FIFO MEMORY USING SHIFT REGISTER

Mr. AVN HANUMAN[1], Mrs. NAVEEN GEDELA[2]

[1]Associate Professor, International School of Technology and Sciences for Women, Rajanagaram, Andhra Pradesh-533294.

[2]Assistant Professor, International School of Technology and Sciences for Women, Rajanagaram, Andhra Pradesh-533294.

ABSTRACT

Because of flexibility of application and high cost performance, the low-and-middle end FPGA has obtained an extensive market. As a fundamental memory structure, the FIFO memory is widely used in FPGA based project in various manners. But limited by the resources in chip and imperfection of development tools, the problem that the number of memory is insufficient while the overall capacity is enough often occurs in the implementation of multi-channel FIFO. This paper surveys various occasions of applications of multi-channel FIFO and put forward a method to achieve multichannel FIFO in a single FPGA Block RAM, which would support the parallel access to one port. The method may help to solve the problem mentioned above and improve the utilization of storage resources obviously. The steps of implementation and partial source code are present together with the detail analysis of simulation timing. Practical application indicates that the method is successful and effective.

Key words: FIFO, FPGA, XYLINK.

1. INTRODUCTION

The transfer of data over a point-to-point or point-to-multi point communication channel. Examples of such channels are copper wires, optical fibers, wireless communication channels, storage media and computer buses. Earlier days to achieve the high speed we were using parallel/serial transmission. In parallel transmission, Binary data consisting of 1s and 0s which are transmitted in framing/streaming. By grouping, we can send data n bits at a time instead of one. We use n wires to send n bits at one time. That way each bit has its own wire, and all n bits of one group can be transmitted with each clock pulse from one device to another. Form = 8. Typically, the n number of bits are bundled in a frame with a connector at each end that may be 2:4:8:16:32. The advantage of parallel transmission is speed. All else being equal, parallel transmission can increase the transfer speed by a factor of n over serial transmission. Insignificant disadvantage of parallel transmission is cost. In serial transmission one bit follows another, so we need only one communication channel rather than n to transmit data between two communicating devices. The advantage of serial over parallel transmission is that with only one communication channel, serial transmission reduces the cost of transmission

over parallel by roughly a factor of n Since communication within devices is parallel, conversion devices are required at the interface between the sender and the line (parallel-to-serial) and between the line and the receiver (serial-to parallel). Serial transmission occurs in one of two ways; asynchronous or synchronous. With the rapid development of FPGA technology, PFGA is widely used in the field of communication, medical instrument, consumer electronics, display, portable terminal, and so on. Among various levels of FPGA, the low-and middle-end FPGA has won a wide market due to its low cost, high performance, technical maturity and short design cycle. As a fundamental storage structure in the design of system based on FPGA, FIFO is usually used as data buffer of digital signal processing system, Communication Bridge between network of different data rates and communication interface between modules in different clock domain. Sometimes, in practice, it is needed to combine more than one FIFO into a new structure as multi-channel FIFO, according to rule of access to it, multi-channel FIFO generally can be classified into four categories as follows: 1) Serial Input and Serial Output FIFO (SISOFIFO), each channel of SISOFIFO is independent, no parallel operation at read port or write port ispermitted, i.e., the data of the SISOFIFO is written in channel by channel and read out channel by channel. there is no constraint for access among channels; 2) Serial Input and Parallel Output FIFO (SIPOFIFO), just as its name implies, data of the SIPOFIFO is written in channel by channel but read out at the same time; 3) Parallel Input and Serial Output FIFO (PISOFIFO), the data of the PISOFIFO is written in at the same time but read out channel by channel, namely, the write port should be able to be accessed in parallel; 4) Parallel Input and Parallel Output FIFO (PIPOFIFO), the data of the PIPOFIFO is written in simultaneously and read out simultaneously, in other words, both ports of PIPOFIFO should be able to be accessed in parallel.

2. LITERATURE SURVEY

In the previous section we mentioned that NoC is composed of elements like FIFO's (Virtual channels), Interconnects, Arbitration logic etc. The access mechanism in buffer structure plays an important role in the design of NoC applications. In turn, the mechanism influences how efficiently packets share link bandwidth. Buffers are used to store packets or flits when they cannot be forwarded right away onto the output port. Queuing buffers consume the most area and power among building blocks in the NoC [1, 2, 27]. We have investigated related work as follows:

SHORT FIXED LENGTH QUEUE: In short fixed length queue, each physical channel of router has single queue. The arriving flit is stored at the tail of the queue. The flit present at head is read and forwarded to output channel through crossbar. The single queue buffer has fixed and short length, therefore, upstream router keep strack of free buffers. The flit is forwarded to downstream router only when a free buffer is available. The single queue buffer may lead to the problem of HoL blocking.

