Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Prof. Dr. J. Reichardt
(V2.0 rch / 23.02.12)

Reprogramming Block RAM Memory in Virtex FPGAs

1. Introduction

This document describes how to reprogram Block RAM memory in Virtex FPGAs. The size of each BRAM-block in Virtex devices is 16384 Bits (plus some additional parity bits in some of the Virtex families) which corresponds to 2048 Bytes (address range 0x000… 0x7FF). So you can store up to 1024 16-bit words or 512 32-bit words in a single BRAM block.
The standard ISE design flow requires that the initialised content of Block RAM memory is defined in either the VHDL-code or the UCF file. Consequently it seems to require a new design implementation if the initialised Block RAM (BRAM) memory shall change which is the situation if for example a FPGA microprocessor shall execute a different program. Anyway using the Xilinx data2mem utility it is possible to write the content of BRAM memory with time consuming re-implementation.

2. Files Required

In order to reconfigure the initialised content of Block RAM memory three input files are required:

- The original *.bit file
- a *. bmm (Block Memory MAP) file which describes the Block RAM instance which is to be reinitialised
- a *.mem file which holds the desired memory content.

The last two files can be easily edited using a standard ASCII editor. The mem file can be also automatically generated using the hex2vhdl_v2.0 utility that can be run in a simple command window.

2. 1 The *.mem-File

The file starts with the memory start address in a separate line. Memory data needs to be simply separated by blanks. Memory data will have to be specified with the lowest memory address first. An example for a 256x16 bit memory file which starts at address 0x00 is shown below:

```
@00
4911 4A22 4B33 4C44 4D55 6820 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
4F02 B000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0B70 0000 0000 1374 0000 0000 6840 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
16AC 0000 7005 0F28 0000 0000 0000 6860 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
6120 5A20 0000 0000 6864 1248 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
```

Alternatively you can open a (DOS-) command-window and locate the folder which holds the *.hex file that has been generated by the HASM table assembler.
The hex2vhdl_v2.0.exe requires three parameters to be separated by blanks:

- the name of the hex-file (without file extension)
- depth of memory (can be one of 126, 256, 512, 1024 2048 or 4096 words)
- width of memory (can be either 16 or 32 bit)

A valid example which creates a mem-file with 256 words of 16 bit from a file test.hex is:

```
D:\test_project>hex2vhdl_v2.0 test 256 16
```

(Note that the folder which holds the hex2vhdl_v2.0.exe must be registered, or you will have to copy the *.exe in your project folder)

2.2 The BMM-File:

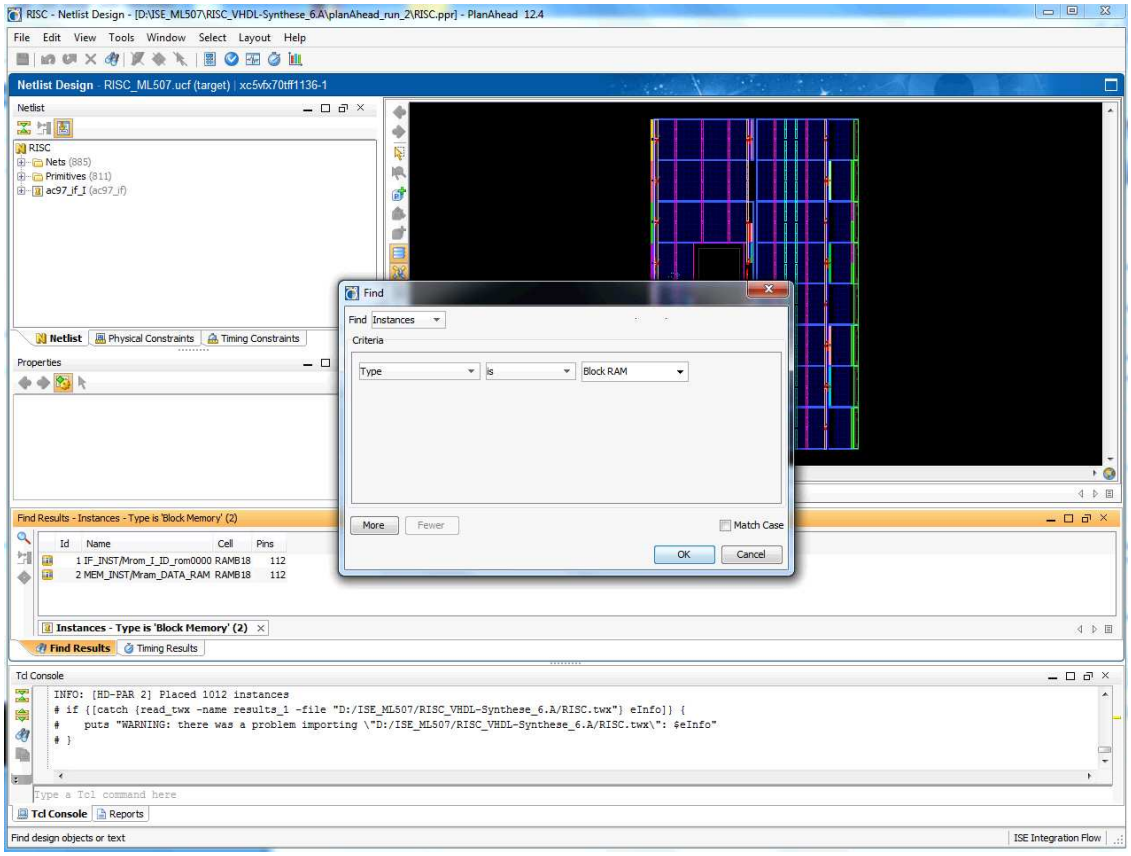
Before you create the BMM File you will have to identify the instance name and especially the location of the BRAM cell within the FPGA. Take the following steps:

