CSA vs. Conditional Signal Assignment

- Concurrent Signal Assignment Statements are suitable for describing gate-level circuits.
- Models for higher level abstraction are difficult to express with Concurrent Signal Assignment Statements.
- Higher level abstraction models are required for multiplexors, decoders …
- VHDL provides Conditional Signal Assignment statements for these situations.

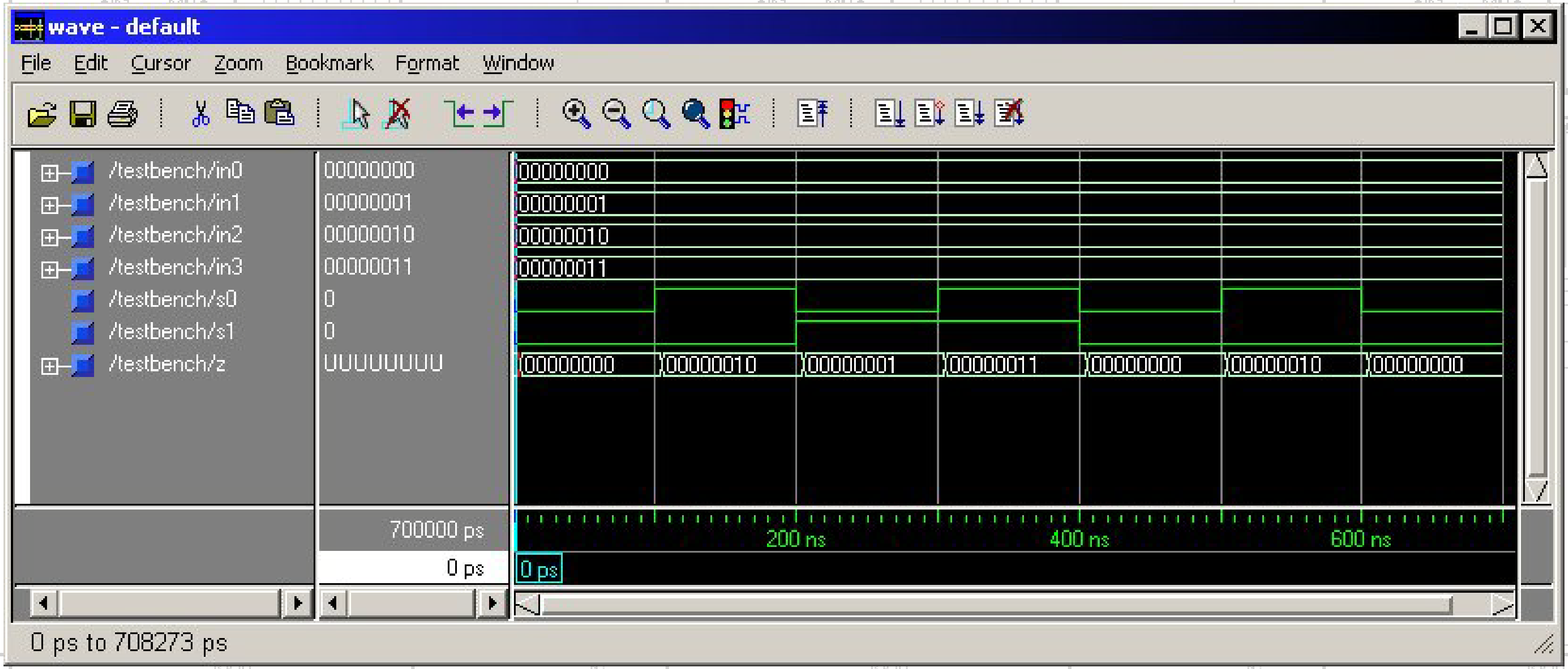
4 to 1 - 8bit Multiplexor

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity mux4 is
   Port (in0, in1, in2, in3: in std_logic_vector (7 downto 0);
            s0, s1: in std_logic;
            z: out std_logic_vector (7 downto 0));
end mux4;

