Replacement of a RAM with Atmel FreeRAM™ in Verilog™-based Designs

Features
- Verilog Source Code for FreeRAM Implementation
- Examples for Converting Xilinx® RAM to Atmel FreeRAM

FreeRAM Features
Atmel’s FreeRAM is a versatile component. It can be configured to four different types:

Figure 1. Single-port RAM: Asynchronous or Synchronous

![Schematic of Single-port RAM]

Figure 2. Dual-port RAM: Asynchronous or Synchronous

![Schematic of Dual-port RAM]

The routing resources connecting the RAM are optimized for each mode. For example, the routing nets needed to route the DOUT signals on the dual-port RAM are not used on the single-port RAMs. These unused resources can be used for other routing schemes for another area in the design.
Architectural Differences

Atmel has dedicated RAM blocks inside the FPGA devices. Within every 4 x 4 core cell sector, there is a FreeRAM cell that can be used as 32 x 4 dual-port RAM. This frees up the core cells to be used for logic. Xilinx RAM does not have any separate RAM and must use its CLB to generate them.

Area consumption becomes a major drawback for Xilinx’s RAM. For example, a FIFO design is created that requires the use of a 128 x 16 dual-port RAM. This design would require 128 Xilinx CLBs alone just for the RAM; logic would increase that number. The smallest device required would be a 4005E. If this same design was developed using Atmel’s FreeRAM, the RAM would fit into an AT40K05, but the core cells are not affected by the FreeRAM, allowing additional logic to fit into a small device.

Dual Port Capabilities

Dual-port RAM capability is truly achieved in the Atmel architecture. Xilinx uses only one signal (WE) for this write and output enable. Simultaneous read and write is not possible. By asserting WE high, the RAM can only perform a write operation; disabling WE performs a read operation. Atmel RAM implements separate signals for the write and output enables as well as the data input and output. This allows simultaneous read and write operations.

Modifying Verilog Codes to Implement Atmel RAM

Figure 3 shows how a Xilinx RAM is implemented in Verilog. Figure 4 shows how that same design is converted to implement Atmel RAM. It is very important to remember that the polarity for the write enable in the Atmel RAM is active low. For designs where the write signals were active high, an inverter is necessary to correct the polarity.

During synthesis, the front-end tool will generate a black box for the RAM component. Creation of the RAM block will be generated by the Macro Generator in the IDS software. The RAM will be stored inside a user-defined library. When the design netlist is imported to IDS, the black box definition calls the RAM definition within the user library. The design can be placed and routed after the successful import.

Replacing Xilinx Components in Verilog

It is possible to convert a design using Xilinx RAM into the Atmel architecture without modifying the original Verilog code. A new file needs to be created. Inside this file is the entity and architecture declaration of the Xilinx RAM. Under the architecture block section, an Atmel component is called and port mapped to simulate the Xilinx RAM. Figure 5, Figure 6 and Figure 7 show how to implement an Atmel RAM underneath the Xilinx component. Synthesis of the design follows the same procedure previously mentioned.

Importing design into Figaro

Prior to importing the *.edf netlist, the RAM must be generated and stored within a user-defined library. This is achieved by using the Macro Generator tool integrated with the software. After generating the RAM, the macro will be called during the *.edf import.
Replacement of a RAM with Atmel FreeRAM in Verilog

Figure 3. Sample Verilog File Implementing Xilinx RAM

```verilog
module example (clk, wr, data_a, data_b);
    input        clk;
    input        wr;
    input [31:0] data_a;
    input [15:0] data_b;
    wire [3:0] ptra;
    wire [4:0] ptrb;
    wire [15:0] nullnode;
    wire wrblo;
    wire wrbhi;

    assign wrblo = wr & (~ ptrb[0]);
    assign wrbhi = wr & ptrb[0];

dpr16x16 RAMLO(data_a[15:0], nullnode[15:0], data_b[15:0], ptrb[4:1], ptra[3:0], clk, wrblo);

dpr16x16 RAMHI(data_a[31:16], nullnode[15:0], data_b[15:0], ptrb[4:1], ptra[3:0], clk, wrbhi);
.
.
.
endmodule

module dpr16x16 (DPO, SPO, DI, A, DPRA, WR_CLK, WR_EN);
    output [15:0] DPO;
    output [15:0] SPO;
    input  [15:0] DI;
    input  [3:0] A;
    input  [3:0] DPRA;
    input    WR_CLK;
    input    WR_EN;
endmodule
```

