FPGA on-chip FIFO read and write Experiment

Technical Email: alinx@aithtech.com Sales Email: rachel.zhou@alinx.com

1 Experiment Introduction

This document describes how to use the internal FIFO of the FPGA and the program to read and write data to the FIFO.

2 Experiment Principle

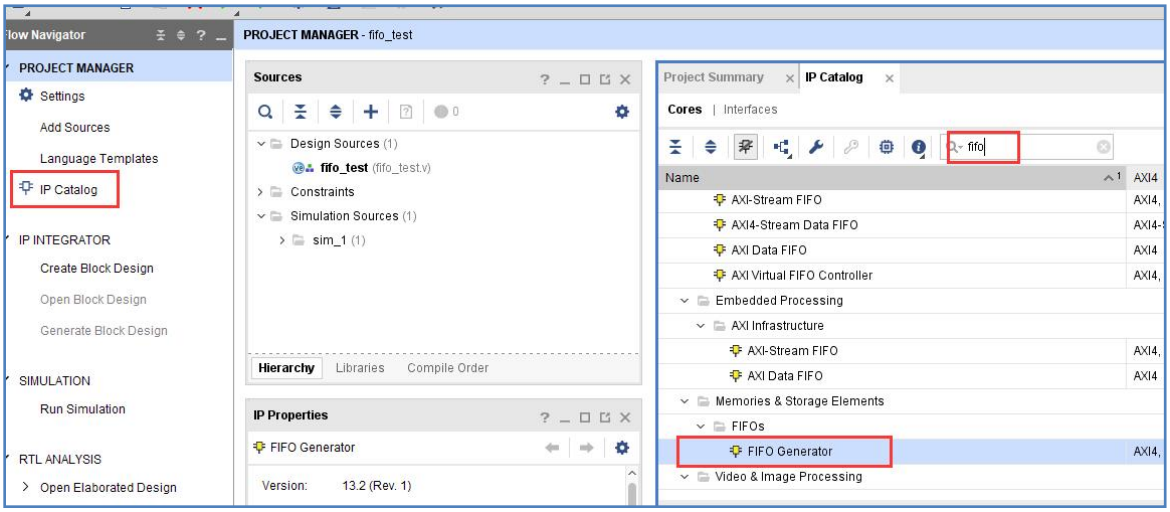
FIFO: First in, First out. Xilinx provides the FIFO IP core in VIVADO. We only need to instantiate a FIFO through the IP core to write and read the data stored in the FIFO according to the read and write timing of the FIFO. In the experiment, VIVADO integrated online logic analyzer “ila” will be used to observe the read and write timing of the FIFO and the data read from the FIFO.