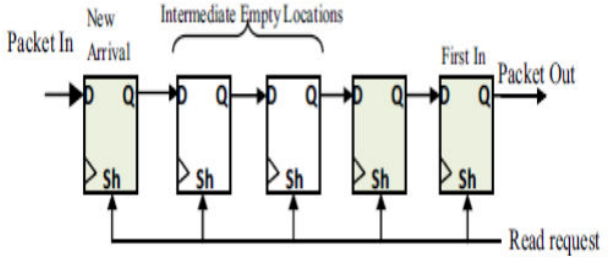
MULTIPLE FIXED LENGTH QUEUE: In multiple fixed length queues, each input physical channel has multiple queues of fixed length that helps in removing the HoL problem to certain extent. The collective representation of queue is called a Virtual Channel. The virtual channel provides multiplexing of

incoming flits at different queues. Thus, physical channel bandwidth sharing becomes efficient in VC based NoC.

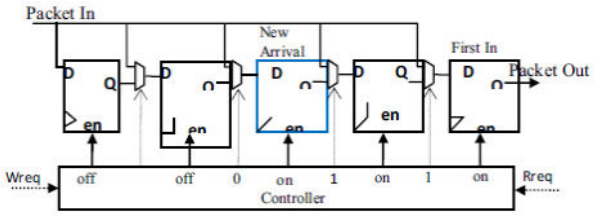
MULTIPLE VARIABLE LENGTH QUEUES: The above discussed buffers are fixed in length. This may lead congestion in the network when there is abnormal traffic pattern. There may be many empty virtual channels that could not be utilized. This situation leads to poor buffer utilization and low throughput. To overcome this problem, a variable length multiple queues have been proposed. This mechanism provides better buffer utilization but at the cost of complex circuitry. The design needs a complex data structure to keep track of head and tail of queue.

Existing system:

DFF-BASED QUEUING BUFFER DESIGN: The queuing buffer can be implement using flip-flop register and SRAM. In this discussion we will present four different design consists of flip-flop registers.



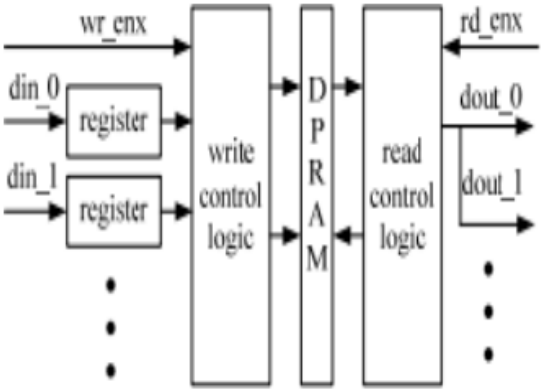
presents a conventional shift-register that is made offlip-flops. The data is written from one end and retrieved at other. The structure helps in synchronization of data between sender and receiver. However, the register can create bubble cells when the packet input and output rates are different. The design is simple but not suitable for NoC communications. To remove the problem of empty bubble, the arrival packet can be stored at the front empty cell rather than at the tail of a queue. "push-in" technique to avoid the bubble problem.



3. PROPOSED NOC CROSSBAR ARCHITECTURE

We propose a new solution for the remaining three kinds of multi-channel FIFO to implement them in one Block RAM. As to SISOFIFO, the channels are independent and there are no timing constraints among them, so it is relatively easy to achieve. In the coming sections, we would concentrate our attention on the implementation of SIPOFIFO and PISIFIFO in one BlockRAM, it can significantly improve utilization of BlockRAM, reduce the cost of product and help to enhance market competitiveness. BASIC FIFO: From the analysis above, we have to implement multi-channel FIFO in one Block RAM and provide write or read operation in parallel to some extent. It's obviously impossible to realize write or read operation at the same time for multiple FIFOs that built in one Block RAM directly. In order to solve the problem of parallel access, it is necessary to place registers into the ports with function of parallel access as data buffer. The width of data buffer should be set in accord with the width of corresponding channel, and the depth is decided by the level of parallel access operation. The general structure of multi-channel FIFO is depicted in Figure 2. It is mainly composed of write control logic, DPRAM, read control logic and input/output

buffering registers. The buffering registers are located at the parallel port, i.e., write port for PISOFIFO, and read port for SIPOFIFO. DPRAM can be instantiated from Block RAM with IP core generator. Simple DPRAM is enough here because there is only one write port and so half of memory capacity can be saved by this means. Each channel have its own private memory space in the DPRAM, all the private memory spaces can't be overlapped. The read control logic and write control logic are relatively complex and there are differences in their structure between the scenes of serial access and parallel access, we'll describe this in detail later.



