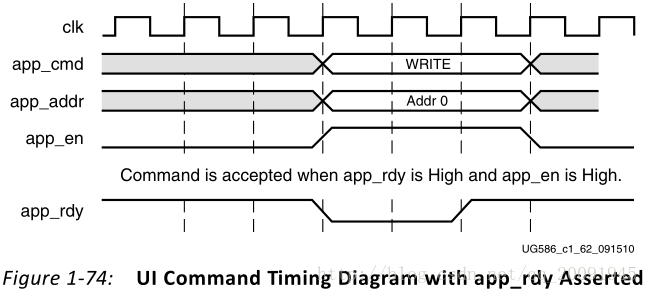
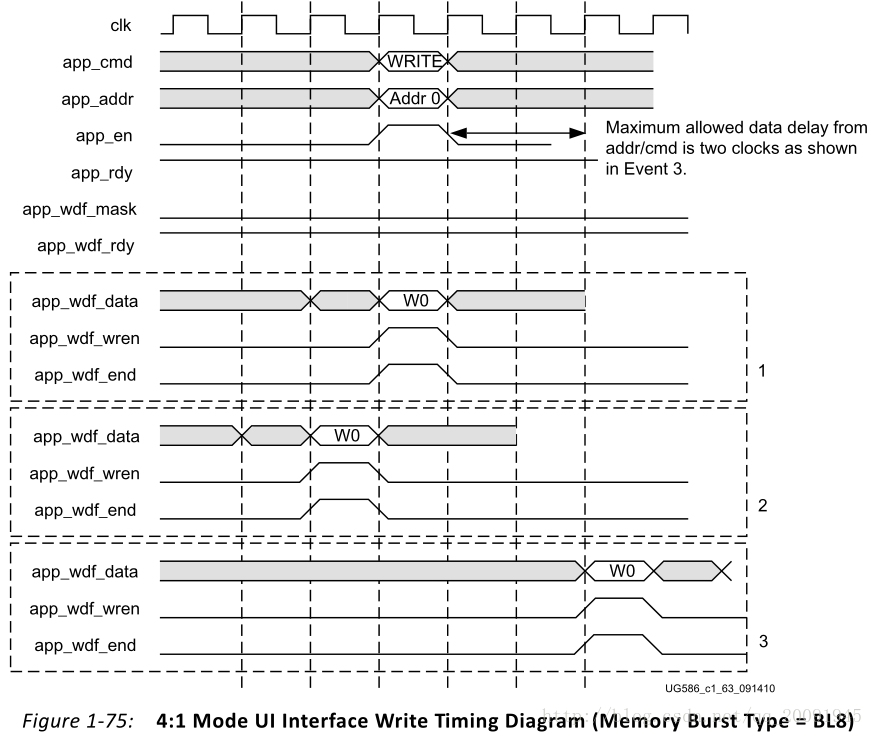
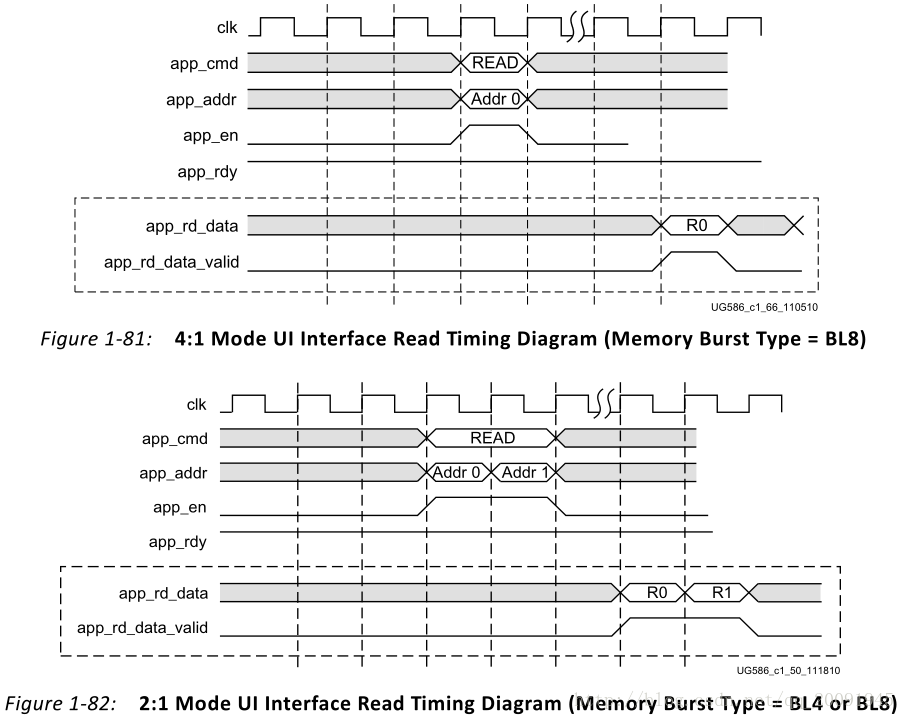
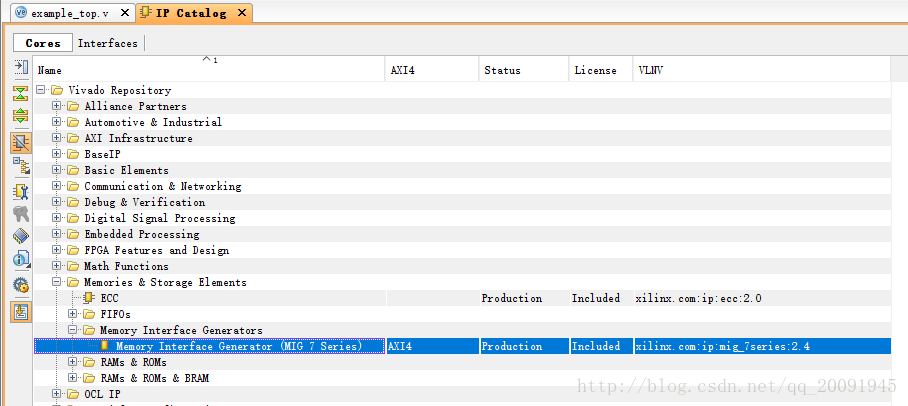