3 Programming

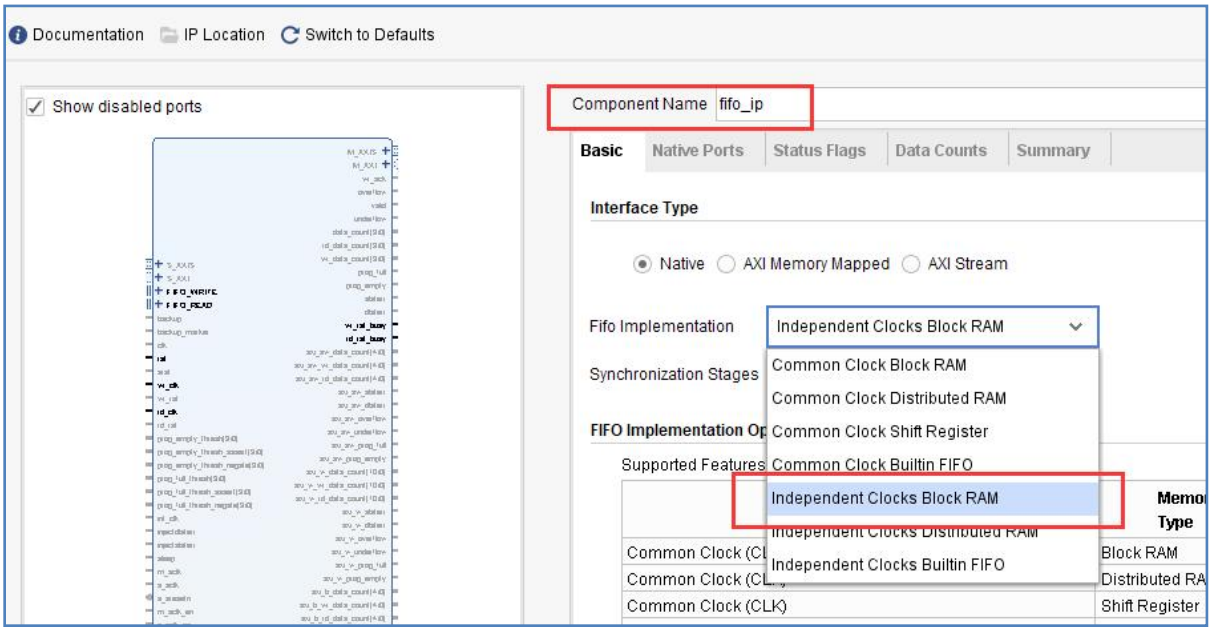
3.1 FIFO IP addition and configuration

Create a new fifo_test project before adding the FIFO IP, then add the FIFO IP to the project as follows:

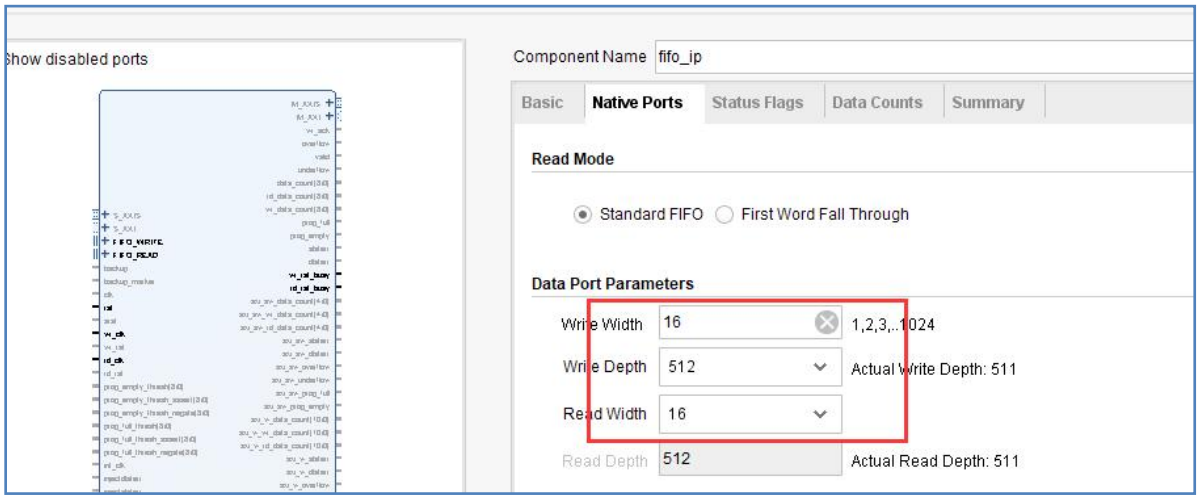
1. Click on the “IP Catalog” in the figure below, search for “fifo” in the pop-up interface on the right, find the “FIFO Generator”, and double-click to open it.



