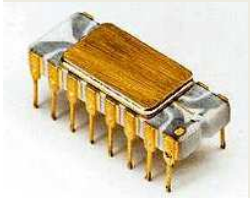
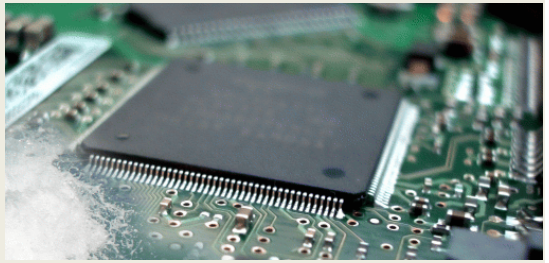




GHDL, a VHDL compiler

Una herramienta *opensource* para analizar y simular código VHDL

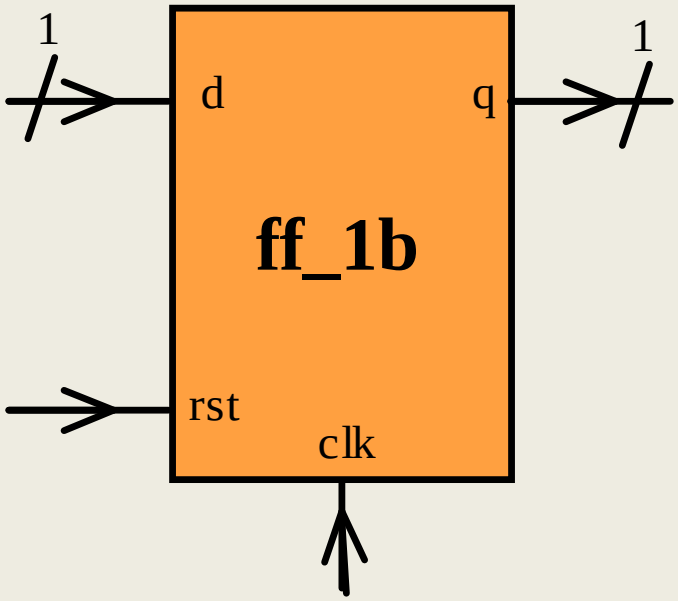
Esp. Eduardo A. Sanchez

esanchez@famaf.unc.edu.ar

Que es *GHDL*?

- Es un compilador VHDL que puede ejecutar “casi” cualquier programa VHDL.
- No es una herramienta de síntesis (Xilinx ISE, Altera Quartus).
- Traduce directamente los archivos VHDL a código maquina utilizando el *back-end* de GCC.

Ejemplo: Registro de un bit




```
library ieee;
use ieee.std_logic_1164.all;
```

```
entity ff_1b is
port (
    q       :out std_logic;
    d       :in  std_logic;
    clk :in  std_logic;
    rst :in  std_logic
);
end entity;
```

```
architecture behavior of ff_1b is
begin
process (clk, rst) begin
    if (rst = '1') then
        q <= '0';
    elsif (clk'event and clk='1') then
        q <= d;
    end if;
end process;
end architecture;
```


esanchez@famaf.unc.edu.ar

Ejemplo: Registro de un bit (cont.)

```
library ieee;
use ieee.std_logic_1164.all;

entity ff_1b_tb is
end entity;

architecture TB of ff_1b_tb is
    component ff_1b
    port (
        q           :out std_logic;
        d, clk, rst :in  std_logic;
    );
    end component;
    signal q, d, rst, clk:    std_logic;
```

```
begin
    dut: ff_1b port map (q=>q, d=>d, clk=>clk, rst=>rst);
    process
    begin
        clk <= '0';
        wait for 5 ns;
        clk <= '1';
        wait for 5 ns;
    end process;

    process
    begin
        rst <= '1';
        wait for 10 ns;
        rst <= '0';
        d <= '1';
        wait for 10 ns;
            d <= '0';
            wait for 10 ns;
            d <= '1';
            rst <= '1';
            wait for 100 ns;
    end process;
end;
```


esanchez@famaf.unc.edu.ar

Pasos para correr un *testbench*

- Analizar el archivo de diseño:

  $ *ghdl -a ff_1b.vhdl*
- Analizar el archivo *testbench*:

