INF5430 VHDL Testbench Design Case

Roar Skogstrøm,
Institutt for Informatikk, Universitetet i Oslo

- The complete testbench design case is on the INF5430 semester web page!

- The design is compiled into 3 Modelsim libraries made by the vlib command.

- All libraries shall be in the top/libs directory and they are:
    - ”chip_lib”, ”tb_chip_lib” and ”proasic3e”.

- The design is compiled in the ”scripts/modelsim” directory.

- **All file paths in the scripts in the ”scripts/modelsim” directory and library paths in the ”scripts/modelsim/modelsim.ini” file have to be changed.**

- The design is compiled in the scripts/modelsim directory with the commands:
    - vcom –work proasic3e ../../packages/proasic3e.vhd
    - comp_all

- The testbench is started in the simulation ”case_?” directory.

- The simulation case directory must have a Linux link to the modelsim.ini file in the ”scripts/modelsim” directory.

- The simulation case directory must also have:
  - Linux link ”testbench” to the vhdl testbench file (i.e. ”top/tb/tb_chip_beh.vhd”)
  - Linux link ”chip_addrmap” to the project design memory map file (i.e. the ”packages/chip_pck.vhd” file)

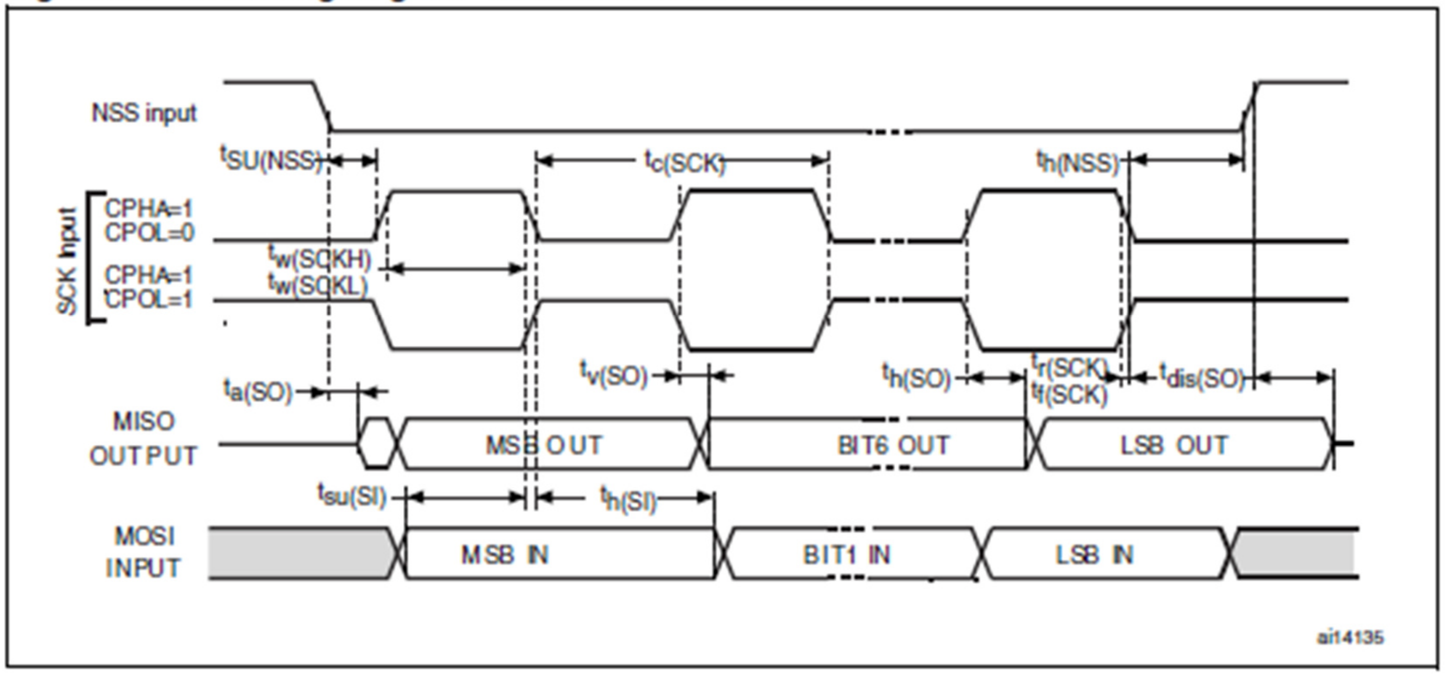
- These links may be changed to file copies in each simulation case for Windows, but NOT a good solution (use Linux …..).

