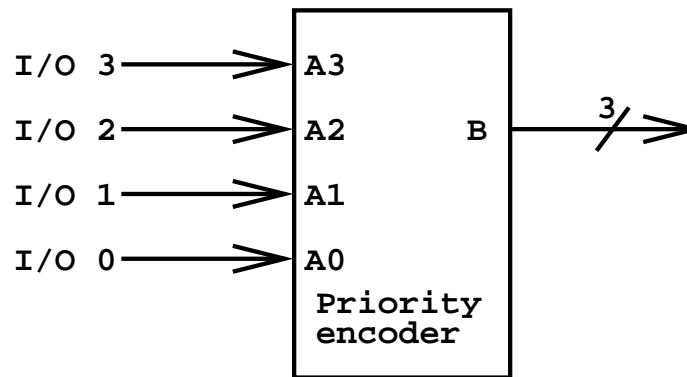


Priority Decoder :

0

A CPU may have one or more interrupt input lines, which need to be prioritized. This is done with a priority decoder. In this example 'A' could represent 4 separate interrupt lines and output 'B' identifies the active input with the highest priority. The highest priority is assumed to be the most significant bit of the input bus.



B could carry the decimal value 4, 3, 2, 1 or 0 (in binary) representing the most significant active input.

Priority Decoder :

1

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity decod is
port( A : in std_logic_vector(3 downto 0);
      B : out std_logic_vector(2 downto 0));
end decod;
```

Priority Decoder :

2

architecture rtl_1 of decod is