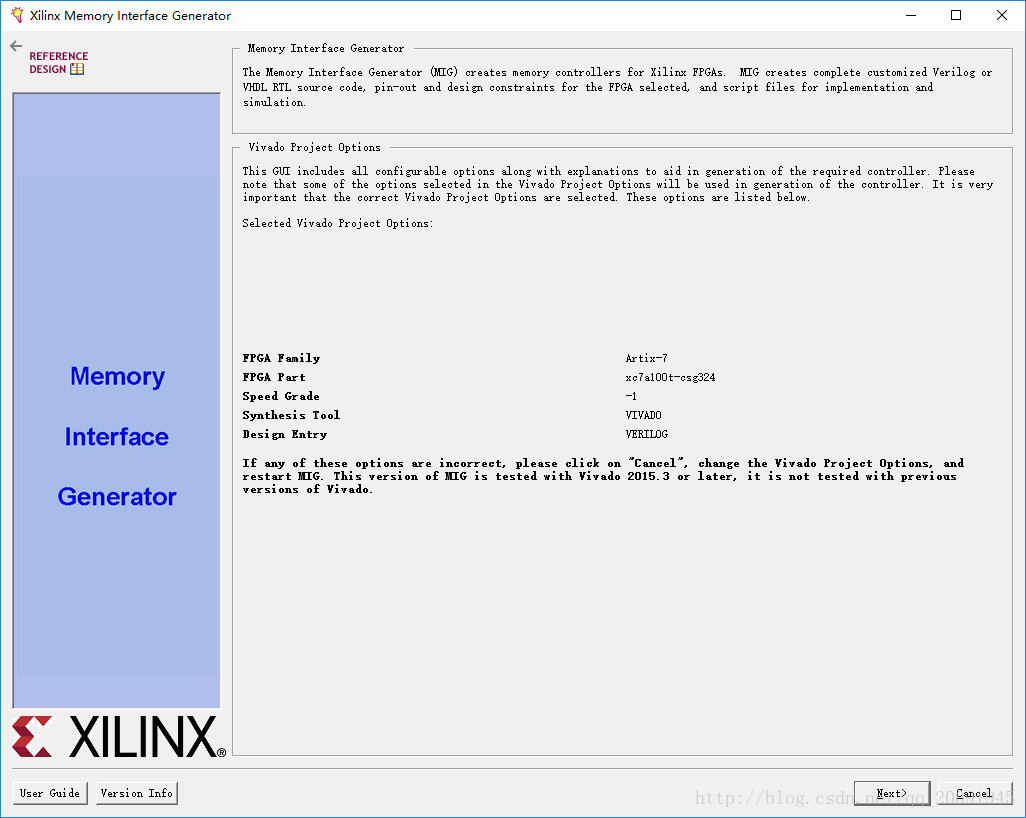
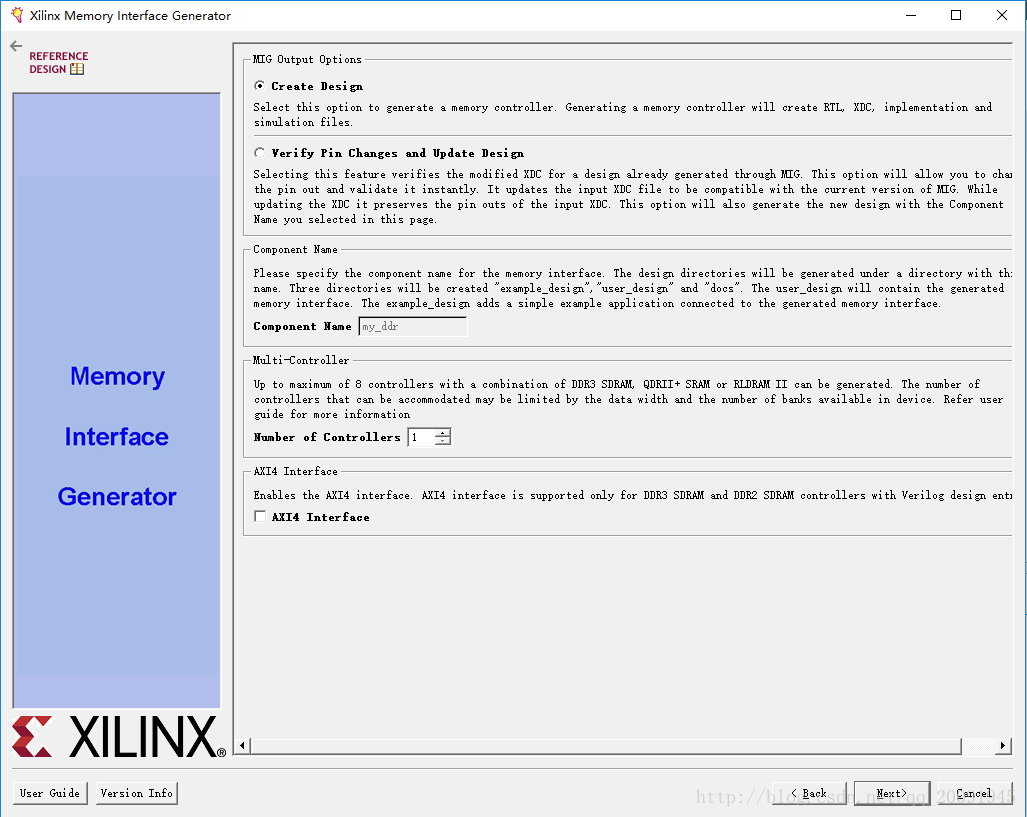
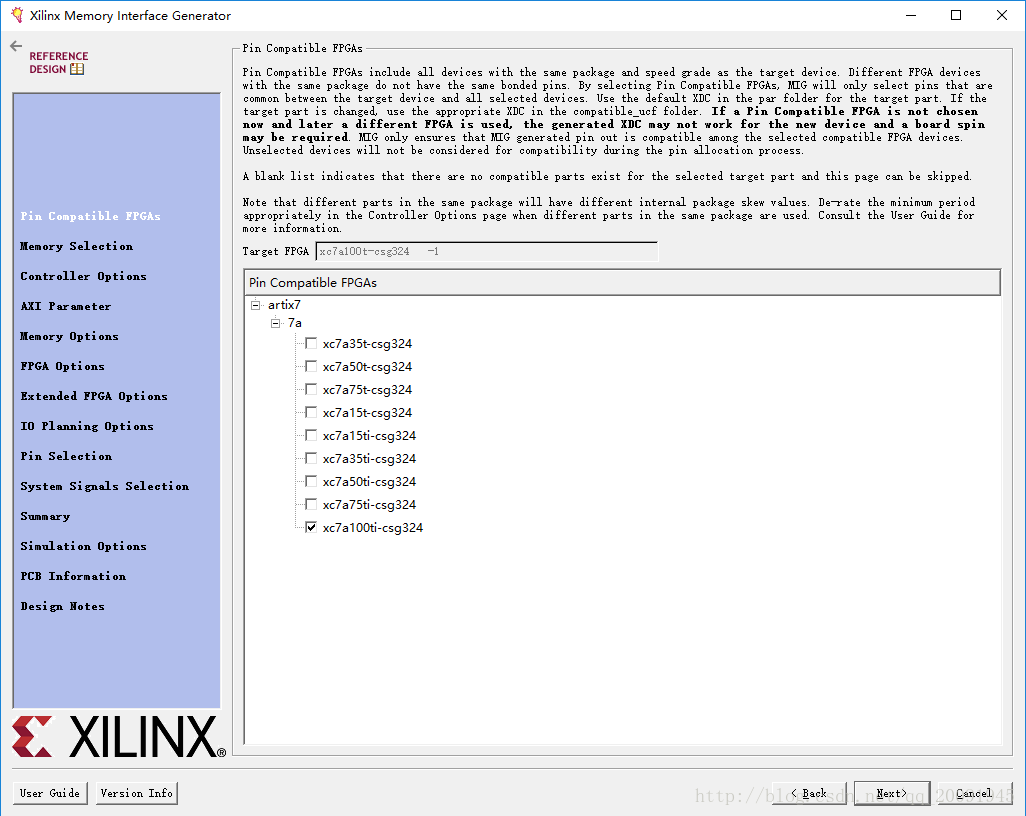