architecture behavioural of mux4 is
begin
z <= in0 after 5ns when s0 = `0` and s1 = `0` else
        in1 after 5ns when s0 = `0` and s1 = `1` else
        in2 after 5ns when s0 = `1` and s1 = `0` else
        in3 after 5ns when s0 = `1` and s1 = `1` else
        “00000000” after 5ns;
end behavioural;
```

Conditional Signal Assignment

- In the 4to1 – 8bit multiplexor example:
- If S1 or S2 changes the concurrent assignment statement is executed.
  - All four conditions may be checked.
- The order is relevant.
- The evaluation takes place in the order that they appear.
- The first true condition determines the output.
- The order should reflect the physical implementation.

Waveform (4 to 1 – 8bit Multiplexor)



Signal	Value	Waveform
/testbench/in0	00000000	00000000
/testbench/in1	00000001	00000001
/testbench/in2	00000010	00000010
/testbench/in3	00000011	00000011
/testbench/s0	0	
/testbench/s1	0	
/testbench/z	UUUUUUUU	00000000 00000010 00000001 00000011 00000000 00000010 00000000

700000 ps

0 ps

0 ps to 708273 ps

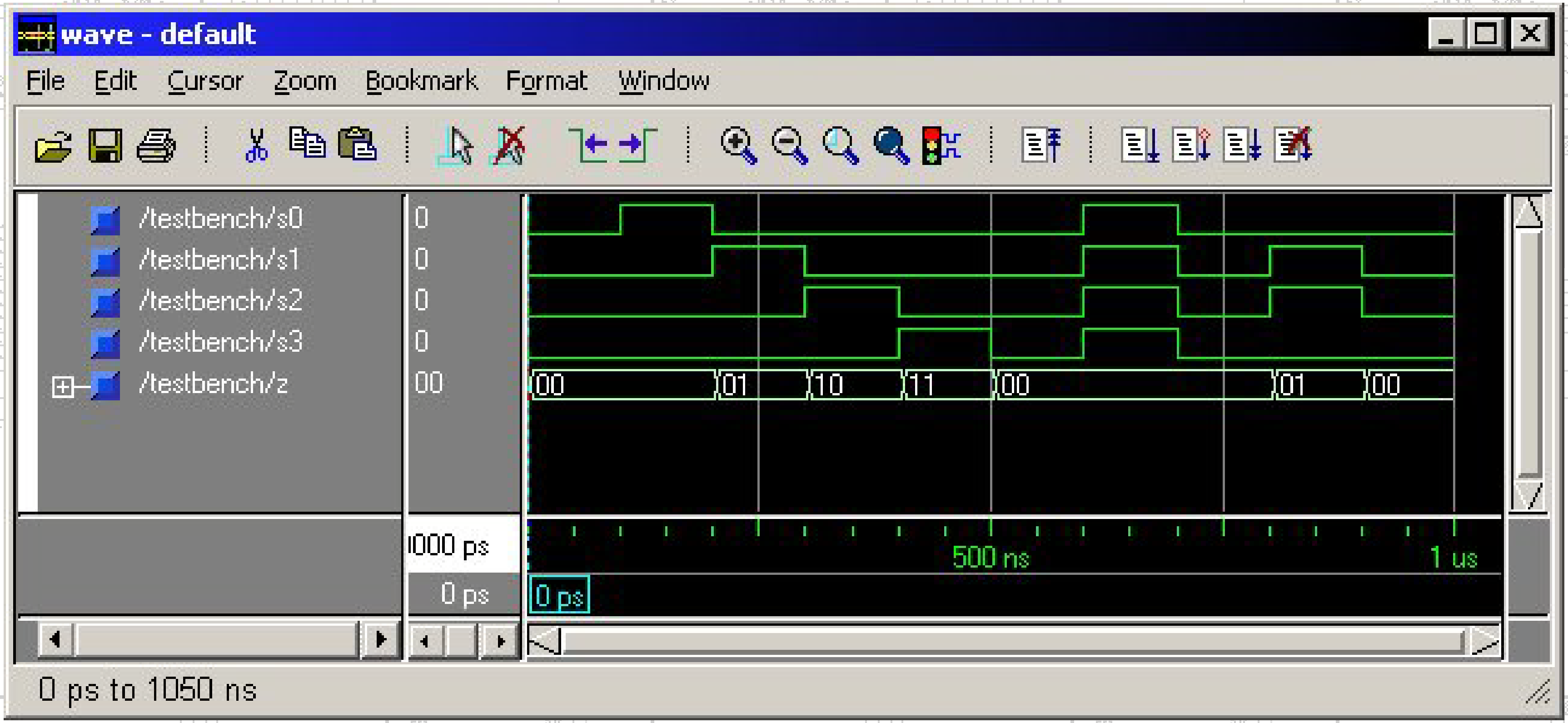
4 to 2 - Priority Encoder

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity four_to_two_priority is
   Port ( S0, S1, S2, S3 : in std_logic;
          Z : out std_logic_vector(1 downto 0));
end four_to_two_priority;

architecture Behavioral of four_to_two_priority is
begin
Z <= "00" after 5 ns when S0='1' else
      "01" after 5 ns when S1='1' else
      "10" after 5 ns when S2='1' else
      "11" after 5 ns when S3='1' else
      "00" after 5 ns;
end Behavioral;
```

Conditional Signal Assignment

- The order in the conditional signal assignment statement in the 4 to 2- Priority Encoder example is important.
- Note in the example:
  - The last statement set the output to zero.
  - This is necessary because the select signals can have values other than '1' and '0'.
  - This is the case because the select signals are declared as std_logic and std_logic_vector.