Note: 1. SPO is never used in the design. The user must create the dummy signal nullnode and connect it to the component so synthesis will not complain.
Figure 4. Sample Verilog File Implementing Atmel RAM with HDLPlanner\(1)\(2)

```
'define Clockedge posedge
'define setOrReset 'b0
'define setOrResetOn negedge
'define setOrResetLevel 'b0

module SDPR32x8 (AIN, AOUT, DIN, DOUT, WEN, OEN, CLK);

// synopsys template
parameter addr_width = 16;
parameter width = 32;

input [addr_width-1 : 0] AIN;
input [addr_width-1 : 0] AOUT;
input [width-1 : 0] DIN;
output [width-1 : 0] DOUT;
input WEN, OEN, CLK;

endmodule

module RAMBLOCK (wen, clk, din, dout, ain, aout);
input wen, clk;
input [31:0] din;
output [31:0] dout;
input [15:0] ain, aout;
wire oen;
assign oen = 'b0;

// HDLPlanner Instance sdpram_prl
// Do not **DELETE** previous line

parameter InstName_addr_width = 8;
parameter InstName_width = 32;
SDRP32x8 #(InstName_addr_width, InstName_width) RAMLow (.AIN(ain[7:0]), .AOUT(aout[7:0]), .DIN(din), .DOUT(dout), .WEN(wen), .OEN(oen), .CLK(clk) );

SDRP32x8 #(InstName_addr_width, InstName_width) RAMHigh (.AIN(ain[15:8]), .AOUT(aout[15:8]), .DIN(din), .DOUT(dout), .WEN(wen), .OEN(oen), .CLK(clk) );
// Do not **DELETE** next line
// HDLPlanner End Instance sdpram_prl

endmodule
```

Notes: 1. This component will be created using the Macro Generator in IDS within the HDLPlanner. For this example, the following information will be entered:
   – Section: Memory, RAM Dual-port Address Width: 8
   – Width: 32 Ram Type: Synchronous
2. Xilinx RAM uses one signal to control the write and output enables of their RAM. Atmel has separate ports for these controllers. The simplest solution is to always assert the output enable. Xilinx RAM can only perform a write or read but not both. It’s assumed that the data coming out of DOUT will only be sampled when the write enable is disabled. The original design assumes that the write enable is active high. Atmel RAM defines its write enable to be active LOW. Therefore, inverters are needed to correct the signal interface.
Replacement of a RAM with Atmel FreeRAM in Verilog

Figure 5. Xilinx Component with Buried Atmel Component – Example 1

```
module ram32x4 (D, A, WE, O);
input [3:0] D;
input [4:0] A;
input         WE;
output [3:0]  O;

  dpr32x4 U1(.WEN(~WE), .OEN(WE), .AIN(A), .AOUT(A), .DIN(D), .DOUT(O));

endmodule

module dpr32x4 (WEN, OEN, AIN, AOUT, DIN, DOUT);
input    WEN, OEN;
input [4:0] AIN;
input [4:0] AOUT;
input [3:0] DIN;
output [3:0] DOUT;

endmodule
```

![Diagram of Xilinx Component with Buried Atmel Component]

```
ATMEL RAMD

DOUT[3:0] DIN[3:0]

O3 O2 O1 O0

A4 A3 A2 A1 A0

A0

WE

ATMEL RAMD

XILINX RAM32X4
```
module ram32x4s (D, A, WE, WCLK, O);
input [3:0] D;
input [4:0] A;
input WE;
input WCLK;
output [3:0] O;

sdpr32x4 U1(.WEN(~WE), .OEN(WE), .CLK(WCLK), .AIN(A), .AOUT(A), .DIN(D), .DOUT(O));
endmodule

module sdpr32x4 (WEN, OEN, CLK, AIN, AOUT, DIN, DOUT);
input WEN, OEN, CLK;
input [4:0] AIN;
input [4:0] AOUT;
input [3:0] DIN;
output [3:0] DOUT;
endmodule
Replacement of a RAM with Atmel FreeRAM in Verilog

**Figure 7. Xilinx Component with Buried Atmel Component – Example 3**

```
module raml64x4d (D, A, DPRA, WE, WCLK, SPO, DPO);

    input  [3:0] D;
    input  [4:0] A;
    input  [3:0] DPRA;
    input    WE;
    input    WCLK;
    output [3:0] SPO;
    output [3:0] DPO;

    wire    OE;
    assign OE = 'b0;

    sdpr16x4 U1(.WEN(~WE), .OEN(OE), .CLK(WCLK), .AIN(A), .AOUT(A), .DIN(D), .DOUT(SPO));
    sdpr16x4 U2(.WEN(~WE), .OEN(WE), .CLK(WCLK), .AIN(A), .AOUT(DPRA), .DIN(D), .DOUT(DPO));

endmodule

module sdpr16x4 (WEN, OEN, CLK, AIN, AOUT, DIN, DOUT);

    input    WEN, OEN, CLK;
    input [4:0] AIN;
    input [4:0] AOUT;
    input [3:0] DIN;
    output [3:0] DOUT;

endmodule
```
Replacement of a RAM with Atmel FreeRAM in Verilog