1. Inspect the synthesis report generated by XST and search for the desired BRAM instantiation. An example for the instruction ROM of a simple RISC processor is shown below:

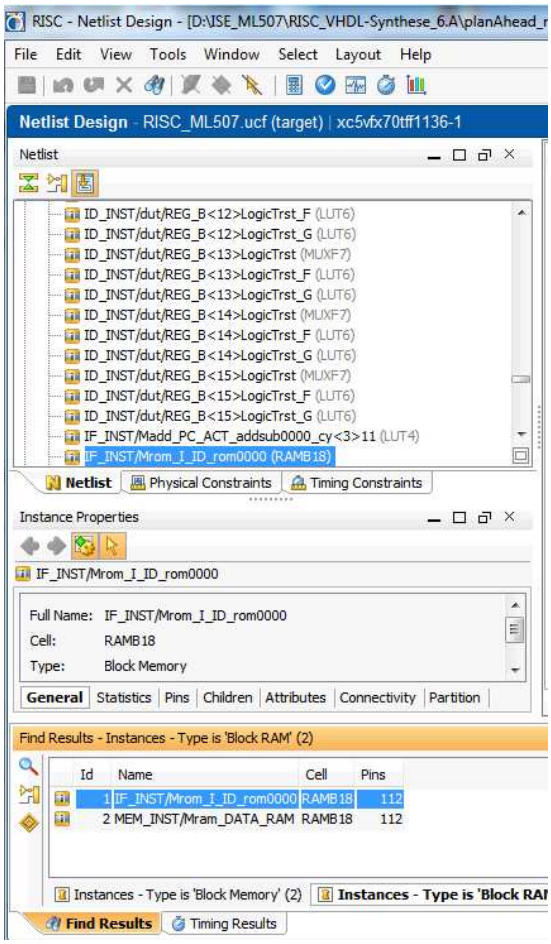
```
Synthesizing (advanced) Unit <IF_PHASE>.
INFO:Xst:3044 - The ROM <Mrom_I_ID_rom0000> will be implemented as a read-only BLOCK RAM, ab-
sorbing the register: <I_ID>.
INFO:Xst:3039 - The RAM <Mrom_I_ID_rom0000> will be implemented as BLOCK RAM
    -----------------------------------------------------------------------
    | ram_type           | Block                                 |          |
    -----------------------------------------------------------------------
    | Port A                                                     |          |
    |     aspect ratio   | 256-word x 16-bit                     |          |
    |     mode           | write-first                           |          |
    |     clkA           | connected to signal <CLK>             | rise     |
    |     weA            | connected to internal node            | high     |
    |     addrA          | connected to signal <PC_ACT>          |          |
    |     diA            | connected to internal node            |          |
    |     doA            | connected to signal <I_ID>            |          |
    -----------------------------------------------------------------------
    | optimization       | speed                                 |          |
    -----------------------------------------------------------------------
```

2. After design implementation you shall now run PlanAhead which is one of the Place & Route Tools of the ISE Design Suite. It can be accessed by double clicking *Analyze Timing / Floorplan Design (PlanAhead).* in the processes window.
Within PlanAhead perform the following steps:

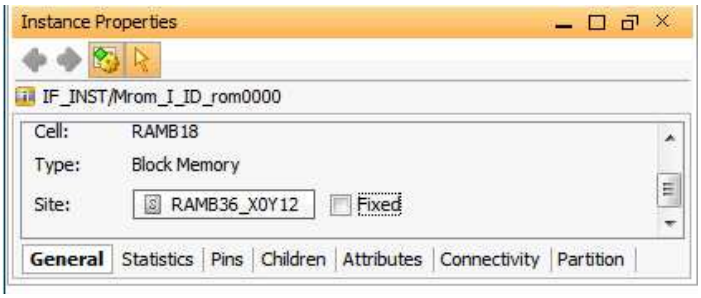
- *Edit->Find Type is Block RAM*
- The "Find Results" window will list all Block RAM instances of the design as shown below



- Now double click the desired `IF_INST/Mrom_I_ID_0000` instance and the "Instance Properties" window will show up as can be seen below:



After scrolling down within the "Instance Properties" window you will find the desired information of the site in which this BRAM is lacated (see below).



