Designing of 8-bit Synchronous FIFO Memory using Register File

Harish Sharma
I.T.M. University, Gurgaon

Charu Rana
I.T.M. University, Gurgaon

ABSTRACT
FIFO is implies first in first out using queue methodology for memories read and write of any information and data using some control logic. The whole work of FIFO is fully dependent on the control circuitry and clock domain. It is often used to control the flow of data from source to destination by the transition of every clock. Basically FIFO differentiate by clock domain either Synchronous or Asynchronous. There are various methods to designing and synthesized FIFO but here fully focused on the memory which is used to store the data in domain of clock either sync. and async. or single and multiple clock cycles.
This paper will differentiate the design, synthesize and analyze a Synchronous FIFO using Register file memory by older version of Synchronous FIFO. In this paper, conclude the effect of using register file instead of random access memory for storage of data in FIFO memory. This work shows change the parameters like on-chip components (clock, signal, input and outputs etc), clock domain, type of resources, and how to minimize and optimize hierarchy of the device.The RTL description for the FIFO is written using Verilog HDL (hardware description language). And design is simulated and synthesizes in Xilinx ISE Design suit 12.4.The RTL code simulated in ISim Simulator.

Keywords
Verilog, FIFO, RTL, fifo_full, fifo_empty, sync. fifo, async. Fifo, RAM, Register file, read, write.

1. INTRODUCTION
A Synchronous FIFO describes the FIFO design where the data and information is stored in the memory and transition a data in a appropriate fashion using clock pulse. Both read and write operation handle by control circuit. In computer programming, FIFO (first-in, first-out) is an approach to handling program work requests from queues or stacks so that the oldest request is handled first. In hardware it is either an array of flops or Read/Write memory that store data given from one clock domain and on request supplies with the same data to other clock domain following the first in first out logic. Basically FIFO divided in two categories like Synchronous and Asynchronous. In synchronous FIFO, write operation to the FIFO buffer and read operation from a same FIFO buffer are occurring in same clock domain. But in asynchronous FIFO these two operation of write and read to and from respectively FIFO buffer in different clock domain. As clear that the clock domain is different in asynchronous FIFO. Write operation is occurred in one clock domain, and read operation is in another clock domain. In FIFO concept, there is a restrictions for writing any data and read data from FIFO memory. In other way we can say that we can't write and read data without fulfill some appropriate conditions, are called fifo_full and fifo_empty [2]. Here we considered these are two flags which shows status of FIFO condition.

In designing time we fully concern on these two flags. If fifo_full flag is asserted then the user does not write any data in FIFO buffer and for fifo_empty, user does not read any data from FIFO buffer [4]. For assertion of these condition we design pointers to check these conditions. So these pointers are called write and read pointers or we can say,
Firstly Read Pointer/Read Address Register.
Secondly Write Pointer/Write Address Register.
1. FIFO Empty when read address register catches a write address register, the FIFO asserts the Empty signal [4].
2. FIFO FULL when write address register catches a read address register, the FIFO asserts the FULL signal [4].
FIFOs are often used to safely pass data from one clock domain to another asynchronous clock domain FIFOs are used in designs to safely pass multi-bit data words from one clock domain to another. In Async FIFO, Data words are placed into a FIFO buffer memory array by control signals in one clock domain, and the data words are taken from another port of the same FIFO buffer memory array by control signals from a second clock domain. In Sync FIFO, FIFO where writes to, and reads from the FIFO buffer are conducted in the same clock domain.

2. NEED OF THIS RESEARCH
In this my paper I discuss about the FIFO designing using register file instead of RAM for storage and passing the data from source to destination, as well as I discuss the changes in parameters. The effect of using register file, acting much efficient, high speed, less power consumption, and minimize the area of a on-chip components, in a FIFO memory.
Firstly we fully concern on the basic architecture of register file and then after designing verilog code of FIFO, and check test bench, with series of data input till better response is coming. After that checking parameters of that, and compare the FIFO design with the older design of FIFO memory using RAM.