Parallel Write Control Logic: Parallel write control logic is designed to receive input data of all channels and write them into the DPRAM at corresponding area. The structure of write control logic in parallel is depicted in Figure 3. After receive a parallel write command, i.e. when wr_en x is active, the input data of all channels are registered. On detecting of data coming, the internal write control logic would activate the internal write signals for every channel in turn to fetch the data from registers and write to the DPRAM at corresponding memory area. Here is how it works

Step 1. The control logic enters the idle state when system resets, all the labels are initialized at this time, e.g. p_write_ready, the label of write in parallel getting ready, is set to 1.

Step 2. When wr_enx is active, internal control logic accepts the input data of all channels and then stores them in corresponding buffering registers. p_write_ready is set to 0. A new external write command would not be accepted before p_write_ready is set to 1 again.

Step 3. Set write_in_process to 1, which means the control logic is enter the procedure of carrying data from buffering registers to DPRAM.

Step 4. Internal control logic generates the internal write command vector wr_eni, this causes a internal write operation for each channel, take channel 0 for example, the internal write operation goes as follows:

Step 5. Number of write channel is decided by ch_write, it is set to 0 firstly, i.e., the first channel is selected;

Step6. The write address of DPRAM wr_addr is a combination of ch_write and write_p_0, i.e., wr_addr={ch_write,write_p_0}, where write_p_0 means the write address pointer of channel 0.

Step 7. The LSB of the internal write commands vector wr_eni is set to 1, and the input data of channel 0 is written to corresponding memory area in DPRAM ;

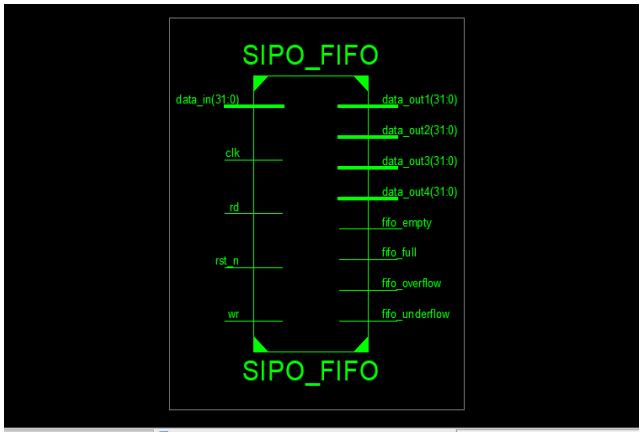
Step 8. Write address pointer write_p_0 is increased by 1

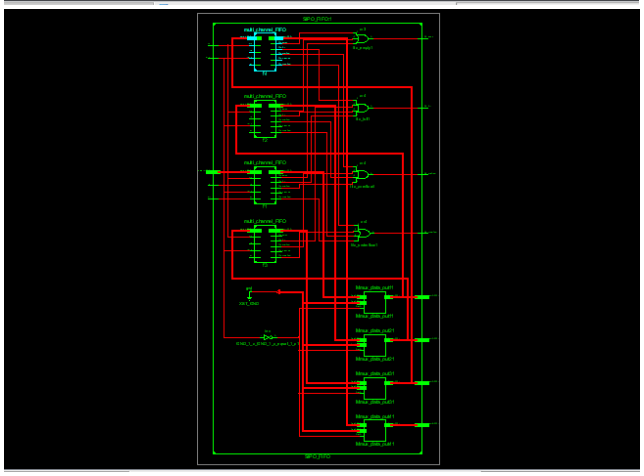
Step 9. Write channel number ch_write is increased by 1, so the next channel is selected;

Step 10. The remaining channels write data to DPRAM in the same way as channel 0 in accordance with the operation sequence from step 5 to step9 above;

Step 11. After the whole parallel write operation is completed, and if the FIFO is not full, p_write_ready is set to 1 to show the write control logic is ready for the next write operation;