Waveform (4 to 2 -Priority Encoder)



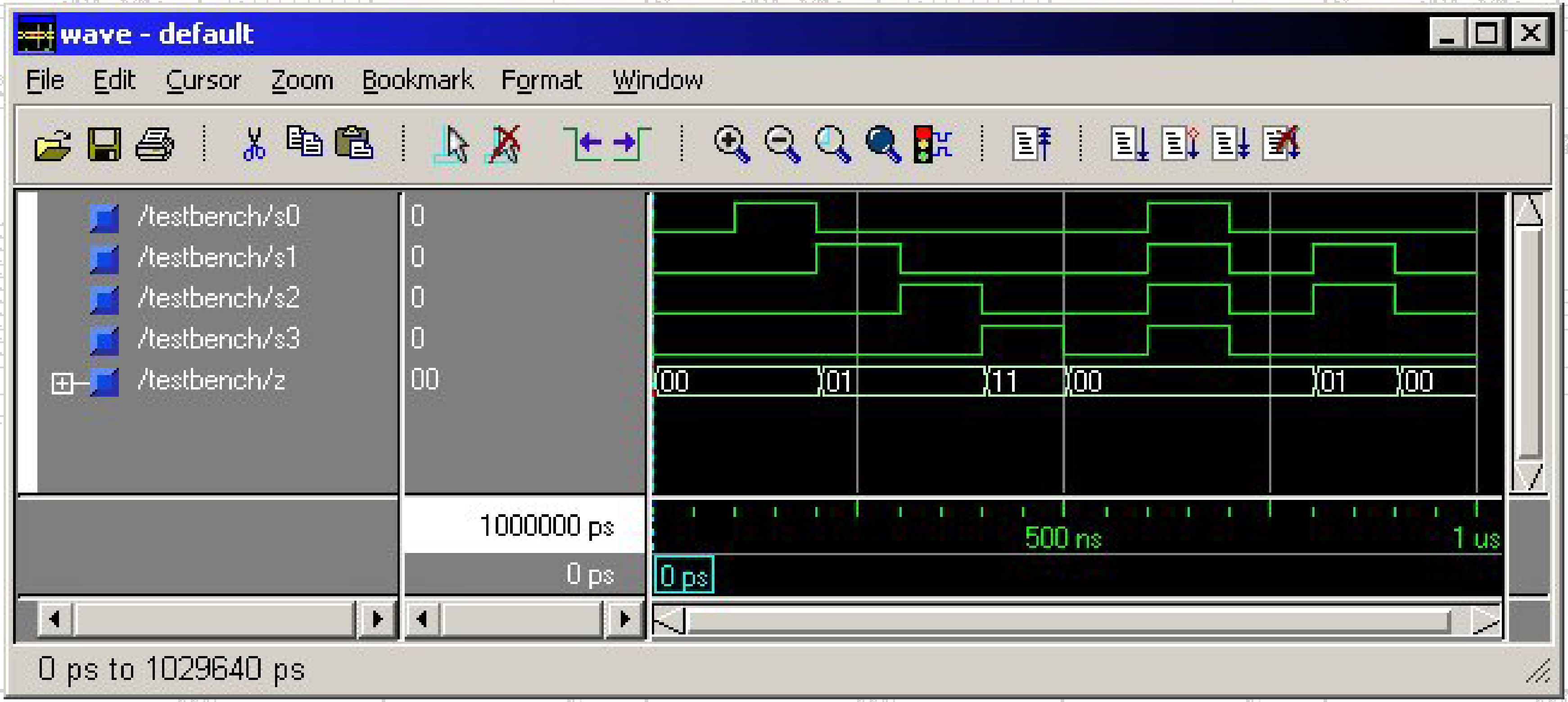
```
/testbench/s0   0
/testbench/s1   0
/testbench/s2   0
/testbench/s3   0
/testbench/z    00

000 ps
   0 ps
```