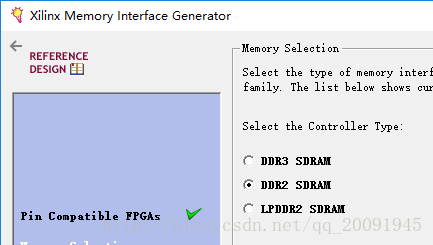
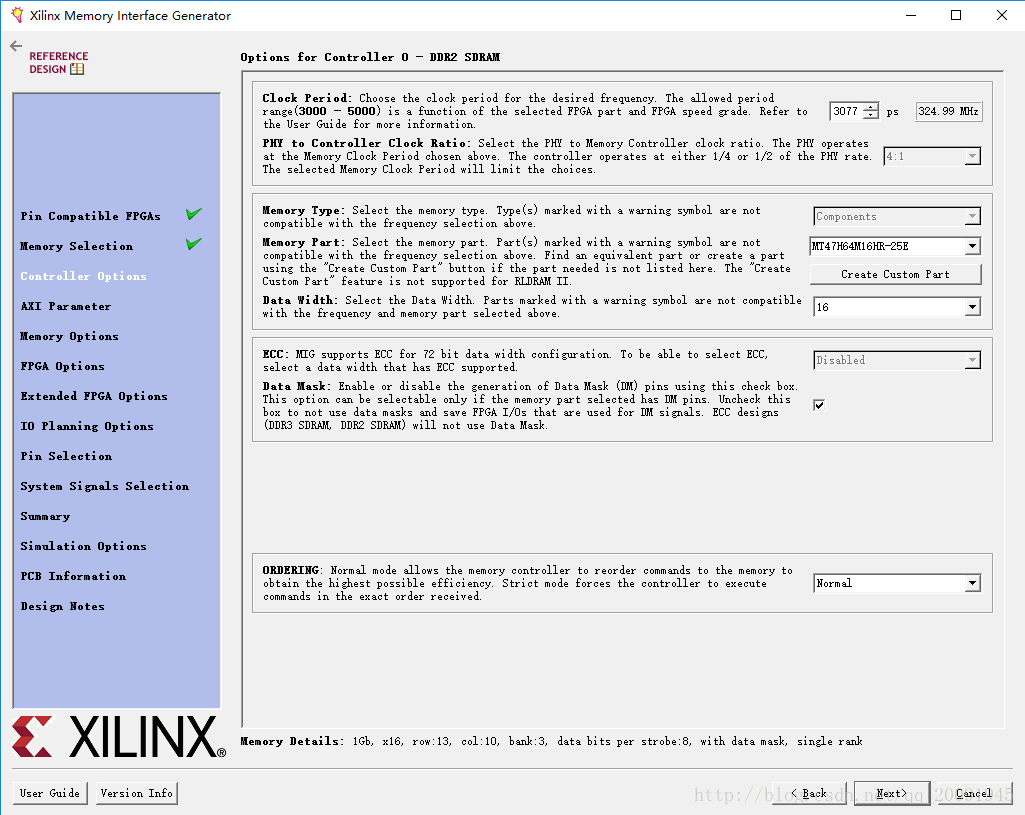
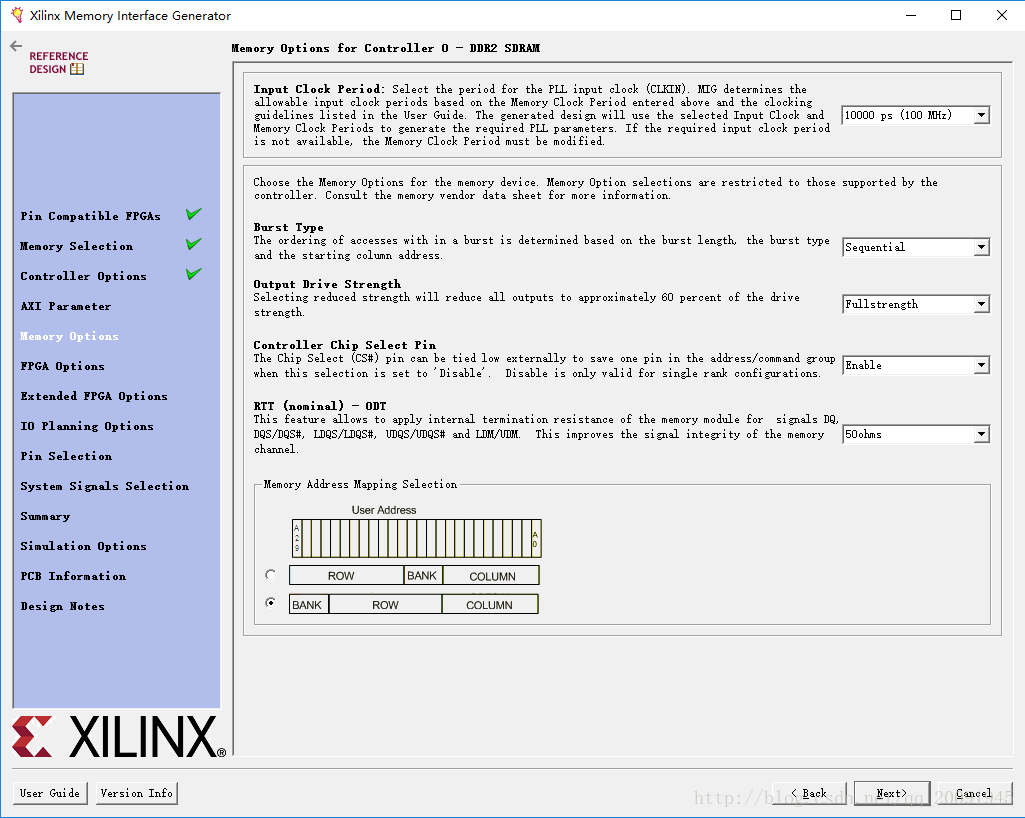
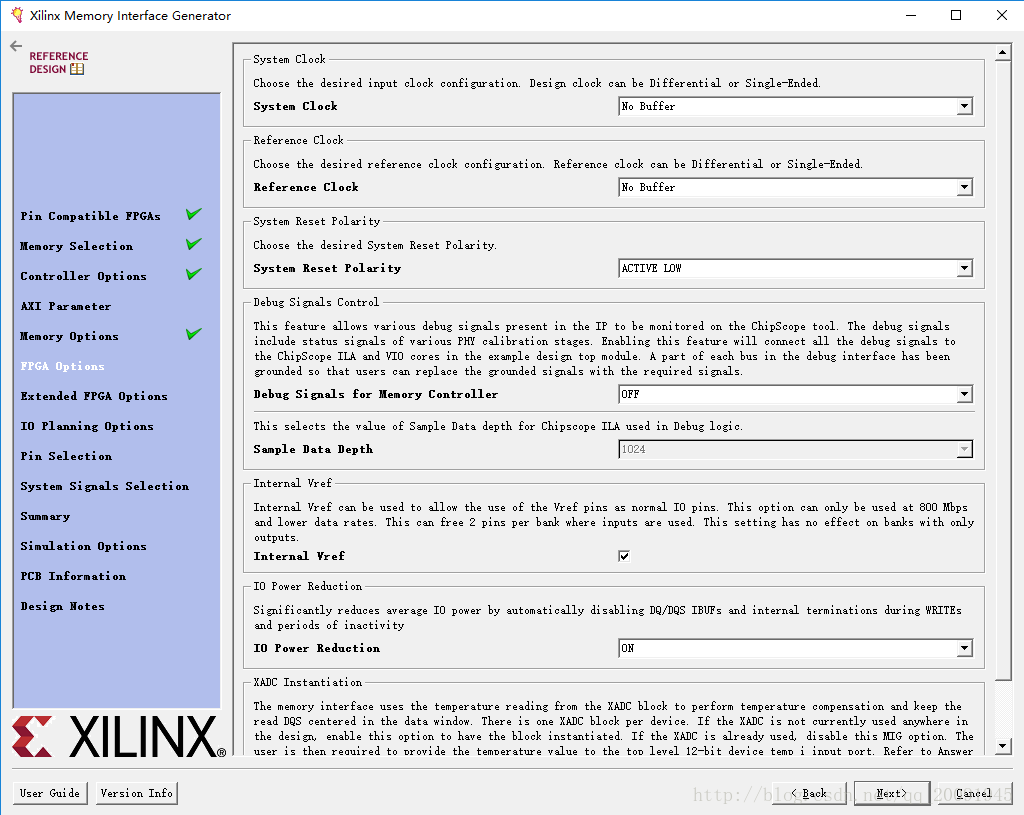