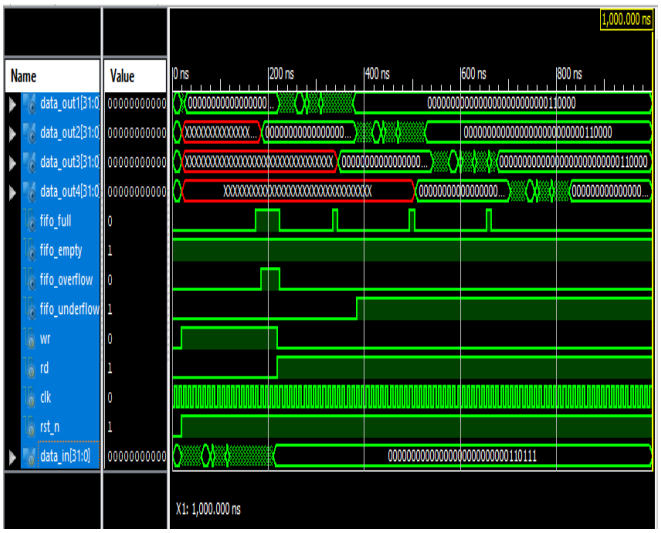
SIPO



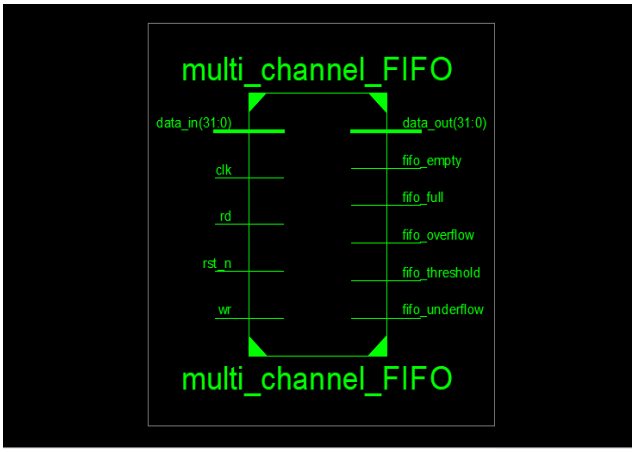


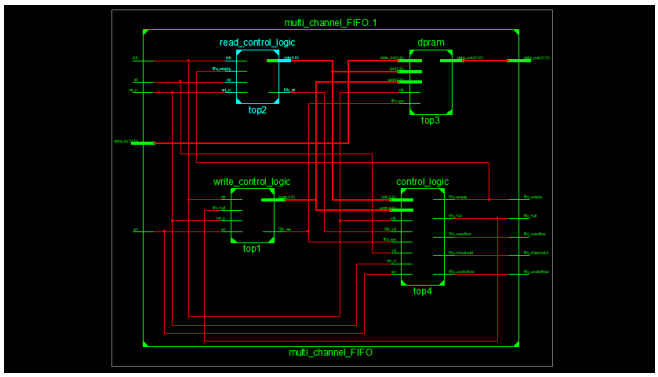
Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice Registers	31	93120	0%
Number of Slice LUTs	277	46560	0%
Number of fully used LUT-FF pairs	27	281	9%
Number of bonded IOBs	168	240	70%
Number of BUFG/BUFGCTRL/BUFHCEs	1	104	0%

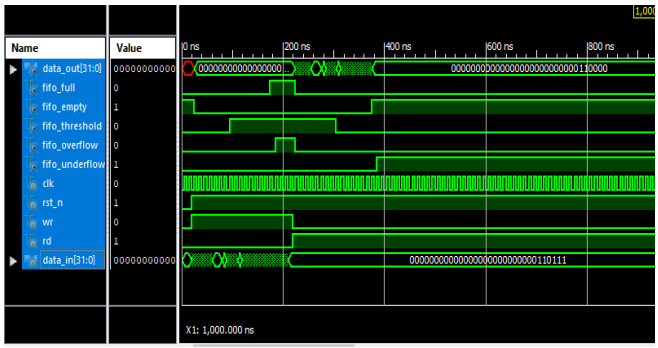
```
Timing Summary:
---------------
Speed Grade: -2

   Minimum period: 2.414ns (Maximum Frequency: 414.336MHz)
   Minimum input arrival time before clock: 1.591ns
   Maximum output required time after clock: 2.524ns
   Maximum combinational path delay: 0.985ns
```