- The simulation is then started in the ”case_?” directory with the command:
  - vsim –lib tb_chip_lib tb_top
  - Then do the «run –all» command and the «start.cmd» file is simulated.
  - The result log file «tb_top.log» shows the result of all commands.

```
UUT: entity chip_lib.top(str)
    port map (
        refclk       => mclk,
        fpga_rst_n   => fpga_rst_n,
        a            => a,
        advn_ale     => advn_ale,
        be_n         => be_n,
        cs_n         => cs_n,
        ad           => ad,
        oe_n         => oe_n,
        we_n         => we_n,
        wait_n       => wait_n,
        wp_n         => wp_n,
        sck          => sck,
        nss          => nss,
        mosi         => mosi,
        miso         => miso,
        irq_uc       => irq_uc,
        d            => stim_d,
        q            => check_d,
        we           => we,
        re           => re,
        full         => full,
        empty        => empty,
        rdcnt        => rdcnt,
        wrcnt        => wrcnt,
        irq_n        => irq_n);
```

```
stim_0: stim
    port map (
        mclk         => mclk,
        rst_n        => fpga_rst_n,
        stim_rst     => stim_rst,
        stim_run_str => stim_run_str,
        stim_d       => stim_d,
        we           => we,
        full         => full,
        wrcnt        => wrcnt);

check_0: check
    port map (
        mclk           => mclk,
        rst_n          => fpga_rst_n,
        check_rst      => check_rst,
        check_run_str  => check_run_str,
        check_complete => check_complete,
        check_result   => check_result,
        check_d        => check_d,
        re             => re,
        empty          => empty,
        rdcnt          => rdcnt);
```

Figure 22. SPI timing diagram - slave mode and CPHA = 1[1]



• STM32L15xxx microcontroller Serial Peripheral Interface (SPI) timing diagram; SCK with CPHA=1 and CPOL=0