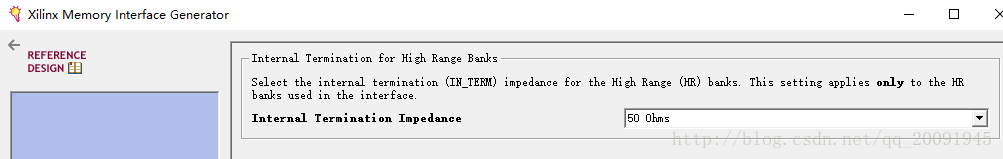
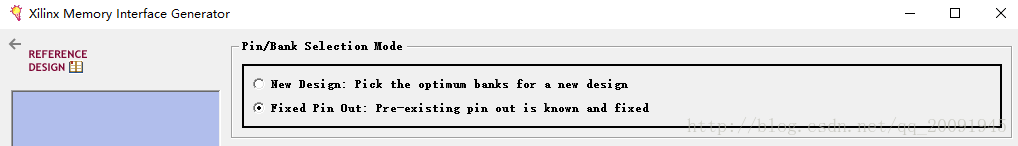
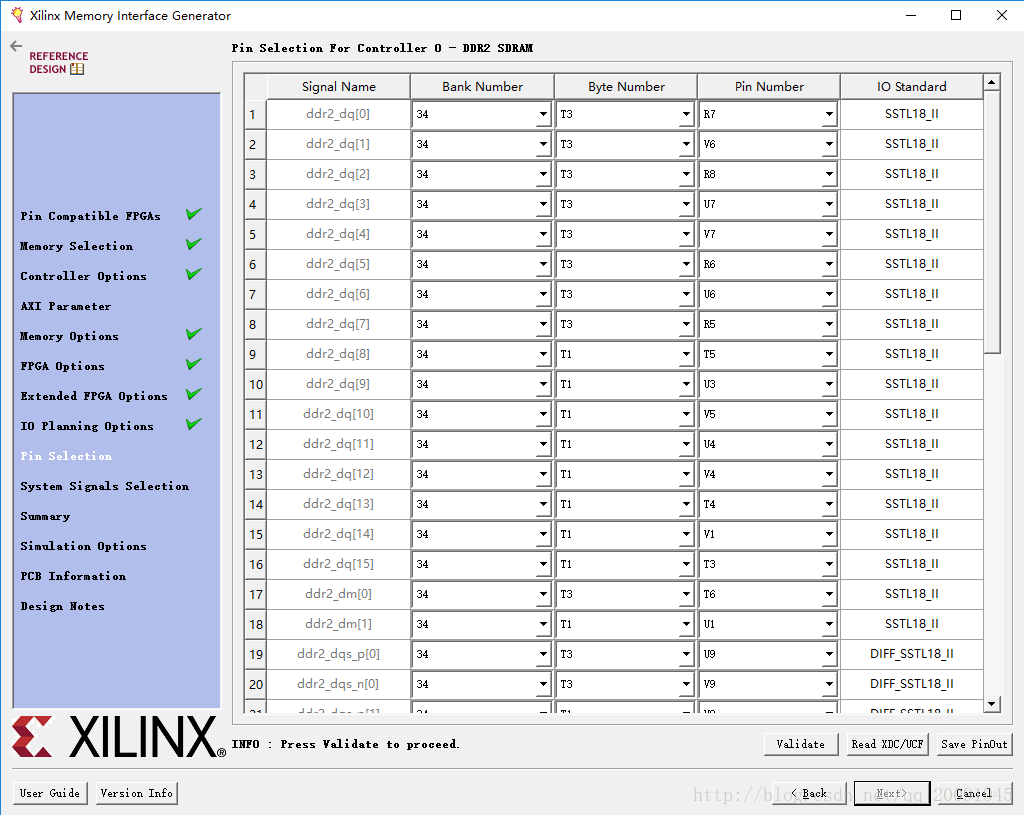