SISO



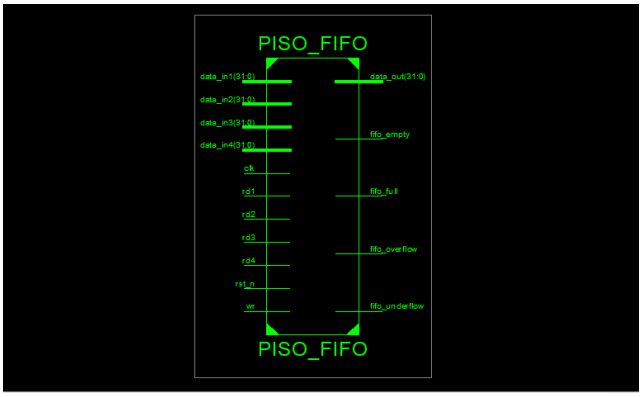


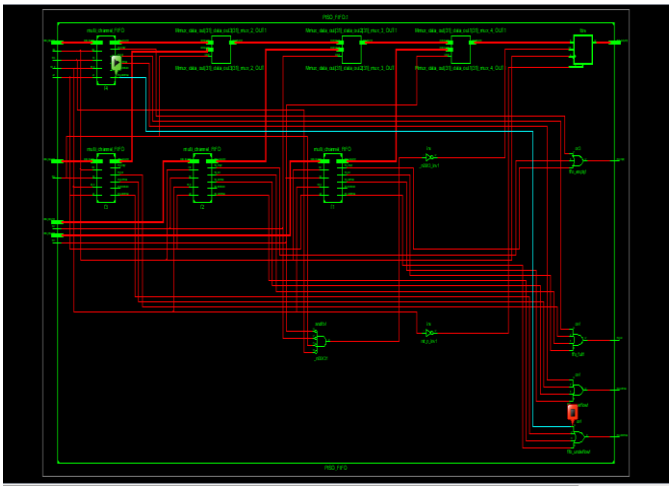


Device Utilization Summary (estimated values)			[-]
Logic Utilization	Used	Available	Utilization
Number of Slice Registers	15	93120	0%
Number of Slice LUTs	56	46560	0%
Number of fully used LUT-FF pairs	12	59	20%
Number of bonded IOBs	73	240	30%
Number of BUFG/BUFGCTRL/BUFHCEs	1	104	0%

```
Timing Summary:
---------------
Speed Grade: -2

   Minimum period: 2.458ns (Maximum Frequency: 406.884MHz)
   Minimum input arrival time before clock: 1.591ns
   Maximum output required time after clock: 2.098ns
   Maximum combinational path delay: No path found
```

PISO





4. CONCLUSION

This paper introduces a method of implementation of multi-channel FIFO with the utility of parallel access in a single piece of Block RAM. This method can help to make full use of limited Block RAM resources in FPGA and reduce the cost of terminal product, and it shows some practical value in project base on low-cost FPGA design. The simulation test and practical application indicate that this method is correct. The multi-channel FIFO created with it can run at a high frequency and is suitable for integration in most FPGA based projects.

REFERENCES

[1] Z.Lu, Y.Wu, "A Rotation-based Data Buffering Architecture for Convolution Filtering in a Field Programmable Gate Array", Journal of Computers, 8(6): pp.1411- 16, 2013.

[2] S.Kasap and K.Benkrid, "Parallel Processor Design and Implementation for Molecular Dynamics Simulations on a FPGA-Based Supercomputer", Journal of Computers, 7(6): pp.1312-1328, 2012.

[3] A.M.Fernandes, "HDL Based FPGA Interface Library for Data Acquisition and Multipurpose Real Time Algorithms", IEEE Transactions on Nuclear Science, vol.58, pp. 1526-1530, 2011.

[4] A.Afaneh, Y.He, "Implementation of accurate frame interleaved sampling in a low cost FPGA-based data acquisition system", International Conference on Intelligent Data Acquisition and Advanced Computing Systems, IEEE, pp.20-25, Sept. 2011.

[5] G..W.Zhong; H.B.Zheng, et al., "1024-point pipeline FFT processor with pointer FIFOs based on FPGA", International Conference on VLSI and System-on-Chip, IEEE, pp.122-125, Oct. 2011.

[6] H.S.Han, K.S.Stevens, "Clocked and asynchronous FIFO characterization and comparison", IEEE International Conference on Very Large Scale Integration Oct, IEEE, pp.101-108, 2009

[7] Xilinx Inc, LogiCORE IP FIFO Generator v8.2, http://www.xilinx.com, 2011.

[8] Xilinx Inc, LogiCORE IP Block Memory Generator v6.2, http://www.xilinx.com, 2011

[9] M.A.Khan, A.Q.Ansari, "n-Bit multiple read and write FIFO memory model for network-on-chip" , World Congress on Information and Communication Technologies, IEEE, pp.1322-1327, Dec. 2011.

[10] Xilinx Inc, ISE Design Suite Software Manuals, http://www.xilinx.com, 2011.