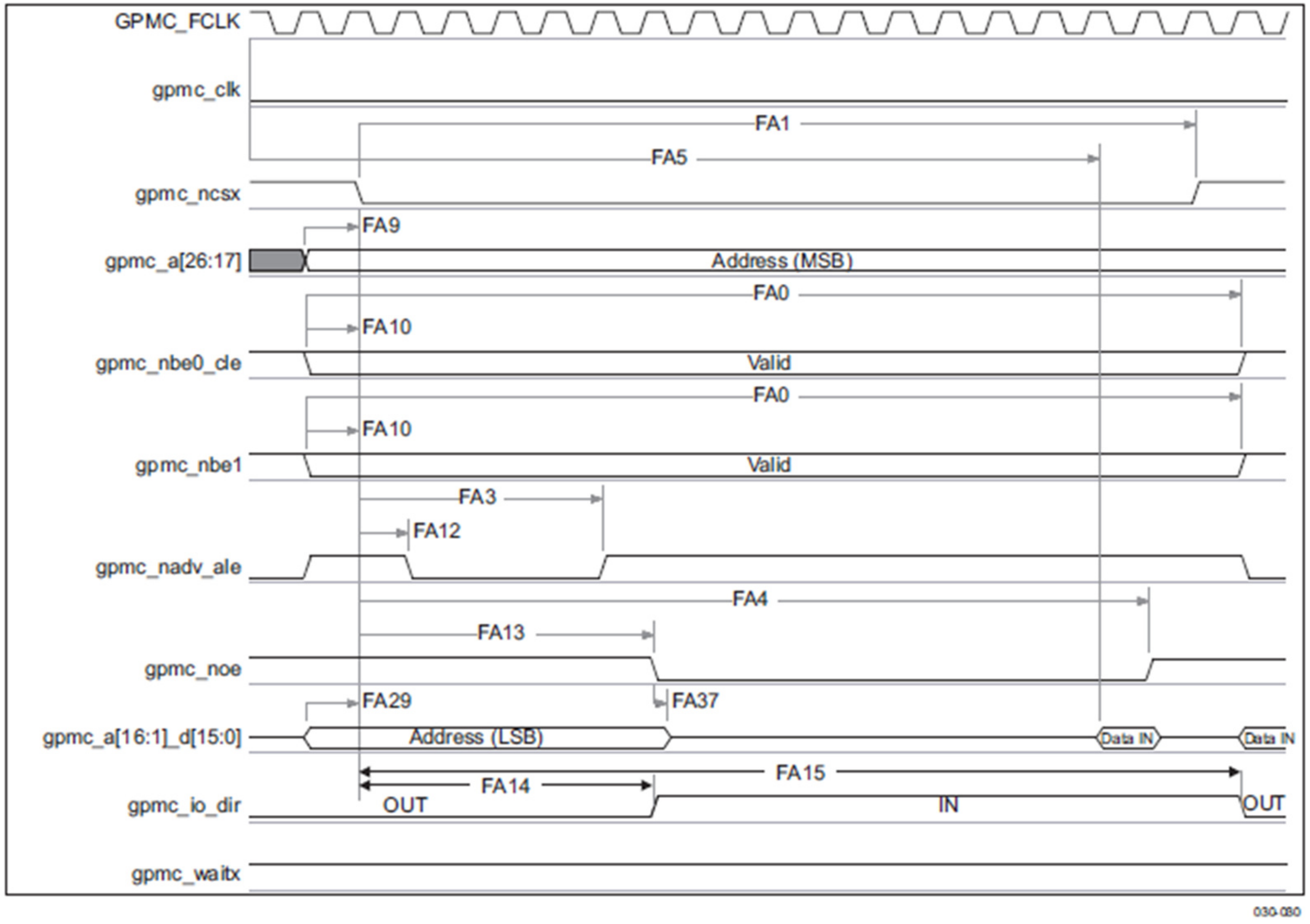


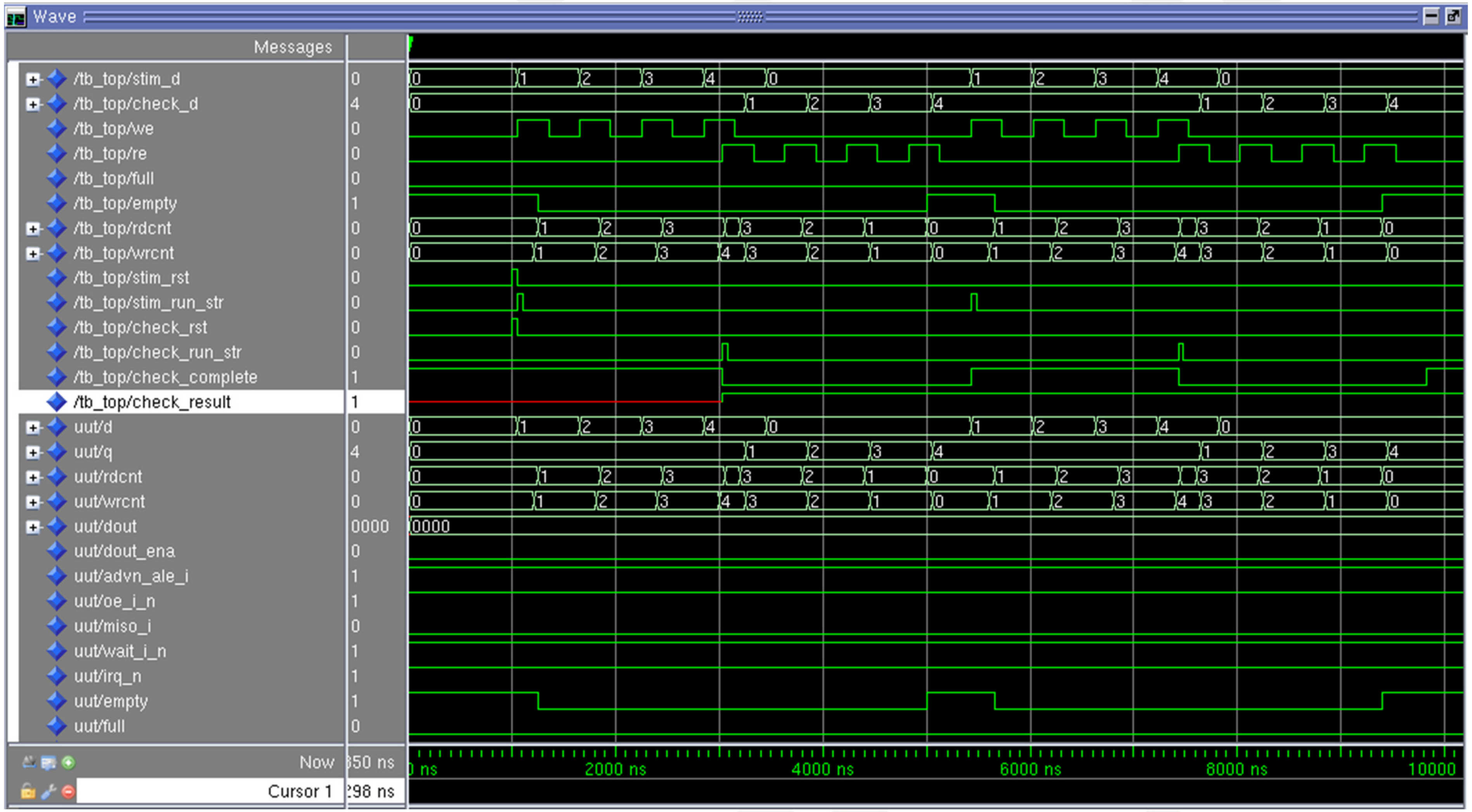
Figure 6-11. GPMC/Multiplexed NOR Flash – Asynchronous Read – Single Word Timing(1)(2)(3)

```
P_STIM_GEN: process
  begin

    stim_d_i <= (others => '0');
    we       <= '0';

    wait until rising_edge(stim_run_str);

    loop1: for i in 1 to 4 loop

      stim_d_i <=
           std_logic_vector(unsigned(stim_d_i)+1);

      we <= '1';

      wait on mclk until stim_rst='1' for 300 ns;
      if stim_rst='1' then
        exit loop1;
      end if;

      we <= '0';

      wait on mclk until stim_rst='1' for 300 ns;
      if stim_rst='1' then
        exit loop1;
      end if;

    end loop;
  end process;

  stim_d <= stim_d_i;
```

```
P_CHECK: process
  begin

    check_complete <= '1';
    re <= '0';

    wait until rising_edge(check_run_str);

    check_complete <= '0';
    check_result   <= '1';

    loop1: for i in 1 to 4 loop

      re <= '1';

      wait on mclk until check_rst='1' for 300 ns;
      if check_rst='1' then
        exit loop1;
      end if;

      if unsigned(check_d)/=to_unsigned(i,16) then
        check_result <= '0';
      end if;

      re <= '0';

      wait on mclk until check_rst='1' for 300 ns;
      if check_rst='1' then
        exit loop1;
      end if;

    end loop;
  end process;
```