3. DESIGN METHODOLOGY
(A). FIFO memory using "RAM".
(B). FIFO memory using "Register File".
Finally a full concern on architecture and designing of register file as a memory device for FIFO, and details of how we use and application of these.

3.1 Register file
A register file is an array of registers in a central processing unit (CPU). In digital VLSI and computer applications, register files are usually implemented by way of fast SRAMs with multiple ports. Basically registers are storage addressed locations in the processor. CPU instructions operate on these

values directly. On RISC processors, all data must be moved into a register before it can be operated. On CISC (Intel) chips, there are a few operations that can load data from RAM, process it, and save the result back out, but the speed of operations of work by using register file compare then random access memory is fast. In another way the designing and manufacturing cost of "RAM" is much expensive to "REG's". The register files also called as memory array.

In these memory array both write and read operation take place at a same clock domain. Here horizontal and vertical wires for read and write operations or vice-versa. But we have a simple disadvantage with registers that it store 1 slot of bit patterns at a clock cycle, compare then RAM. The basic architecture of register file is shown in fig.1.

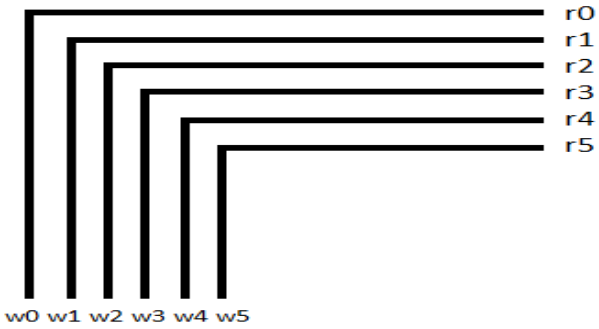

Figure 1: Register file

r0,r1,r2,r3,r4,r5 are read data and
w0,w1,w2,w3,w4,w5 are write data.

The fig 1. Shows array of the memory cell where describes flow of data in memory. We can write data by w0,w1,……Wn, and read data by r0,r1……Rn. So in this array we pass the data safely to the output port or read_port through input port or write_data port.Now above from this description of register file we concern on its designing of RTL code, synthesize and simulation part. In this paper I am presenting the Synchronous FIFO using register file and compare the effects are involved.

4. RTL DIAGRAMS AND SIMULATIONS

In this section, after written the verilog code of fifo using register files, and create rtl diagrams of fifo using "ram" and fifo using "register file". After this, concern on output corresponding to given inputs in xilinx(software) simulator. The name of simulator is ISim. In this simulator we observe test benches. In test bench we get waveforms representation of the given parameters and response of the system according to given parameters.

4.1 RTL diagram of fifo using "ram"

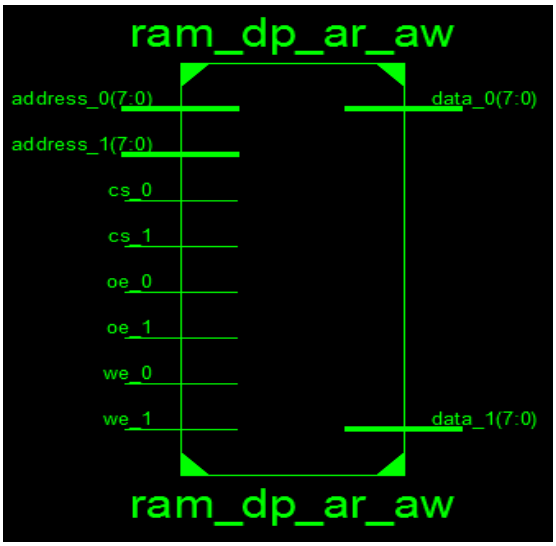

Figure 2: RTL diagram of fifo using ram

In fig 2 "ram_dp_ar_aw(dual port random access memory with asynchronous read and write)" implies the top level module name of FIFO. A RAM and FIFO both are instantiated by this name. here,

1. cs_0,cs_1 are chip selector pins (dual port).
2. oe_0,oe_1 are output enable pins (dual port).
3. we_0,we_1 are write enable pins (dual port).
4. address_0(7:0), address_1(7:0),data_0(7:0) and data_0(7:0) are input and output pins respectively, both are 8 bits.