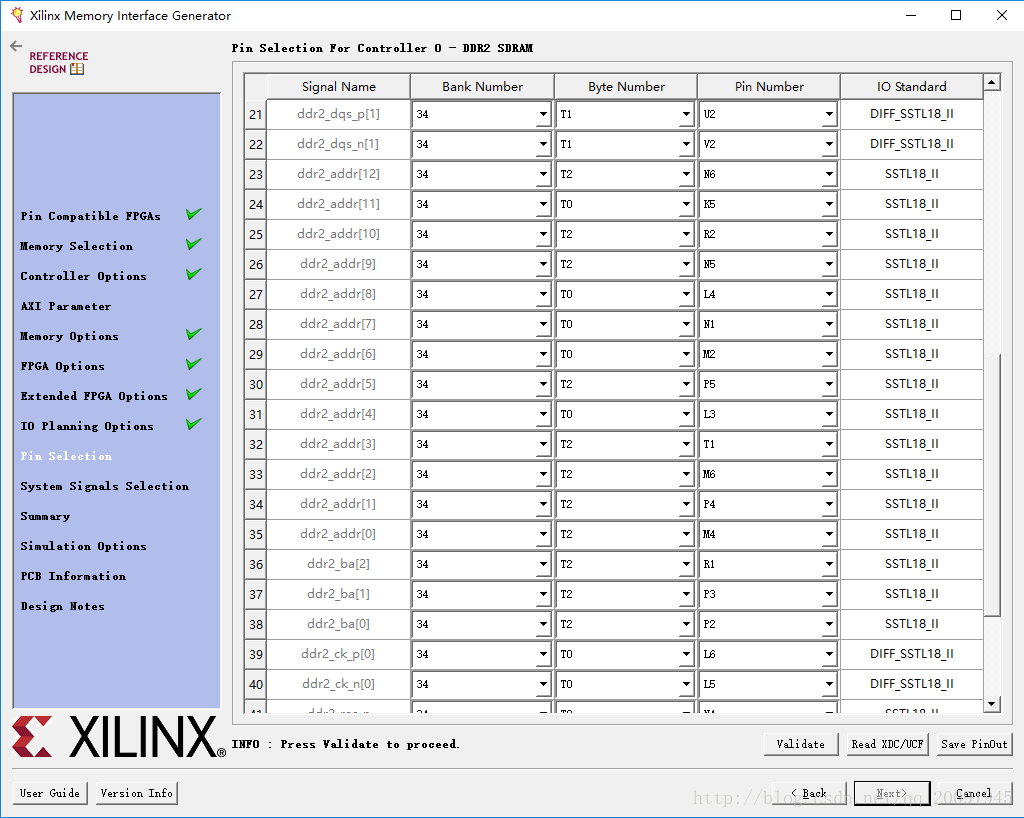
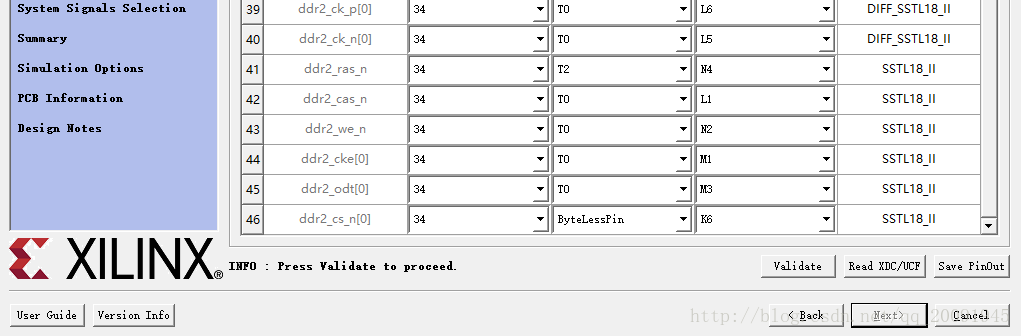
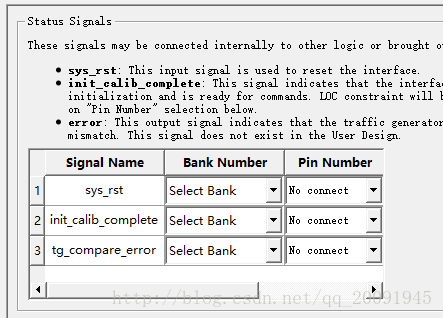
# 在Nexys4 DDR上实现的DDR2读写例程