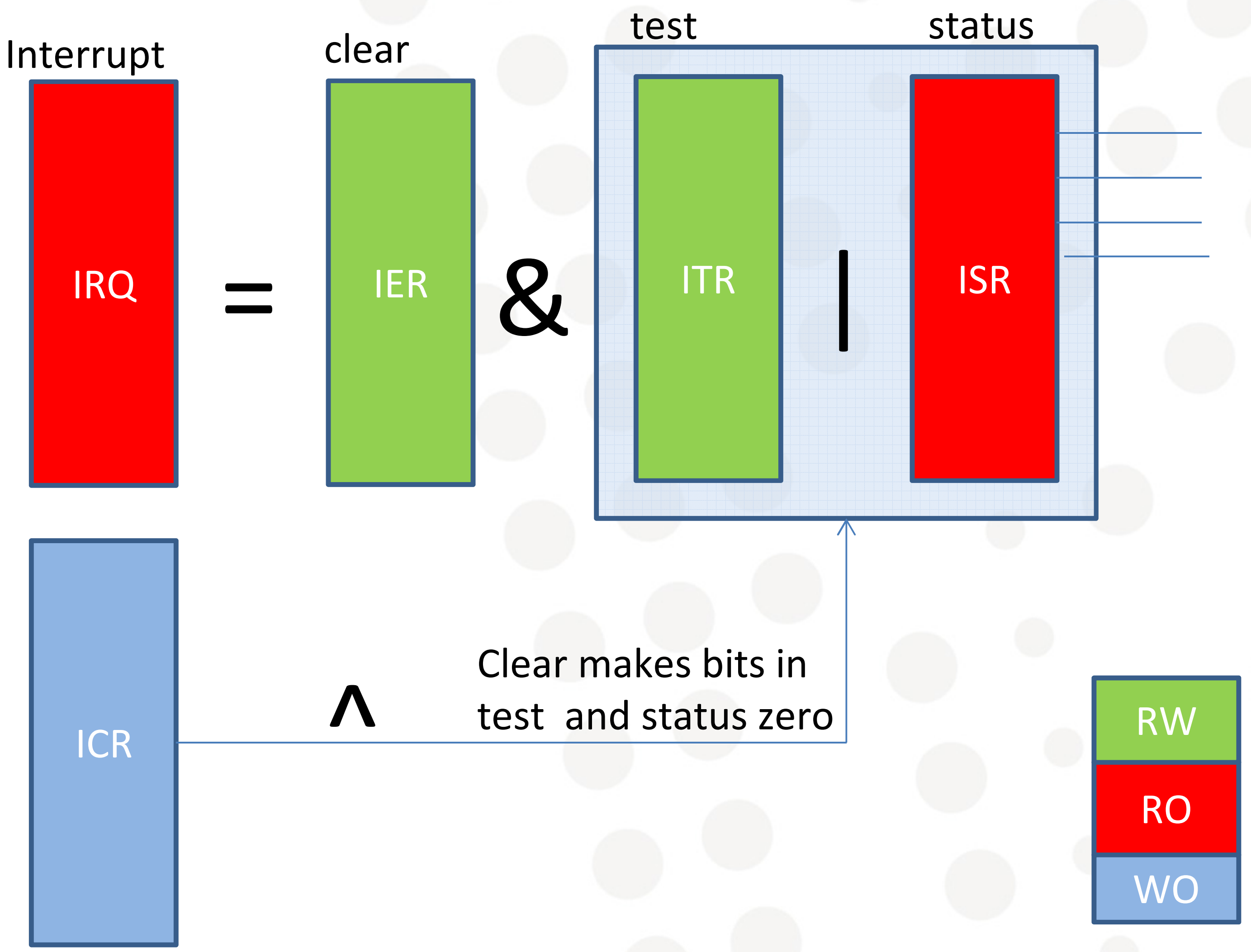
```
map chip_addrmap

include ../case_lib/init.cmd
include ../case_lib/setup.cmd

set stim_run_str 0
set check_run_str 0

set stim_rst 1
set check_rst 1
run -c 1
set stim_rst 0
set check_rst 0

set stim_run_str 1
run -c 1
set stim_run_str 0

wait4 wrcnt 00000100
set check_run_str 1
run -c 1
set check_run_str 0

wait4 check_complete 1
check check_result 1
if
  report -n Verific. 1 completed without errors.
else
  report -n Verific. 1 completed with ERRORS.
end
```

```
NOTE: Invalid probe signals in testbench:     0
    0 Start of simulation
NOTE: Invalid addresses in address map package:     0
NOTE: Invalid data in address map package:     5
    0   NOTE: Open file: ../case_lib/init.cmd
    0 Set signal stim_rst to value: 0
    0 Set signal check_rst to value: 0
    0 Set signal stim_run_str to value: 0
    0 Set signal check_run_str to value: 0
    0   NOTE: Init done
    0   NOTE: Close file: ../case_lib/init.cmd
    0   NOTE: Open file: ../case_lib/setup.cmd
    0 Set signal fpga_rst_n to value: 0
   10 Set signal fpga_rst_n to value: 1
   20   NOTE: Reset done
   20   NOTE: Close file: ../case_lib/setup.cmd
   20 Set signal stim_run_str to value: 0
   20 Set signal check_run_str to value: 0
   20 Set signal stim_rst to value: 1
   20 Set signal check_rst to value: 1
   21 Set signal stim_rst to value: 0
   21 Set signal check_rst to value: 0
   21 Set signal stim_run_str to value: 1
   22 Set signal stim_run_str to value: 0
   22 Waiting on signal wrcnt to get value: 00000100 ....
   61 Waiting on signal wrcnt completed. Got value: 00000100
   61 Set signal check_run_str to value: 1
   62 Set signal check_run_str to value: 0
   62 Waiting on signal check_complete to get value: 1 ....
  109 Waiting on signal check_complete completed. Got value: 1
  109 Checked signal check_result with value: 1 and got the expected
       value: 1
  109   NOTE: Verific. 1 completed without errors.
```