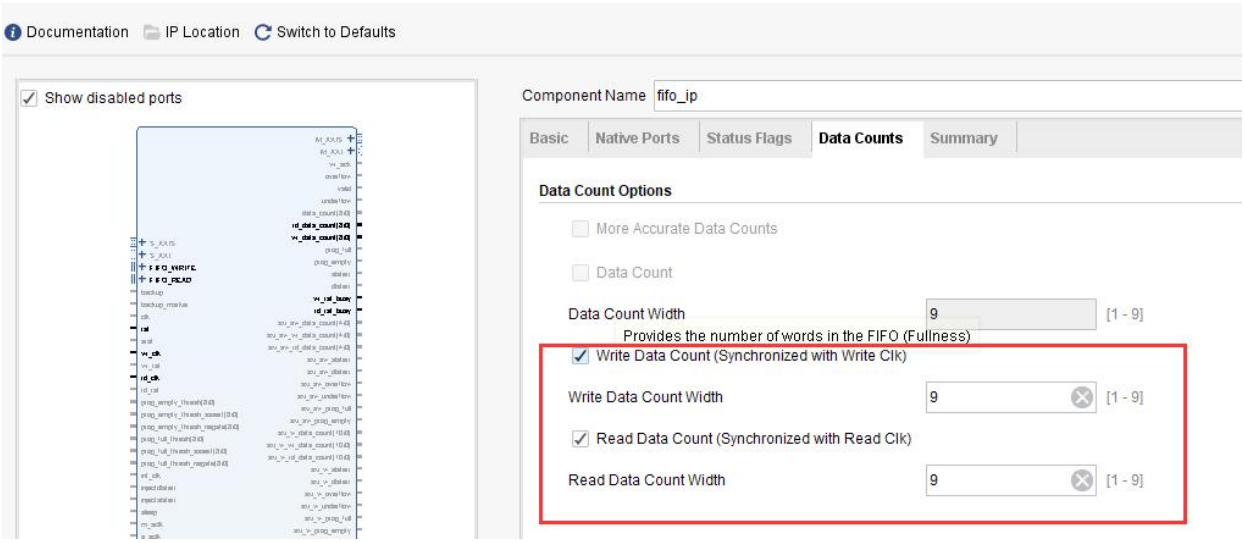
2. In the pop-up configuration page, you can choose whether to read or write the clock separately or use the same one. Generally speaking, we use “FIFO” to cache the data, usually the clock speed on both sides is different. So the independent clock is the most commonly used, we select "Independent Clocks Block RAM" here, and then click "Next" to the next configuration page.



3. Switch to the “Native Ports” column and select the data bit width of “16”; FIFO deep selection of “512”, the actual use can be set according to your own needs. Other configurations are OK bydefault.



4. Switch to the “Data Counts” column and enable the “Write Data Count” and “Read Data Count” so that we can use these two values to see how much data is inside the “FIFO”. Click “OK” and Generate generates the “FIFO IP”.

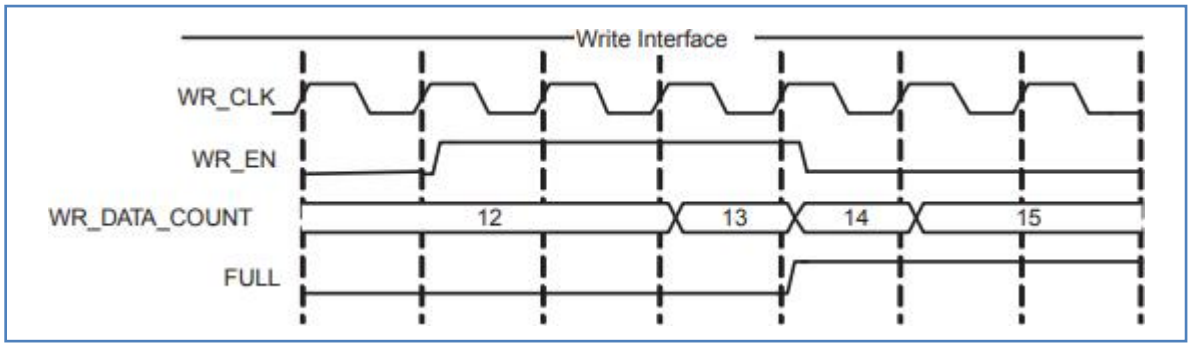


3.2 FIFO port definition and timing

Simple Dual Port RAM Module port description as below:

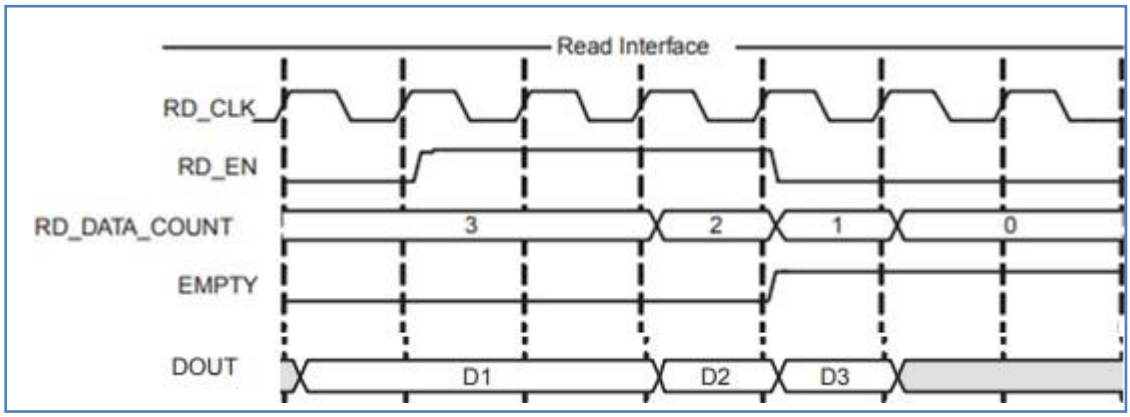
Signal Name	direction	Description
rst	in	Reset signal, high effective
wr_clk	in	Write clock input
rd_clk	in	Read clock input
din	in	Write data
wr_en	in	Write enable, high effective
rd_en	in	Read enable, high effective
dout	out	Reading data
full	out	Full Signal
empty	out	Empty signal
rd_data_count	out	Number of readable data
wr_data_count	out	Number of writable data

The data writing and reading of the FIFO are all operated according to the rising edge of the clock. When the “WR_EN” signal is high, the FIFO data is written. The data of “WR_DATA_COUNT” will not take effect immediately. For example, the third clock of the following figure starts to write the FIFO data, “WR_DATA_COUNT” The value will change on the 4th clock. The figure below shows the FIFO write timing.



FIFO Write Timing

When the "RD_EN" signal is high, the FIFO data is read. The data of "RD_DATA_COUNT" does not take effect immediately. For example, the third clock in the figure below starts to write to the "FIFO" data, and the value of "RD_DATA_COUNT" changes when the fourth clock is used. The figure below shows the FIFO read timing



FIFO Read Timing

3.3 FIFO test program

First add an instantiation of the FIFO IP. The instantiation and programming of the FIFO IP are as follows:

```
99  //------------------------------------------------------------------
100 //实例化FIFO
101 fifo_ip fifo_ip_inst (
102   .rst            (~rst_n       ),  // input rst
103   .wr_clk         (clk          ),  // input wr_clk
104   .rd_clk         (clk          ),  // input rd_clk
105   .din            (w_data       ),  // input [15 : 0] din
106   .wr_en          (wr_en        ),  // input wr_en
107   .rd_en          (rd_en        ),  // input rd_en
108   .dout           (r_data       ),  // output [15 : 0] dout
109   .full           (full         ),  // output full
110   .empty          (empty        ),  // output empty
111   .rd_data_count  (rd_data_count),  // output [8 : 0] rd_data_count
112   .wr_data_count  (wr_data_count)   // output [8 : 0] wr_data_count
113 );
```

The FIFO write process will determine if the FIFO is empty. If it is empty, start writing data and write until it is full.

```
37  always@(*)
38  begin
39      case(write_state)
40          W_IDLE:
41              if(empty == 1'b1)                //FIFO空，开始写FIFO
42                  next_write_state <= W_FIFO;
43              else
44                  next_write_state <= W_IDLE;
45          W_FIFO:
46              if(full == 1'b1)                 //FIFO满
47                  next_write_state <= W_IDLE;
48              else
49                  next_write_state <= W_FIFO;
50          default:
51              next_write_state <= W_IDLE;
52      endcase
53  end
54
55  assign wr_en = (next_write_state == W_FIFO) ? 1'b1 : 1'b0;
```