· 本文使用Vivado 2015.4在Nexys4 DDR（以下简称N4DDR）开发板上实现DDR的读写。   
· FPGA如果需要对DDR进行读写，则需要一个DDR的控制器。根据官方的文档（UG586，下载链接在文末），DDR控制器的时序主要有三：   
（1）首先是控制信号，如下图：   
   
· 从上图可以看出，只有当app\_rdy信号有效时，程序所发出的读写命令才会被控制器接收。这点必须注意。   
（2）然后是写操作时序，如下图：   
· 由图可知，在向DDR写数据时，需要提供写命令app\_cmd、地址app\_addr、数据app\_wdf\_data等信号，且写入的数据最多可以比app\_cmd提前一个时钟周期有效，最迟可以比app\_cmd晚两个时钟周期有效。   
【特别注意】在写数据的时候必须检测**app\_rdy和app\_wdf\_rdy信号是否**同时**有效**，否则写入命令无法成功写入到DDR控制器的命令FIFO中，从而导致写操作失败。   
（3）最后是读操作时序，如下图所示：   
  
读操作的时序比较简单，只需要**注意app\_rdy是否有效**即可，其余不再赘述。

Xilinx在Vivado中提供的Memory Interface Generator的IP核就是我们需要的DDR控制器，如下图所示。   
  
这里我们可以直接双击上面的MIG的IP核，开始例化我们所需的DDR控制器。（**此时Win7以后的Windows版本(不含Win7)打开此IP核会报错，解决方法见**[http://blog.csdn.net/qq\_20091945/article/details/53862467](http://blog.csdn.net/qq_20091945/article/details/53862467" \t "_blank)）   
打开后是如下图所示的界面，点Next。   
  
