**Testbench:**

The testbench only instantiates the DLX, RAM and ROM components and connects them together. It also generates the global clock and reset signals.

**DLX:**

port map ( CLK, RST, IRAM\_ADDRESS, IRAM\_ENABLE, IRAM\_READY, IRAM\_DATA, DRAM\_ADDRESS, DRAM\_ENABLE, DRAM\_READNOTWRITE, DRAM\_READY, DRAM\_DATA ) ;

**Read-only RAM:**

The read-only ram works in two different ways, depending on the value of the RST signal.

generic (

FILE\_PATH : string ; −− ROM data file

ENTRIES : integer := 128; −− Number of lines in the ROM

WORD\_SIZE : integer := 32; −− Number of bits per word

DATA\_DELAY : natural := 2 −− Delay ( in # of clock cycles ) ) ;

port (

CLK : in std\_logic ;

RST : in std\_logic ;

ADDRESS : in std\_logic\_ vector (WORD\_SIZE − 1 downto 0 ) ;

ENABLE : in std\_logic ;

DATA\_READY : out std\_logic ;

DATA : out std\_logic\_vector ( 2∗WORD\_SIZE − 1 downto 0 ) ) ;

If RST is high, than the ROM refreshes its internal buffer contents with the ones in the dump file, whose path is to be set in the FILE\_PATH variable.

If RST is low, than depending on the value of the ENABLE signal, the ROM is either ready to accept incoming requests ( ENABLE high ) or simply remains in its idle state ( ENABLE low ).

Usage: In order to send a request to the memory, we must drive the signals as follows:

• CLK -> Clock of the memory. Can be equal to the one used by the uP.

• RST -> Low

• ENABLE -> High

• ADDRESS -> The address is a vector of WORD\_SIZE bits, representing the index of the word to be read. It has to be stable until the DATA\_READY signal becomes high. After DATA\_DELAY clock cycles, the ROM will drive DATA\_READY high and place the data in the bus named DATA.

**Read-write RAM:**

generic (

FILE\_PATH: string ; −− RAM out put data file

FILE\_PATH\_INIT: string ; −− RAM initialization data file

WORD\_SIZE: natural := 32; −− Number of bits per word

ENTRIES: natural := 128; −− Number of lines in the ROM

DATA\_DELAY: n a t u r al := 2 −− Delay ( in # of clock cycles ) ) ;

port (

CLK : in std\_logic ;

RST : in std\_logic ;

ADDRESS : in std\_logic\_ vector (WORD\_SIZE − 1 downto 0 ) ;

ENABLE : in std\_logic ;

READNOTWRITE : in std\_logic ;

DATA\_READY : out std\_logic ;

NOUT\_DATA : inout std\_logic\_ vector ( 2∗WORD\_SIZE−1 downto 0 ) ) ;

The type of operation to be performed in controlled by the READNOTWRITE signal, which, if high, tells the RAM that we’re interested in reading the memory cell (and its adjacent one, in this current implementation) located at ÆDDRESS; while if low, that we’re interested in writing to those cells with the contents of the bus.