Messages	
/tb_top/stim_d	0
/tb_top/check_d	4
/tb_top/we	0
/tb_top/re	0
/tb_top/full	0
/tb_top/empty	1
/tb_top/rdcnt	0
/tb_top/wrcnt	0
/tb_top/stim_rst	0
/tb_top/stim_run_str	0
/tb_top/check_rst	0
/tb_top/check_run_str	0
/tb_top/check_complete	1
/tb_top/check_result	1
uut/d	0
uut/q	4
uut/rdcnt	0
uut/wrcnt	0
uut/dout	0000
uut/dout_ena	0
uut/advn_ale_i	1
uut/oe_i_n	1
uut/miso_i	0
uut/wait_i_n	1
uut/irq_n	1
uut/empty	1
uut/full	0
Now	?50 ns
Cursor 1	?98 ns

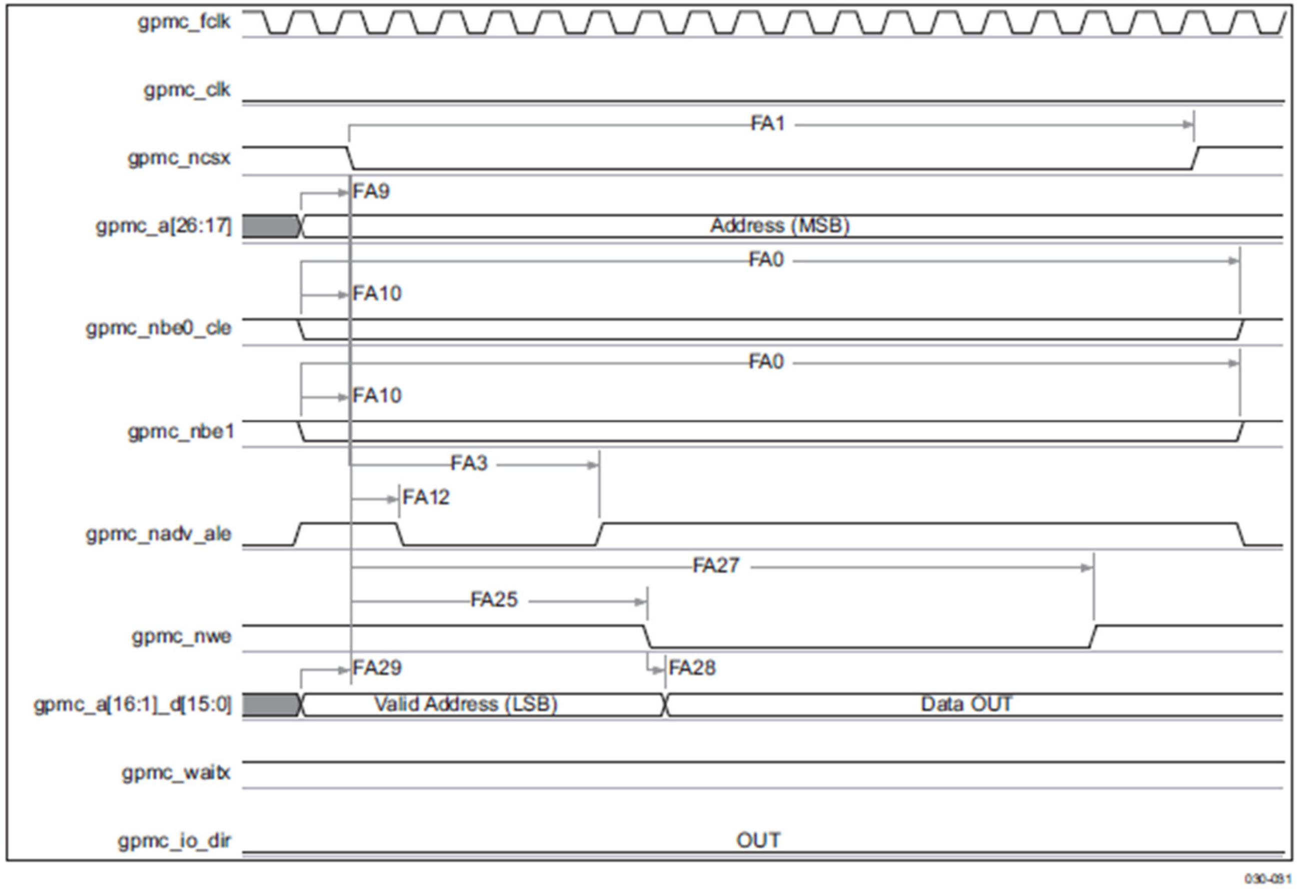
```
package chip_pck is

  :
  :

  -- PIF register adresses
  constant PIF_IRQ            : reg_addr := x"0001000"; -- RO U16,16
  constant PIF_IER            : reg_addr := x"0001004"; -- RW U16,16
  constant PIF_ITR            : reg_addr := x"0001008"; -- RW U16,16
  constant PIF_ISR            : reg_addr := x"000100C"; -- RO U16,16
  constant PIF_ICR            : reg_addr := x"0001010"; -- WO U16,16

  constant PIF_FPGA_DUMMY     : reg_addr := x"0001020"; -- RW U16,16

  :
  :

  -- REG register addresses
  constant P_TEST_REG_IRQ     : reg_addr := x"00F0000"; -- RO U16,16
  constant P_TEST_REG_IER     : reg_addr := x"00F0004"; -- RW U16,16
  constant P_TEST_REG_ITR     : reg_addr := x"00F0008"; -- RW U16,16
  constant P_TEST_REG_ISR     : reg_addr := x"00F000C"; -- RO U16,16
  constant P_TEST_REG_ICR     : reg_addr := x"00F0010"; -- WO U16,16

  constant P_TEST_REG_CTRL    : reg_addr := x"00F0020"; -- RW U8,8
  constant P_TEST_REG_ENA     : reg_addr := x"00F0021"; -- RW U8,8

end chip_pck;
```