In this my paper I design synchronous FIFO using register file by verilog code and simulate it, and parameterize the effects.

4.2 Rtl diagram of fifo using "register file"

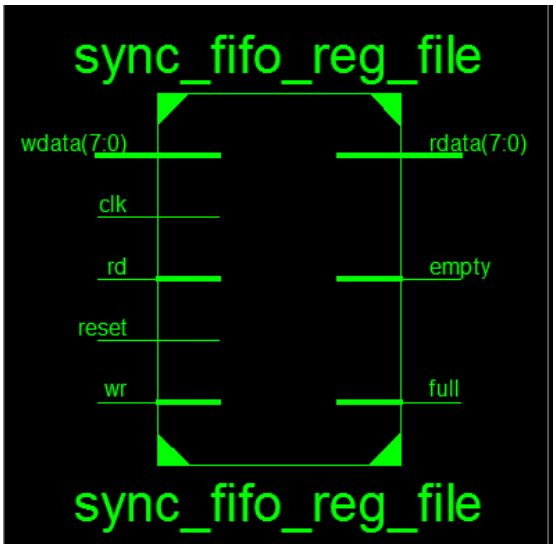

Figure 3: RTL diagram of fifo using register file

In fig 3 "sync_fifo_reg_file(synchronous fifo using register file)" implies the top level module name of FIFO. A REG. and FIFO, both are instantiated by this name. here
1. rd , wr are read enable and write enable.
2. Clk is clock.
3. Empty and full are flags for showing that FIFO is either full and empty.
4. w_data and r_data are input and output ports respectively.

5. TEST BENCH SIMULATION RESULTS

The test bench of this FIFO is shown below in box.

```
initial begin
                        // Initialize Inputs
                        clk = 0;
                        reset = 1;
                        rd = 0;
                        wr = 0;
                        wdata = 0;
                        rd=1'b1;
                        wr=1'b1;
                        #10         reset=1'b0;

 #100       wdata=8'b00000000;
       #100 wdata=8'b00000001;
       #100 wdata=8'b00000010;
       #100 wdata=8'b00000100;
       #100 wdata=8'b00001000;
       #100 wdata=8'b00010000;
       #100 wdata=8'b00100000;
       #100 wdata=8'b01000000;
       #100 wdata=8'b10000000;
       #100 wdata=8'b11111111;
       #1500 $finish;
```

Figure (a): Verilog code of test bench

5.1 TB for write data

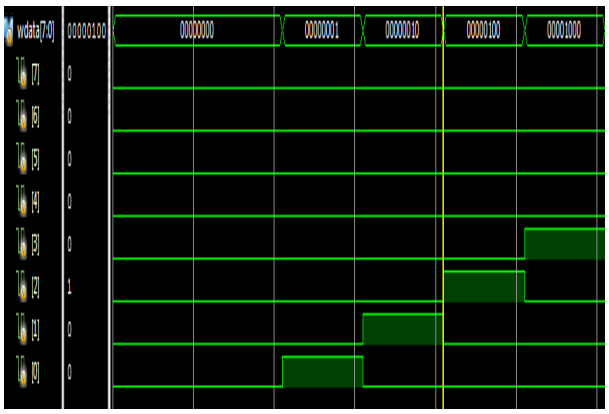

Figure 3: TB for write data

Figure 4 shows tb waveform of the write data. The data is write in fifo memory. In this figure wdata=00000100 taking as a example for write data in fifo memory.