```
begin
```

```
  process( a )
```

```
  begin
```

```
    b <= "100";
```

```
    for i in a'range -- = 3, 2, 1, 0
```

```
    loop
```

```
      exit when a(i)='1';
```

```
      b <= std_logic_vector(to_unsigned(i,3));
```

```
    end loop;
```

```
  end process;
```

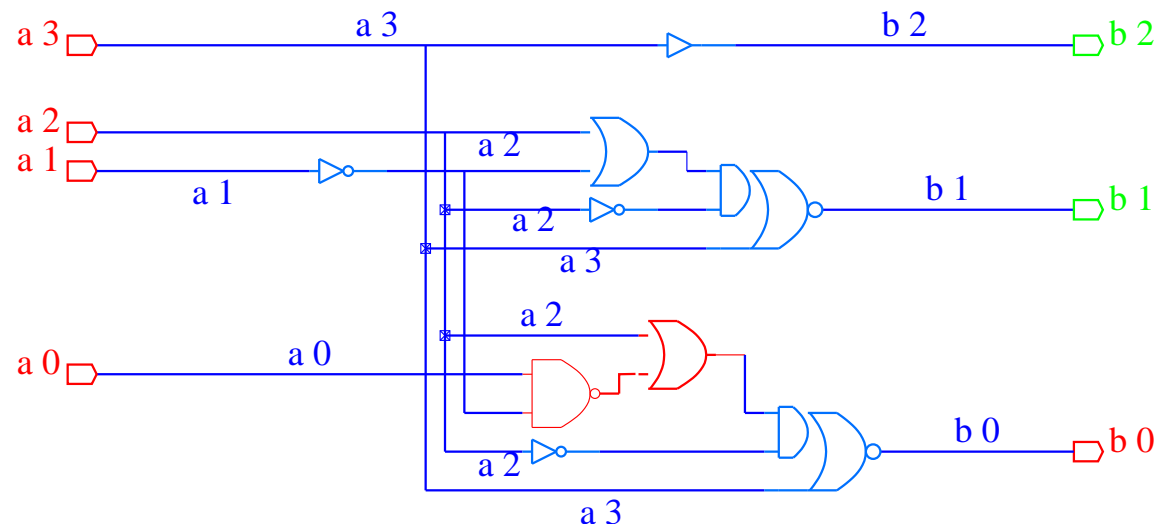
```
end rtl_1;
```

In this example the output will be the binary value +1 of the most significant active bit in the input vector, defined by the direction of the range declaration.

Priority Decoder :

3

The results of synthesis with the Alliance tools



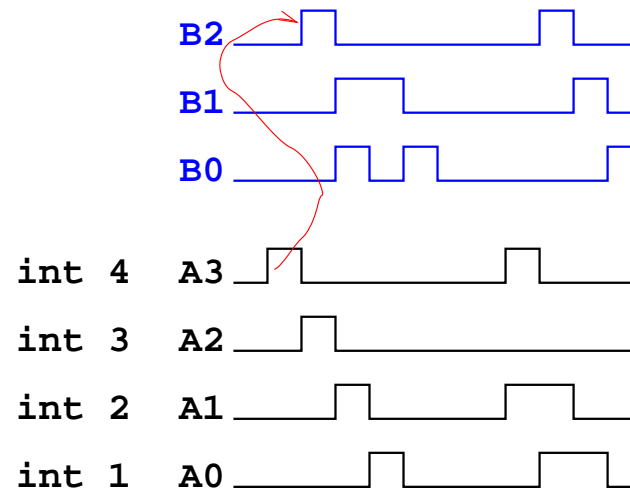
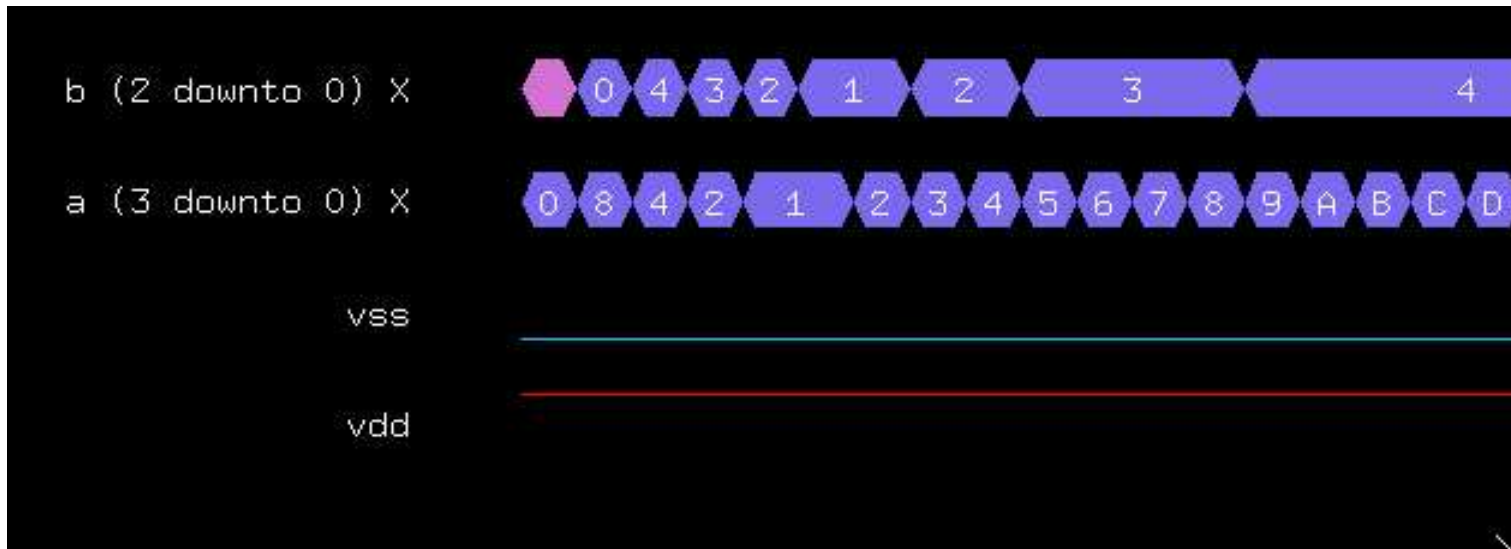
There are several points to note with the synthesis of this code :

- The value output for the highest priority is defined by the initial assignment to b.
- Omitting a default assignment to b results in registers (or failure).

Priority Decoder :

3

This gives the anticipated results when simulated, based on the assignment to b.



Creating a ROM :

1

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity rom is
  port( adr : in std_logic_vector(1 downto 0);
        o0  : out std_logic_vector(3 downto 0)
        );
end rom;
```