给模块起个名字，根据实际情况选择控制器数量（这里笔者选择1），继续Next，如下图所示。   
  
· 在开发板芯片型号所对应的方框前打勾，如下图所示。   
  
· 根据开发板上的DDR芯片选择DDR的种类，如N4DDR的开发板上的DDR芯片是DDR2的，因此如下图选择。   
   
然后在**Clock Period**中输入合适的时钟周期长度（N4DDR的官方文档建议DDR的时钟为325MHz，故此处填3077ps）；   
接着在**Memory Part**中选择开发板上的DDR芯片的具体型号（N4DDR官方文档上说明为MT47H64M16HR-25E）； 然后输入**Data Width**，此处以16为例。如下图所示。   
  
选择**Input Clock Period**，这里填开发板的系统时钟（N4DDR为100MHz）。根据应用需要选择地址映射方式（这里保持默认的Bank-Row-Column）。   
  
· 然后，这里的**System Clock**、**Reference Clock**建议选择No Buffer，**System Reset Polarity**则根据应用需要灵活选择（这里设置为低电平有效），如下图所示。   
  
· **Internal Termination Impedence**的选取应当参考开发板的官方文档说明，这里选50欧姆即可，继续Next。   
  
· 选择**Fixed Pin Out**。   
  
· 接下来是DDR芯片的引脚分配。官网应该能找到，这里直接给出。文末会给出与此对应的引脚约束文件(n4ddr\_ddr2\_io\_assign.ucf)。   
  
  
  
· 耐心填完之后点击Validate按钮，没有错误的话会弹出一个对话框提示“Current Pinout is valid.”   
· 然后的3个信号建议选择No connect，后面由我们自己根据需要连接到板上的相应引脚。   
   
· 后面一直Next下去，点Accept，然后就可以点击Generate了。后面会再弹出一个对话框，直接点默认选中的按钮即可。

· 好了，下面是笔者自己编写的测试DDR2读写的程序。文末将提供对应工程的下载链接。

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*// Author : Z.M.J. @ CSE, SEU*

*// Application : MIG v2.4*

*// Filename : example\_top.v*

*// Date Created : Fri Dec 30 2016*

*//*

*// Device : 7 Series (Nexys 4 DDR)*

*// Design Name : DDR2 SDRAM*

*// Purpose : A demo of DDR2's read and write*

*// Reference : ug586\_7Series\_MIS\_v2.4.pdf*

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

`timescale1ps/1ps

moduleexample\_top (

*// system signals*

inputsys\_rst,

inputsys\_clk\_i,

*// application signals*

input [15:0]switch\_i,

output[15:0] led,

output[7:0] an,

output[7:0]select\_seg,

*// DDR2 chip signals*

inout[15:0] ddr2\_dq,

inout[1:0] ddr2\_dqs\_n,

inout[1:0] ddr2\_dqs\_p,

output[12:0] ddr2\_addr,

output[2:0] ddr2\_ba,

output ddr2\_ras\_n,

output ddr2\_cas\_n,

output ddr2\_we\_n,

output[0:0] ddr2\_ck\_p,

output[0:0] ddr2\_ck\_n,

output[0:0] ddr2\_cke,

output[0:0] ddr2\_cs\_n,

output[1:0] ddr2\_dm,

output[0:0] ddr2\_odt

);

parameter DQ\_WIDTH = 16;

parameter ECC\_TEST = "OFF";

parameter ADDR\_WIDTH = 27;

parameternCK\_PER\_CLK = 4;

localparam DATA\_WIDTH = 16;

localparam PAYLOAD\_WIDTH = (ECC\_TEST == "OFF") ? DATA\_WIDTH : DQ\_WIDTH;

localparam APP\_DATA\_WIDTH = 2 \* nCK\_PER\_CLK \* PAYLOAD\_WIDTH;

localparam APP\_MASK\_WIDTH = APP\_DATA\_WIDTH / 8;

*// Wire declarations*

regapp\_en, app\_wdf\_wren, app\_wdf\_end;

reg[2:0]app\_cmd;

reg[ADDR\_WIDTH-1:0]app\_addr;

reg[APP\_DATA\_WIDTH-1:0]app\_wdf\_data;

wire[APP\_DATA\_WIDTH-1:0]app\_rd\_data;

wire[APP\_MASK\_WIDTH-1:0]app\_wdf\_mask;

wireapp\_rdy, app\_rd\_data\_end, app\_rd\_data\_valid, app\_wdf\_rdy;

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

wire[7:0] an;

wire[7:0]select\_seg;

reg[31:0]digit\_data;

always@ (posedgesys\_clk\_i) begin

if (switch\_i[3])

digit\_data<= app\_addr;

elsecase (switch\_i[1:0])

2'b00 :digit\_data<= read\_data[31:0];

2'b01 :digit\_data<= read\_data[63:32];

2'b10 :digit\_data<= read\_data[95:64];

2'b11 :digit\_data<= read\_data[127:96];

endcase

end

digit U2(

.wb\_clk\_i(sys\_clk\_i),

.wb\_rst\_i(~sys\_rst),

.wb\_dat\_i(digit\_data),

.an(an),

.select\_seg(select\_seg)

