FPGA LAB REPORT 8

Hassan Ahmed, Hamza Shabbir, Neha Chaudhary

JULY 28, 2021

Contents

Implementation of counter and memory using ISE IP cores

Objectives:

- Design the RTL of a 4bit up/down counter using ISE IP cores
- Design the RTL of 32byte memory using block RAM & distributed RAM using IP cores.
- Synthesis the code of both Block RAM and distributed RAM

Introduction:

The Spartan-3E family builds on the success of the earlier Spartan-3 family by increasing the amount of logic per I/O, significantly reducing the cost per logic cell. New features improve system performance and reduce the cost of configuration. These Spartan-3E FPGA enhancements, combined with advanced 90 nm process technology, deliver more functionality and bandwidth per dollar than was previously possible, setting new standards in the programmable logic industry.

Because of their exceptionally low cost, Spartan-3E FPGAs are ideally suited to a wide range of consumer electronics applications, including broadband access, home networking, display/projection, and digital television equipment.

IP (Intellectual property) Cores are HDL codes which are built in Xilinx ISE. They can be used by users without getting into details. Since these are intellectual property, the user is unable to see HDL codes of IP cores. These can include functions delivered through the Xilinx CORE Generator™ software, through the Xilinx Architecture Wizard, as standalone archives, from third parties, through Xilinx Platform Studio (XPS), or through System Generator. Xilinx and its partner companies produce IP ranging in complexity from simple arithmetic operators and delay elements to complex system-level building blocks, such as Digital Signal Processing (DSP) filters, multiplexers, transformers, and memory. Xilinx IP is delivered through the following tools and mechanisms.

IP cores fall into one of three categories: hard cores, firm cores, or soft cores. Hard cores are physical manifestations of the IP design. These are best for plug-and-play applications and are less portable and flexible than the other two types of cores. Like the hard cores, firm (sometimes called semi-hard) cores also carry placement data but are configurable to various applications. The most flexible of the three, soft cores exist either as a netlist (a list of the logic gate s and associated interconnections making up an integrated circuit) or hardware description language (HDL) code.

Four-bit up/down Counter:

Four bit Up/Down counter can be implemented using IP cores of Xilinx ISE. It is available in the Basic element section of available cores. To use it, its instantiation model can be seen from the Core generator property of Counter core. The instantiation model as well as test bench to check the functionality of this IP core is given below:

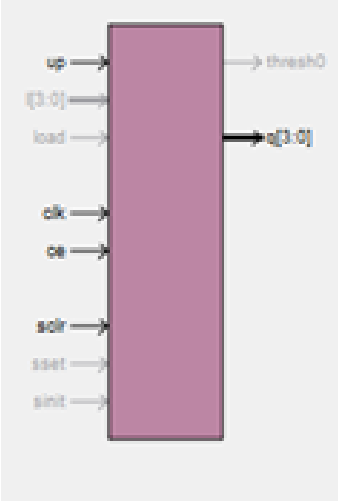

Figure 1: Instantiation Diagram of Up/Down Counter

Test Bench for up/down Instantiation Counter:

```
`timescale 1ns / 1ns
module tb_counter_ip();
wire [7:0]q;
reg up,clk,ce,sclr;
initial begin
clk=1'b0;
ce=1'b1;
up = 1'b1;
sclr=1'b1;
#2 clk=1'b1;
#2 sclr=1'b0;
repeat(100)
#2 clk=~clk;
up=1'b0;
repeat(100)
#2 clk=~clk;
$stop;
end
```

```
Counter_ip c8(.up(up),.clk(clk),.ce(ce),.sclr(sclr),.q(q));
endmodule
```

Simulation Results:

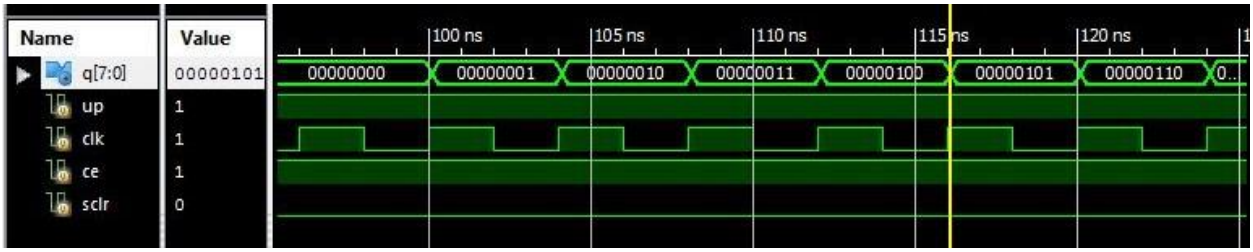

Figure 2: Simulation Result of Up counting

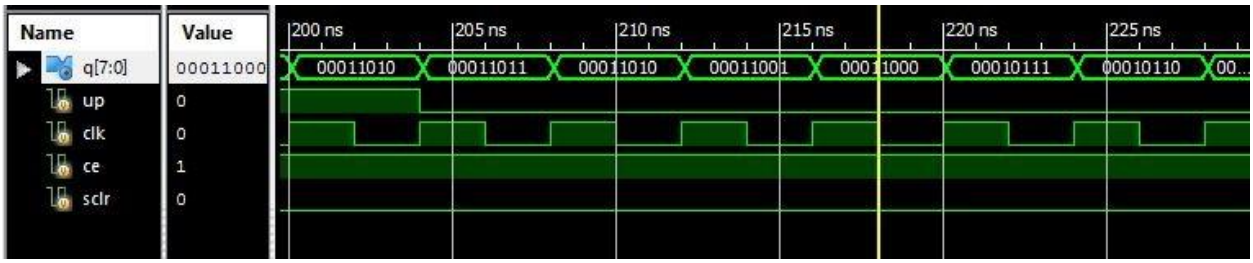

Figure 3: Simulation Result of Down counting

32 Bit Memory using Block Ram:

A dual port block RAM can also be implemented using IP Cores. To use it, its instantiation model can be seen from the Core generator property of Counter core. The instantiation model as well as test bench to check the functionality of Block RAM IP core is given below:

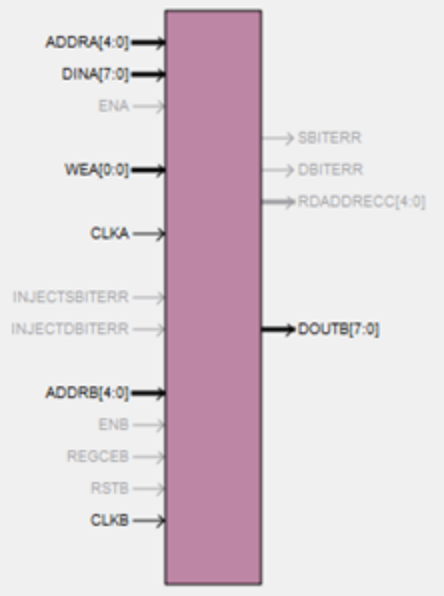

Figure 4: Instantiation Diagram of 32B Block RAM

Test Bench for 32B Block Ram:

```
`timescale 1ns / 1ps
module tb_Block_ram;

	// Inputs
	reg clka;
	reg [0:0] wea;
	reg [4:0] addra;
	reg [7:0] dina;

	// Outputs
	wire [7:0] douta;

	// Instantiate the Unit Under Test (UUT)
	Block_ram_ipcore uut (
		.clka(clka),
		.wea(wea),
		.addra(addra),
		.dina(dina),
		.douta(douta)
	);
integer i;
	initial begin
		// Initialize Inputs
```

```
                clka = 0;
                wea = 1;
                addra = 0;
                dina = 0;
                clka = 0;
for(i=0;i<32;i=i+1)
begin
addra = i;
dina = i;
#1 clka = 1;
#1 clka = 0;
end

for(i=0;i<32;i=i+1)
begin
addra =i;
 $display("%d--------%d", i, addra);
end

        end
endmodule
```

Simulation Results:

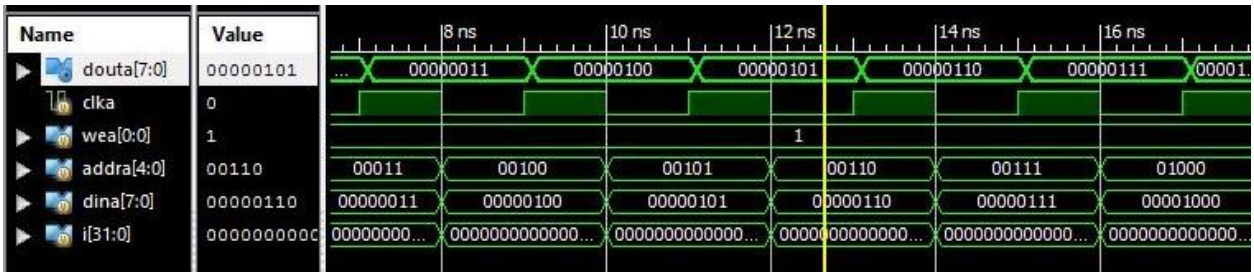

Figure 5: Simulation result of Block RAM

```
Time resolution is 1 ps
# onerror resume
# wave add /
# run 1000 ns
Simulator is doing circuit initialization process.
Block Memory Generator CORE Generator module tb_Bloc
Finished circuit initialization process.
           0-------- 0
           1-------- 1
           2-------- 2
           3-------- 3
           4-------- 4
           5-------- 5
           6-------- 6
           7-------- 7
           8-------- 8
           9-------- 9
          10--------10
          11--------11
          12--------12
          13--------13
          14--------14
          15--------15
          16--------16
          17--------17
```