0 ps to 1050 ns

Unaffected

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity four_to_two_priority is
   Port ( S0, S1, S2, S3 : in std_logic;
          Z : out std_logic_vector(1 downto 0));
end four_to_two_priority;

architecture Behavioral of four_to_two_priority is
begin
Z <= "00" after 5 ns when S0='1' else
      "01" after 5 ns when S1='1' else
      unaffected when S2='1' else
      “11" after 5 ns when S3='1' else
      "00" after 5 ns;
end Behavioral;
```

Waveform (Unaffected)



/testbench/s0 0
/testbench/s1 0
/testbench/s2 0
/testbench/s3 0
/testbench/z 00

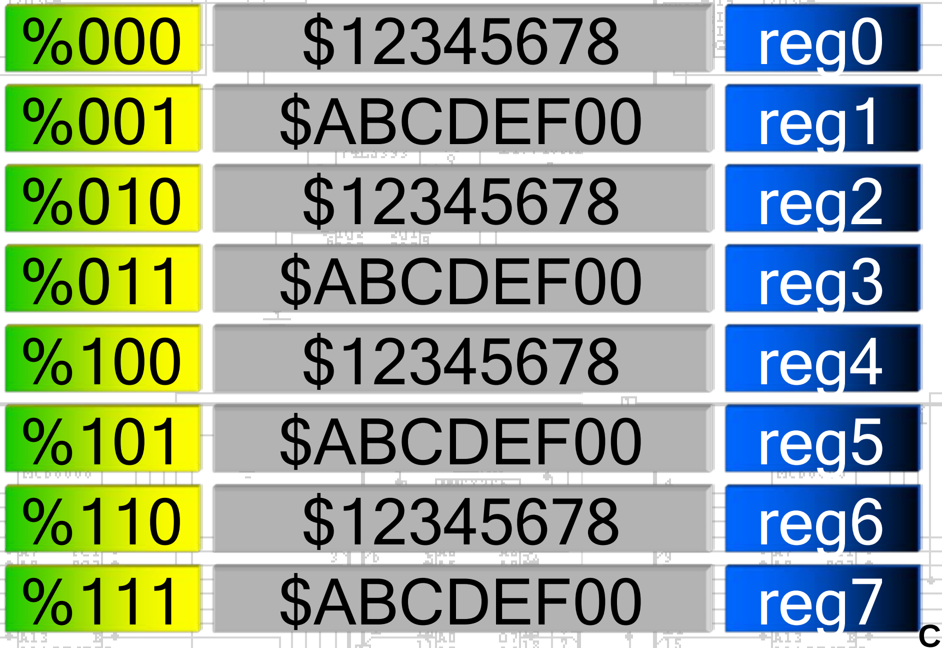
1000000 ps
0 ps

0 ps to 1029640 ps

Selected

Signal Assignment Statement

- Signal value is determined by the select expression
- In this example we read from a register file with eight registers (reg0…reg7)
  - Read only register file with to read ports

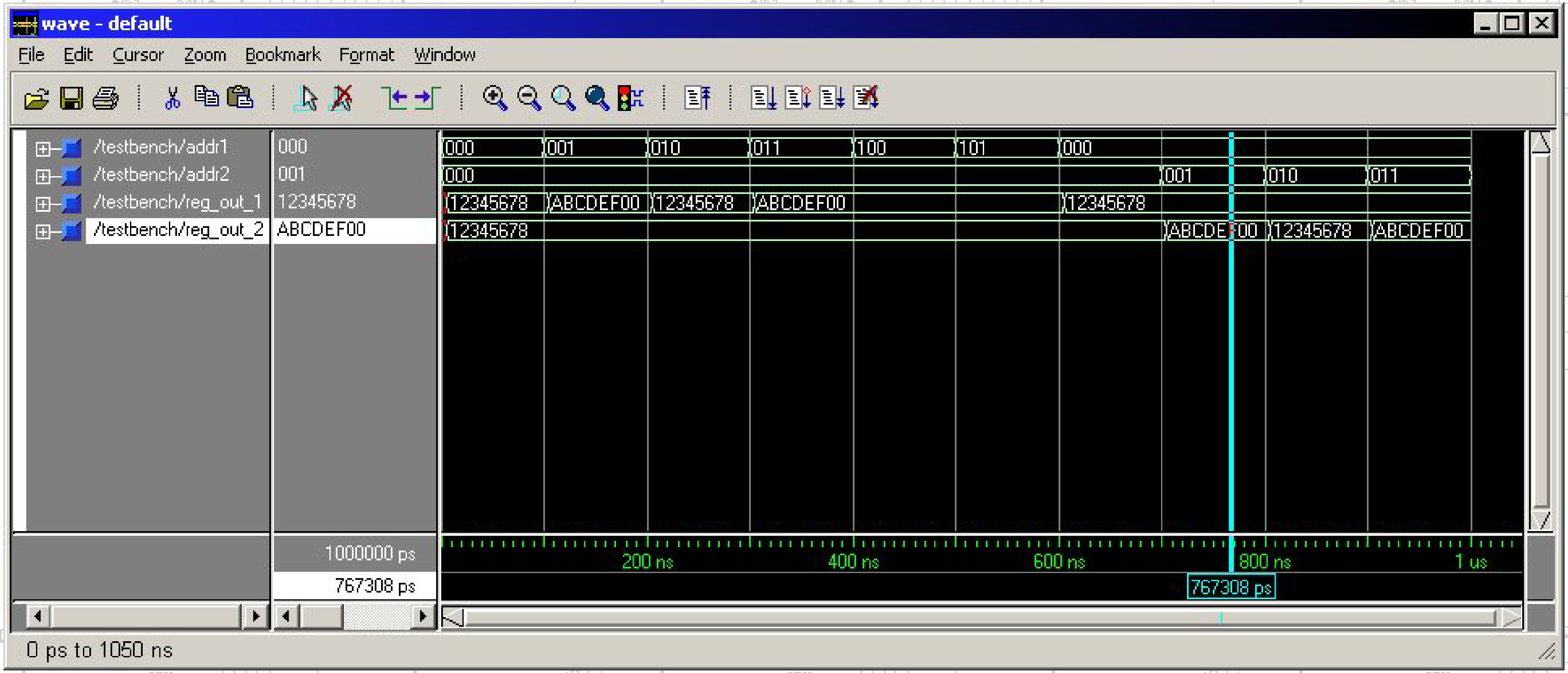


%000	$12345678	reg0
%001	$ABCDEF00	reg1
%010	$12345678	reg2
%011	$ABCDEF00	reg3
%100	$12345678	reg4
%101	$ABCDEF00	reg5
%110	$12345678	reg6
%111	$ABCDEF00	reg7

SSA Statement Example

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity reg_file is
    Port ( addr1, addr2 : in std_logic_vector(2 downto 0);
           reg_out_1, reg_out_2 : out std_logic_vector(31 downto 0));
end reg_file;
architecture Behavioral of reg_file is
signal reg0, reg2, reg4, reg6: std_logic_vector(31 downto 0):= x"12345678";
signal reg1, reg3, reg5, reg7: std_logic_vector(31 downto 0):= x"abcdef00";
begin
with addr1 select
reg_out_1 <= reg0 after 5 ns when "000",
                      reg1 after 5 ns when "001",
                      reg2 after 5 ns when "010",
                      reg3 after 5 ns when "011",
                      reg3 after 5 ns when others;
with addr2 (1 downto 0) select
reg_out_2 <= reg0 after 5 ns when "00",
                      reg1 after 5 ns when "01",
                      reg2 after 5 ns when "10",
                      reg3 after 5 ns when "11",
                      reg3 after 5 ns when others;
end Behavioral;
```