  $ *ghdl -a ff_1b_tb.vhdl*
- Construir el archivo ejecutable (“*elaborate*”):

  $ *ghdl -e ff_1b_tb*
- Correr el *testbench*:

  $ *ghdl -r ff_1b_tb*

GTKWAVE

- Correr el *testbench y guardar un dump de la forma de onda*:

  $ *ghdl -r ff_1b_tb*

  --stop-time=40ns

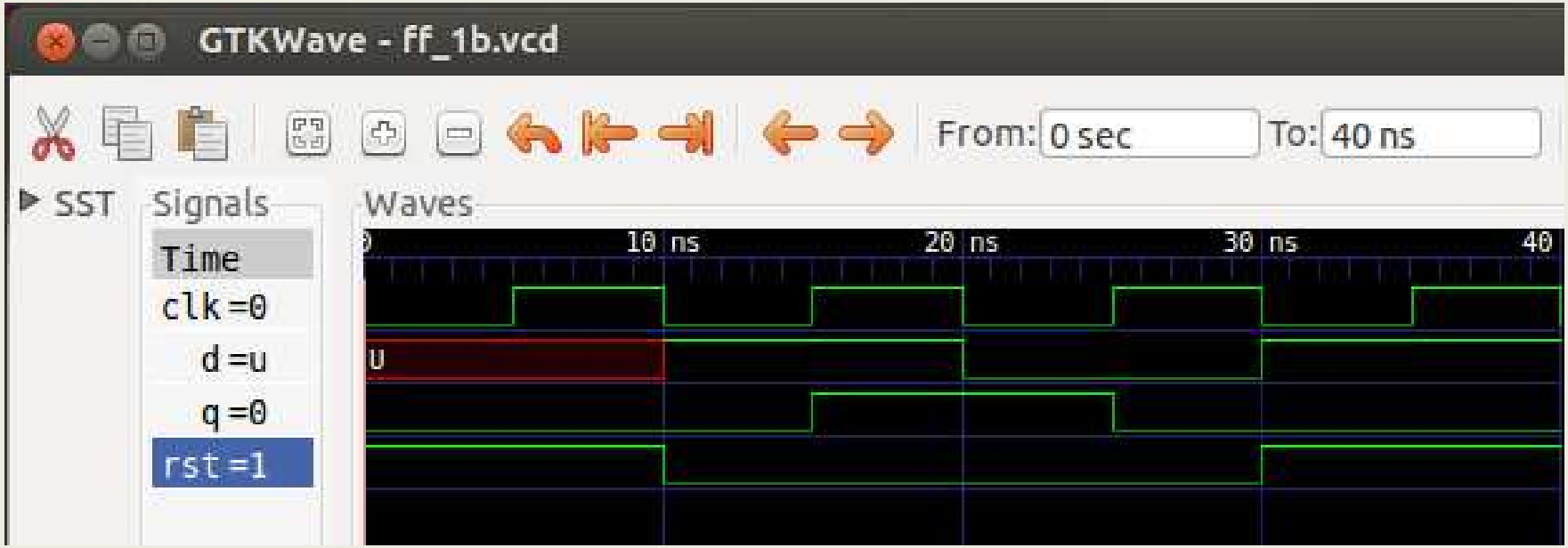
  --vcd =ff_1b.vcd

- Ver la forma de onda generada con el visor *gtkwave*:

  $ *gtkwave ff_1b.vcd*

esanchez@famaf.unc.edu.ar

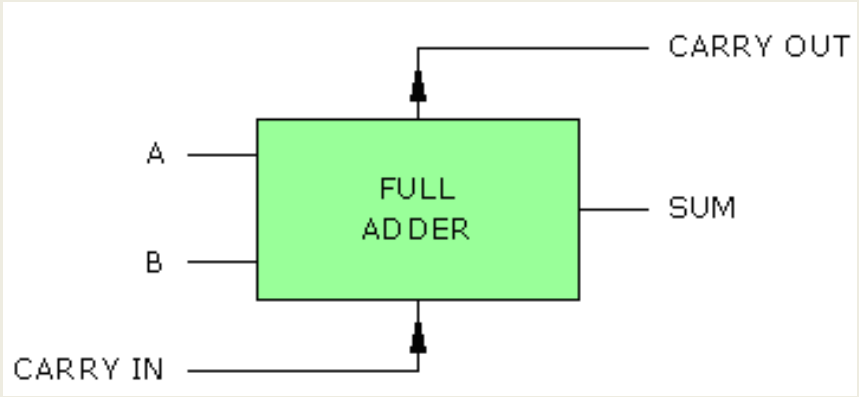
GTKWAVE (Cont.)