This is the black-box description of a 4 x 4 bit ROM

architecture rtl_1 of rom is

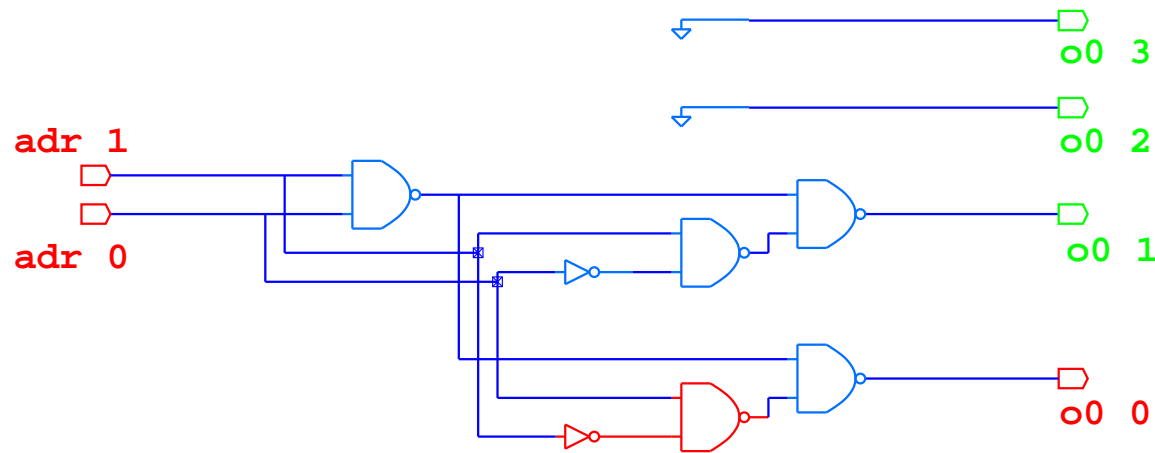
```
subtype my_word is std_logic_vector(3 downto 0);  
type my_array is array (0 to 3) of my_word;  
constant s : my_array := ( "0000", "0001",  
                           "0010", "0011" );
```

begin

```
o0 <= s(to_integer(unsigned(adr)));
```

end rtl_1;

The results of synthesis with the Alliance tools



Note that the 2 most significant bits are wired to the power supply as they are always zero. Is this what you would have expected?

Creating RAM :

1

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity ram is
port( clk,wr : in std_logic;
      adr   : std_logic_vector(3 downto 0);
      i0    : in std_logic_vector(7 downto 0);
      o0    : out std_logic_vector(7 downto 0)
);
end ram;
```

Note that this creates RAM with separate inputs and outputs, but does include a write enable. This is a 15 x 8 bit RAM.

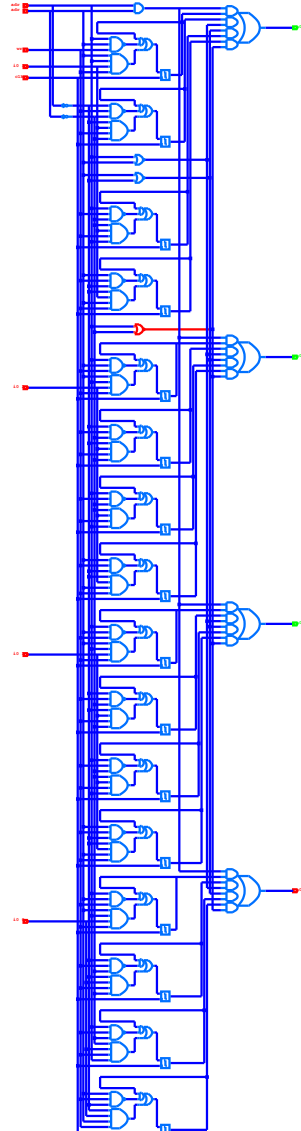
```
architecture rtl_1 of ram is
  type my_array is array (0 to 15) of
    std_logic_vector(7 downto 0);
  signal s : my_array;
begin
  process(clk)
  begin
    if(rising_edge(clk)and w = '1') then
      s(to_integer(unsigned(adr))) <= i0;
    end if;
  end process;
  o0 <= s(to_integer(unsigned(adr)));
end rtl_1;
```

Note that the version given in 'man 5 vasy' doesn't synthesisze.

RAM Synthesis :

3

The results of synthesis with the Alliance tools for a 4 x 4 bit Ram.



A much more efficient implementation of SRAM