The RAMB18 cell is located in site RAMB36_X0Y12 (zero column and 12th row).
As each two RAMB18 blocks fit into a single RAMB36 block the real coordinates for this BRAM are X= 0 and Y=24.

- With this information you might now create and edit a *.bmm file as shown below:

```
// BMM LOC annotation file for RISC processor on ML507 board
// manually edited by J.R.23.02.12
// memory depth: 256, memory width: 16
 ADDRESS_SPACE Mrom_I_ID_rom0000 RAMB16 [0x00:0x7FF]
   BUS_BLOCK
      IF_INST/Mrom_I_ID_rom0000 [15:0] PLACED = X0Y24;
   END_BUS_BLOCK;
 END_ADDRESS_SPACE;
```

Important information within the BMM file are as follows:

- The ADRESS SPACE is 2048 bytes which gives a range of 0x00..0x7FF for the BRAM
- The BUS_BLOCK defines the word access to this BRAM which is 16 bit [15:0] in this example
- The PLACED parameter specifies the column (X) and row (Y) information of the device within the FPGA.

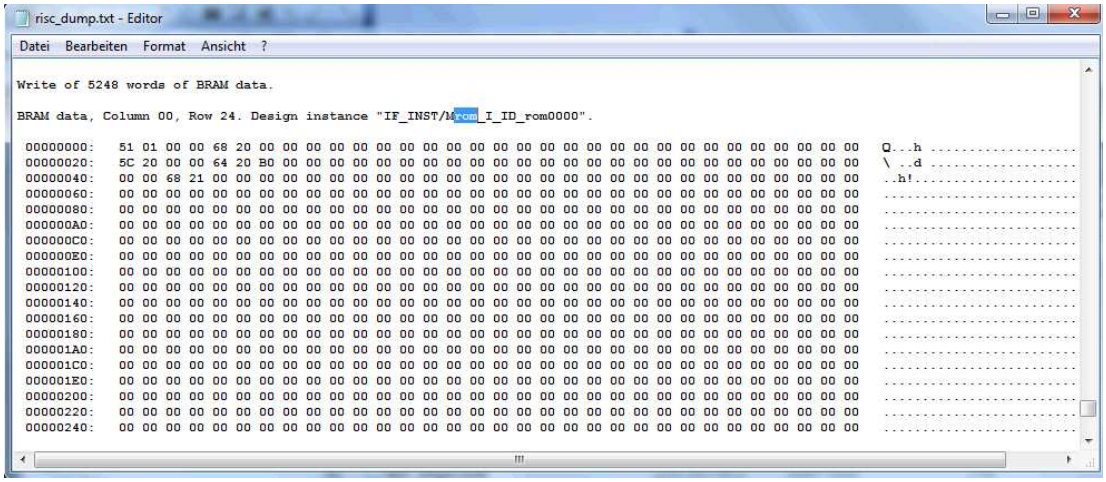
- It is now suggested to dump the original bit-file that has been created by the original design in ISE to see if the specification in the *.bmm is correct. This can be done using the data2mem utility in a command window as follows:

```
set XilinxPath=C:\Xilinx\12.4\ISE_DS\ISE\bin\nt
%XilinxPath%\data2mem -bm risc.bmm -bt risc.bit -d >risc_dump.txt
```

In this example we first set the path to the Xilinx binaries. The second command line holds the following information:

- `-bm`: the bmm file
- `-bt`: the original bit file
- `-d > risc_dump.txt`: the file that makes the content of `risc.bit` readable.

Searching within risc_dump.txt for "Mrom" gives the following information for the content of the BRAM at X=0 and Y=24. It is exactly what was specified in the VHDL-code of the IF_INSTANCE of the RISC processor.



2.3 The *.bit File

Using the data2mem-utility you can now copy the content of the mem-file `ram_mem.mem` to a new bit-file. In the batch script shown below we take the file `risc.bit` as the original and modify the content of *IF_INST/Mrom_I_ID_rom0000* to create a new bit file named `download.bit`.

```
set XilinxPath=C:\Xilinx\12.4\ISE_DS\ISE\bin\nt
%XilinxPath%\data2mem -bm risc.bmm -bt risc.bit -bd ram_mem.mem -o b download.bit
```

Using the dump utility it can be easily demonstrated that the content the BRAM cell in row 24 has been modified. After downloading `download.bit` to the FPGA the processor will run with a different SW program without re-implementation.

3. Conclusion

It has been demonstrated that for a given hardware design the modification of the BRAM content is a relatively simple procedure. Anyway after re-implementation of the design it has to be always checked that the desired BRAM is still in the same position. Eventually the *.bmm file has to be modified.

4. References

[1] www.xilinx.com; Data2mem User Guide UG437