esanchez@famaf.unc.edu.ar

Mas ejemplos…

- Full Adder 1bit:



$S = A \oplus [B \oplus C_{IN}]$

$C_{OUT} = BC_{IN} + AC_{IN} + AB$

Full Adder Truth Table

CARRY IN	*input B*	*input A*	*CARRY OUT*	*SUM digit*
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

esanchez@famaf.unc.edu.ar

Mas ejemplos (cont.)

```
library ieee;
use ieee.std_logic_1164.all;

entity adder is
    port (a, b : in std_logic;
          ci : in std_logic;
          s : out std_logic;
          co : out std_logic);
end adder;

architecture rtl of adder is
begin
    s <= a xor b xor ci;
    co <= (a and b) or (a and ci) or (b and ci);
end rtl;
```

http://home.gna.org/ghdl/ghdl/A-full-adder.html#A-full-adder

esanchez@famaf.unc.edu.ar

Mas ejemplos (cont.)

```
begin
  --  Instanciamos el componente.
  adder_0: adder port map (a => a, b => b, ci => ci, s => s, co => co);
  --  Este process hace realmente el trabajo.
 process
     type pattern_type is record
       a, b, ci,s, co : std_logic;
     end record;
   --  Los patrones a aplicar.
     type pattern_array is array (natural range <>) of pattern_type;
     constant patterns : pattern_array :=(('0', '0', '0', '0', '0'), ('0', '0', '1', '1', '0'),
                                          ('0', '1', '0', '1', '0'), ('0', '1', '1', '0', '1'),
                                          ('1', '0', '0', '1', '0'),('1', '0', '1', '0', '1'),
                                          ('1', '1', '0', '0', '1'), ('1', '1', '1', '1', '1'));
     begin
         for i in patterns'range loop  --  Chequeamos cada patrón.
                a <= patterns(i).a; --  Seteamos las entradas.
                b <= patterns(i).b;
                ci <= patterns(i).ci;
                wait for 1 ns; --  Esperamos por los resultados.
                assert s = patterns(i).s --  Chequeamos las salidas.
                    report "resultado de la suma erróneo" severity error;
               assert co = patterns(i).co
                    report "acarreo de salida erróneo" severity error;
          end loop;
          assert false report "fin del test" severity note;
          --  Esperamos por siempre; esto terminará la simulación.
          wait;
    end process;
end behav;
```

http://home.gna.org/ghdl/ghdl/A-full-adder.html#A-full-adder

GHDL, 32 bits corriendo en 64 bits

- Debemos agregar algunas opciones:

  $ *ghdl -a -Wc,-m32 -Wa,--32 ff_1b.vhdl*

  $ *ghdl -a -Wc,-m32 -Wa,--32 ff_1b_tb.vhdl*

  $ *ghdl -e -Wa,--32 -Wl,-m32 ff_1b_tb*

  $ *ghdl -r ff_1b_tb --stop_time=40ns --vcd =ff_1b.vcd*

  $ *gtkwave ff_1b.vcd*

http://www.cyberciti.biz/tips/compile-32bit-application-using-gcc-64-bit-linux.html
http://www.mail-archive.com/ghdl-discuss@gna.org/msg00755.html

Recursos

- Ghdl: http://ghdl.free.fr/
- Manual:
  - http://home.gna.org/ghdl/ghdl/index.html
- Tutoriales:
  - http://mbmn.net/uer/tutorials/vhdl-with-ghdl/
  - http://sluc.org.ar/node/13
  - http://home.gna.org/ghdl/ghdl/A-full-adder.html#A-full-adder
- Opciones (i.e *std_logic_unsigned*):
  - http://ghdl.free.fr/ghdl/GHDL-options.html

esanchez@famaf.unc.edu.ar