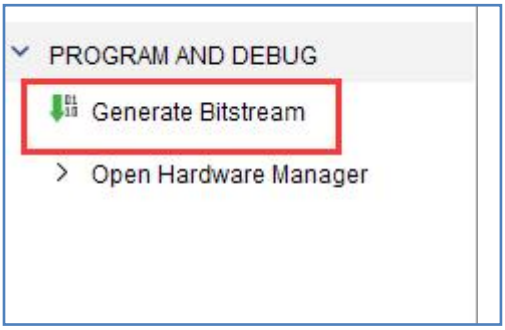
The FIFO read process will determine if the FIFO is full. If it is full, it starts reading the data and reads it until it is empty.

```
79  always@(*)
80  begin
81      case(read_state)
82          R_IDLE:
83              if(full == 1'b1)                //FIFO满，开始读FIFO
84                  next_read_state <= R_FIFO;
85              else
86                  next_read_state <= R_IDLE;
87          R_FIFO:
88              if(empty == 1'b1)               //FIFO空
89                  next_read_state <= R_IDLE;
90              else
91                  next_read_state <= R_FIFO;
92          default:
93              next_read_state <= R_IDLE;
94      endcase
95  end
96
97  assign rd_en = (next_read_state == R_FIFO) ? 1'b1 : 1'b0;
```

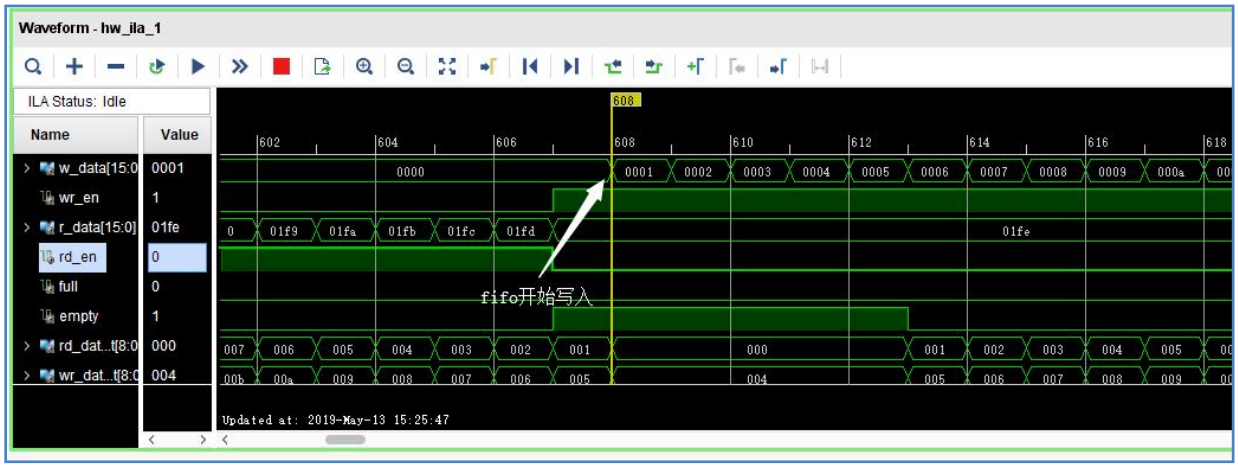
In order to see the FIFO data read and write in real time, we have added the ila tool to observe the FIFO data signal and control signal. Please refer to the "I2C Interface EEPROM Experiment.pdf" tutorial for how to generate ila.

4 Experiment Result

Generate a “bit” file and download it to the “FPGA”. Observe the read and write data of the FIFO through “ila”



In the “Waveform” window we can see that when “empty” becomes high, the “FIFO” starts writing data “(0~1FE)”.



When the “full” signal of the FIFO goes high, the FIFO stops writing and starts reading data (0~1FE).