Waveform (Select)



```
/testbench/addr1      000        000 001 010 011 100 101 000
/testbench/addr2      001        000                         001 010 011
/testbench/reg_out_1  12345678   12345678 ABCDEF00 12345678 ABCDEF00 12345678
/testbench/reg_out_2  ABCDEF00   12345678                    ABCDEF00 12345678 ABCDEF00

                      1000000 ps        200 ns   400 ns   600 ns   800 ns   1 us
                      767308 ps                                767308 ps

0 ps to 1050 ns
```

Selected Signal Assignment

- Similar to case statement in conventional languages.
- Choices are not evaluated in sequence.
- Only one must be true.
- The statement must cover all possible combinations.
- The others clause must be used in situation were not all possible combinations are covered by the select statement.

reg3 after 5 ns when others;

- In the second select statement operates on a subset of the address range (**addr2 (1 downto 0)**).
- The when others clause is still required because addr2 is declared as std_logic_vector and can therefore take 9 values.
- unaffected is may also be used in this type of statement.

```
with addr2 (1 downto 0) select
reg_out_2 <= reg0 after 5 ns when "00",
             reg1 after 5 ns when "01",
             reg2 after 5 ns when "10",
             reg3 after 5 ns when "11",
             reg3 after 5 ns when others;
end Behavioral;
```