5.2 TB for read data

Figure 4: TB for read data

Figure 5 shows the output waveform. Here the read data is same as the write data in fifo memory. The time and user constraints are same for initial process. But in this waveform write data is read after some delay because of un-stability of output. The output take some time to give the response against write data in fifo memory. But we have a exact output from fifo buffer. Here these fig. 4 and 5 shows test bench of write and read data. Let assume we pass a 8 bit data "00000100" passes from w_data, then after some delay (because of unstability) the data read from the r_data, which is clear in Test Bench. After that we concern on the flags that is empty and full.

5.3 TB for flags

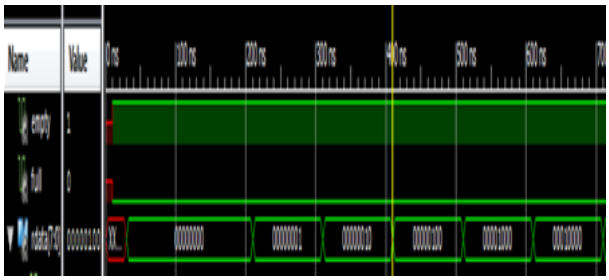

Figure 5: TB for flags

In fig. 6 shows that at the time of reading operation the, EMPTY flag is "**1**", so the FIFO memory is able to read input data till empty flag is "**0**", and FULL flag is "**0**" which mean FIFO memory is not full, when read and write enable are both "**1**" and reset signal falls to "**0**".
So after simulation part there is comparison between FIFO using RAM and REGISTER FILE.

5.4 COMPARISONS

There are different types of categories to compare the fifo design like by clock domain, by hierarchy, by input and outputs, and by resources type.

5.4.1 By clock domain

Table 1. By clock domain

Components	Synchronous-fifo using "ram"	Synchronous-fifo using "register file"
Fan-Out	1916	19
Slice Fan-Out	1008	17

5.4.2 By hierarchy

Table 2. by Hierarchy

Components	Synchronous-fifo using "ram"	Synchronous-fifo using "register file"
Fan-Out	1916	19
Slice Fan-Out	1008	17

5.4.3 By input and outputs

Table 3. By input and output

Components	Synchronous-fifo using "ram"	Synchronous-fifo using "register file"
Fan-out	1916	19
Slice fan-out	1008	17

5.4.4 By recourse type

Table 4. By resource type

On chip	Synchronous-fifo using "ram"			Synchronous-fifo using "register file"		
	Used	Available	Utiliation	Used	Available	Utilization(%
Clock	1	Na	Na	1	Na	Na
Logic	5402	9312	54	37	9312	0
Signal	4680	Na	Na	56	Na	Na
Input-output	38	232	16	32	232	9

6. CONCLUSION

The paper has presented a fifo memory design for multiple read and writes operations in a single clock domain [2], and generating fifo full and empty conditions. The paper has discussed the relevance of fifo in synchronization between input and output data [1]. we have designed, simulated and synthesized a memory using register file for minimize on-chip area, and reduce cost of memory compare then random access memory.

7. REFERENCES

[1] Clifford E. Cummings, "Synthesis and Scripting Techniques for Designing Multi-Asynchronous Clock Designs," *SNUG 2001 (Synopsys Users Group Conference, San Jose, CA, 2001) User Papers*, March 2001, Section MC1, 3rd paper..

[2] Dinesh Tyagi, former CAE Manager for Synopsys DesignWare product, personal communication.

[3] Clifford E. Cummings and Don Mills, "Synchronous Resets? Asynchronous Resets? I am so confused! How will I everknow which to use?," *SNUG 2002 (Synopsys Users Group Conference, San Jose, CA, 2002) User Papers*, March 2002,Section TB2, 1st paper.

[4] Clifford E. Cummings and Peter Alfke, "Simulation and Synthesis Techniques for Asynchronous FIFO Design with Asynchronous Pointer Comparisons," *SNUG 2002 (Synopsys Users Group Conference, San Jose, CA, 2002) User Papers*,March 2002, Section TB2, 3rd paper.