Clear makes bits in test and status zero

```
pmw -h PIF_IER x'8000
pmc -h PIF_IER x'8000

pmw -b P_TEST_REG_ENA  x'5A
pmc -b P_TEST_REG_ENA  x'5A
pmw -b P_TEST_REG_CTRL x'3B
pmc -b P_TEST_REG_CTRL x'3B

pmw -h P_TEST_REG_IER x'0001
pmc -h P_TEST_REG_IER x'0001
pmw -h P_TEST_REG_ITR x'8001
pmc -h P_TEST_REG_ITR x'8001
wait4 irq_n 0
pmc -h P_TEST_REG_ISR x'8001
pmc -h P_TEST_REG_IRQ x'0001
pmw -h P_TEST_REG_ICR x'0001
wait4 irq_n 1
pmc -h P_TEST_REG_IRQ x'0000

pmw -h P_TEST_REG_IER x'8000
pmc -h P_TEST_REG_IER x'8000
wait4 irq_n 0
pmc -h P_TEST_REG_ISR x'8000
pmc -h P_TEST_REG_IRQ x'8000
pmw -h P_TEST_REG_ICR x'8000
wait4 irq_n 1
pmc -h P_TEST_REG_IRQ x'0000

report -n Reg. and irq. verif. completed
```

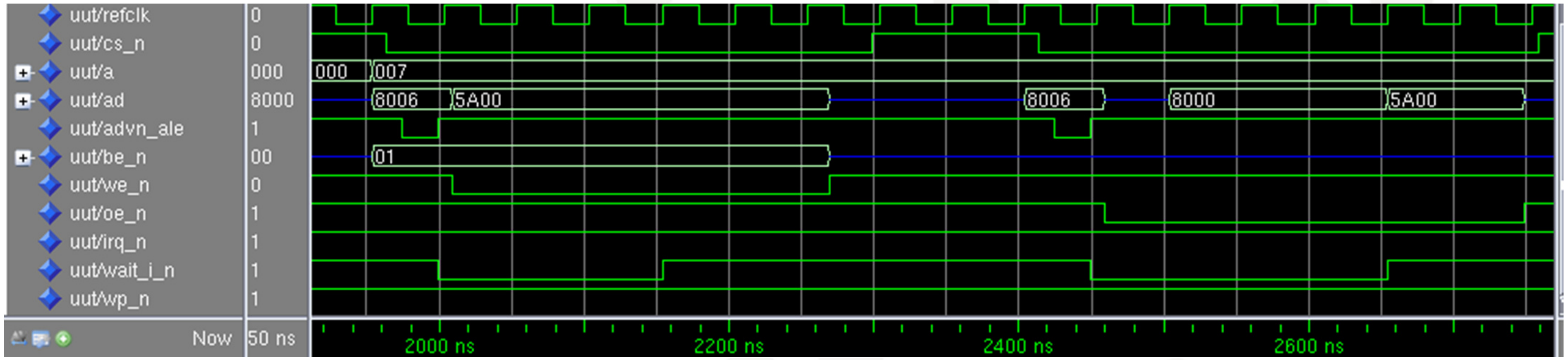
```
  :
  0 Set signal fpga_rst_n to value: 0
 10 Set signal fpga_rst_n to value: 1
  :
 21 Pif mP halfword write to   UUT @ 0001004<= 8000
 36 Pif mP halfword read  from UUT @ 0001004<= 8000
 40 Pif mP MSB byte write to   UUT @ 00F000D<= 5A00
 55 Pif mP MSB byte read  from UUT @ 00F000D<= 5A00
 59 Pif mP LSB byte write to   UUT @ 00F000C<= 003B
 74 Pif mP LSB byte read  from UUT @ 00F000C<= 5A3B
 78 Pif mP halfword write to   UUT @ 00F0004<= 0001
 93 Pif mP halfword read  from UUT @ 00F0004<= 0001
 97 Pif mP halfword write to   UUT @ 00F0008<= 0001
112 Pif mP halfword read  from UUT @ 00F0008<= 0001
115 Waiting on signal irq_n to get value: 0 ....
115 Waiting on signal irq_n completed. Got value: 0
116 Pif mP halfword write to   UUT @ 00F0008<= 0000
124 Waiting on signal irq_n to get value: 1 ....
124 Waiting on signal irq_n completed. Got value: 1
125 Pif mP halfword write to   UUT @ 00F0004<= 8000
140 Pif mP halfword read  from UUT @ 00F0004<= 8000
144 Pif mP halfword write to   UUT @ 00F0008<= 8000
159 Pif mP halfword read  from UUT @ 00F0008<= 8000
162 Waiting on signal irq_n to get value: 0 ....
162 Waiting on signal irq_n completed. Got value: 0
163 Pif mP halfword write to   UUT @ 00F0008<= 0000
171 Waiting on signal irq_n to get value: 1 ....
171 Waiting on signal irq_n completed. Got value: 1
171   NOTE: Reg. and irq. verif. completed
171   NOTE: Close file: reg_irq_rw.cmd
171   NOTE: TEST_REG verific. completed
186   NOTE: Simulation successful!
```