Figure 6: Console window showing Data in Block RAM

32 Bit Memory using Distributed Ram:

A distributed RAM instance can also be found in the IP cores of XILINX ISE. The depth (data locations) and the width (storage limit per location) can be configured as per requirement. The instantiation model as well as test bench to check the functionality of this IP core is given below:

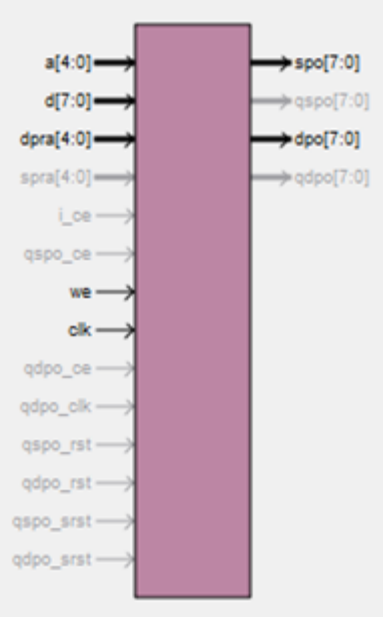

Figure 7: Instantiation Diagram of 32B Distributed RAM

Test Bench for 32B Distributed Ram:

```
`timescale 1ns / 1ps

module tb_d_ram;
// Inputs
        reg [4:0] a;
        reg [7:0] d;
        reg clk;
        reg we;

        // Outputs
        wire [7:0] spo;

        // Instantiate the Unit Under Test (UUT)
        D_ram_ip uut (
                .a(a),
                .d(d),
                .clk(clk),
                .we(we),
                .spo(spo)
        );
```

```
integer i;
        initial begin
                // Initialize Inputs
                a = 0;
                d = 0;
                clk = 0;
                we = 0;
                #2 we = 1;

for(i=0;i<32;i=i+1)
begin
a = i;
d = i;
#1 clk = 1;
#1 clk = 0;
end

$display("\nAddress--------DataStored\n");
for(i=0;i<32;i=i+1)
begin
a =i;
#1 $display("%d----------------%d", i, spo);
end

        end

endmodule
```

Simulation Results:

Figure 8: Simulation result of Distributed RAM

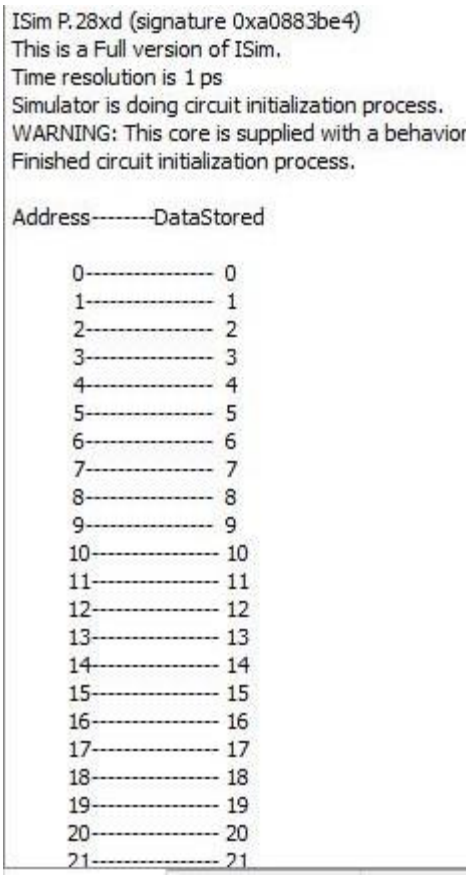

Figure 9: Console window showing data in Distributed RAM

Discussion and Conclusion:

IP Cores are the built in modules in XILINX ISE.

These are the HDL codes on the back end visible to an user in the form of GUI, are easier to use. When the instantiation model is generated the testbenhs are written to test the functionality of the instant

In counter a pin was dedicated for up/down counting (1 for up & 0 for down counting) It also works with the clock & updated on each edge of clock.

A RAM (in the IP Cores) can either be single port or dual port RAM. The single port RAM at a time can either read or write only. While the dual port RAM can read & write both at the same time.

In Block RAM the whole memory is allocated on a chunck (defined hardware memory) while distributed RAM takes the memory at different storage locations. Having two blocks a ~~block~~ Block RAM ~~A can read & write both at once while the second one is for reading only.~~

The setting for block RAM are "read-first", "write-first" and "no change". While there are no such ~~settings~~ settings for distributed RAM. The data is written ~~and~~ and read at the same time.