In gpmc_ncsx, x is equal to 0, 1, 2, 3, 4, 5, 6, or 7. In gpmc_waitx, x is equal to 0, 1, 2, or 3.

Figure 6-12. GPMC/Multiplexed NOR Flash – Asynchronous Write – Single Word Timing

- Processor write and read BFM operation



```
procedure PifWrite (
    constant tsize     : in  transaction_size;                -- Halfword/Upper/Lower size
    constant addr      : in  std_logic_vector(27 downto 0);
    constant data      : in  std_logic_vector(15 downto 0);
    signal   gpmc_fclk : in  std_logic;
    signal   gpmc_a    : out std_logic_vector(10 downto 1);   -- connect to a
    signal   gpmc_d    : inout std_logic_vector(15 downto 0); -- connect to cpu_d
    signal   gpmc_be_n : out std_logic_vector(1 downto 0);
    signal   gpmc_cs_n : out std_logic;
    signal   gpmc_ale  : out std_logic;
    signal   gpmc_we_n : out std_logic;
    signal   gpmc_wait_n : in  std_logic;
    file     log       :     text;
    constant cycle     : in  natural ) is
  begin

    wait until rising_edge(gpmc_fclk);

    :
    :
    :


  end procedure PifWrite;
```

```
gpmc_a    <= addr(26 downto 17);
gpmc_d    <= addr(16 downto 1);

gpmc_be_n <= "11";
if (tsize=BYTE and addr(0)='0') then
  gpmc_be_n(0) <= '0';
  writef(log, cycle,"Pif mP LSB byte write to UUT" &
                     " @ " & lv2strx(addr,27) &
                     "<= " & lv2strx(data,16));
end if;
if (tsize=BYTE and addr(0)='1') then
  gpmc_be_n(1) <= '0';
  writef(log, cycle,"Pif mP MSB byte write to   UUT" &
                    " @ " & lv2strx(addr,27) &
                    "<= " & lv2strx(data,16));
end if;
if tsize=HALFWORD then
  gpmc_be_n <= "00";
  writef(log, cycle,"Pif mP halfword write to   UUT" &
                    " @ " & lv2strx(addr,27) &
                    "<= " & lv2strx(data,16));
end if;

gpmc_ale  <= '1';
gpmc_we_n <= '1';
gpmc_cs_n <= '0' after 10 ns;
```

```
  wait for 20 ns;
  gpmc_ale  <= '0';

  wait for 25 ns;
  gpmc_ale  <= '1';

  wait for 10 ns;
  gpmc_d    <= data;
  gpmc_we_n <= '0';

  -- wait until rising_edge(gpmc_wait_n);
  wait for 260 ns;

  gpmc_we_n <= '1';
  gpmc_d <= (others => 'Z');
  gpmc_be_n <= "ZZ";

  wait for 30 ns;
  gpmc_cs_n <= '1';

  wait until falling_edge(gpmc_fclk);

end procedure PifWrite;
```