);

reg[1:0]read\_valid = 2'b0;

reg [127:0] read\_data = 128'h0;

always@ (posedgeapp\_rd\_data\_valid) begin

read\_data = app\_rd\_data;

read\_valid[0] = (app\_rd\_data == data0);

read\_valid[1] = (app\_rd\_data == data1);

end

assign led[15] = app\_en;

assign led[14] = init\_calib\_complete;

assign led[13] = app\_rdy;

assign led[12] = app\_wdf\_rdy;

assign led[4] = sys\_rst ? read\_valid[1] : 1'b0;

assign led[3] = sys\_rst ? read\_valid[0] : 1'b0;

assign led[2] = stop\_w[1];

assign led[1] = stop\_w[0];

assign led[0] = app\_cmd[0];

reg[15:0] counter = 16'h0;

parametercnt\_init = 16'h1; *// minimum: 1*

reg[26:0] addr0 = 27'h000\_0008;

reg [26:0] addr1 = 27'h003\_0100;

reg[127:0] data0 = 128'h1111\_2222\_3333\_4444\_5555\_6666\_7777\_8888;

reg [127:0] data1 = 128'h9999\_0000\_aaaa\_bbbb\_cccc\_dddd\_eeee\_ffff;

reg[1:0]stop\_w = 2'b00;

always@ (posedgesys\_clk\_i or negedgesys\_rst) begin

if (sys\_rst == 1'b0) begin

counter = 12'b0;

stop\_w = 2'b0;

app\_en = 1'b0;

app\_addr = 27'h0;

app\_cmd = 3'b1;

app\_wdf\_data = 128'h0;

app\_wdf\_end = 1'b0;

app\_wdf\_wren = 1'b0;

endelse begin

if (counter == cnt\_init&& ~stop\_w[0])

if (app\_rdy&app\_wdf\_rdy) begin

app\_wdf\_data = data0;

app\_addr = addr0;

app\_cmd = 3'b0;

app\_wdf\_wren = 1'b1;

app\_wdf\_end = 1'b1;

app\_en = 1'b1;

endelse*// Hold specific signals until app\_wdf\_rdy is asserted.*

counter = counter - 16'h1;

else if (counter == cnt\_init + 1 && ~stop\_w[0])

if (app\_rdy&app\_wdf\_rdy) begin

app\_wdf\_end = 1'b0;

app\_wdf\_wren = 1'b0;

app\_en = 1'b0;

app\_cmd = 3'b1;

stop\_w[0] = 1'b1;

endelse*// Hold specific signals until app\_wdf\_rdy is asserted.*

counter = counter - 16'h1;

else if (counter == cnt\_init + 8 && ~stop\_w[1])

if (app\_rdy&app\_wdf\_rdy) begin

app\_wdf\_data = data1;

app\_addr = addr1;

app\_cmd = 3'b0;

app\_wdf\_wren = 1'b1;

app\_wdf\_end = 1'b1;

app\_en = 1'b1;

end else // Hold specific signals until app\_wdf\_rdy is asserted.

counter = counter - 16'h1;

elseif (counter == cnt\_init + 9&& ~stop\_w[1])

if (app\_rdy&app\_wdf\_rdy) begin

app\_wdf\_end = 1'b0;

app\_wdf\_wren = 1'b0;

app\_en = 1'b0;

app\_cmd = 3'b1;

stop\_w[1] = 1'b1;

end else // Hold specific signals until app\_wdf\_rdy is asserted.

counter = counter - 16'h1;

elseif (counter == cnt\_init + 88) begin

app\_addr = switch\_i[2] ? addr1 : addr0;

app\_en = 1'b1;

if (~app\_rdy) counter = counter - 16'h1;

endelseif (counter == cnt\_init + 89)

app\_en = 1'b0;

counter = counter + 16'h1;

end

end

*// Start of User Design top instance*

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*// The User design is instantiated below. The memory interface ports are*

*// connected to the top-level and the application interface ports are*

*// connected to the traffic generator module. This provides a reference*

*// for connecting the memory controller to system.*

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

my\_ddru\_my\_ddr (

*// Memory interface ports*

.ddr2\_cs\_n (ddr2\_cs\_n),

.ddr2\_addr (ddr2\_addr),

.ddr2\_ba (ddr2\_ba),

.ddr2\_we\_n (ddr2\_we\_n),

.ddr2\_ras\_n (ddr2\_ras\_n),

.ddr2\_cas\_n (ddr2\_cas\_n),

.ddr2\_ck\_n (ddr2\_ck\_n),

.ddr2\_ck\_p (ddr2\_ck\_p),

.ddr2\_cke (ddr2\_cke),

.ddr2\_dq (ddr2\_dq),

.ddr2\_dqs\_n (ddr2\_dqs\_n),

.ddr2\_dqs\_p (ddr2\_dqs\_p),

.ddr2\_dm (ddr2\_dm),

.ddr2\_odt (ddr2\_odt),

*// Application interface ports*

.app\_addr (app\_addr),

.app\_cmd (app\_cmd),

.app\_en (app\_en),

.app\_wdf\_rdy (app\_wdf\_rdy),

.app\_wdf\_data (app\_wdf\_data),

.app\_wdf\_end (app\_wdf\_end),

.app\_wdf\_wren (app\_wdf\_wren),

.app\_rd\_data (app\_rd\_data),

.app\_rd\_data\_end (app\_rd\_data\_end),

.app\_rd\_data\_valid (app\_rd\_data\_valid),

.app\_rdy (app\_rdy),

.app\_sr\_req (1'b0),

.app\_ref\_req (1'b0),

.app\_zq\_req (1'b0),

.app\_wdf\_mask (16'h0000),

.init\_calib\_complete (init\_calib\_complete),

*// System Clock Ports*

.sys\_clk\_i (sys\_clk\_i),

*// Reference Clock Ports*

.clk\_ref\_i (sys\_clk\_i),

.sys\_rst (sys\_rst)

);

endmodule

* 1

· 保存后直接生成比特流就可以下板验证了。   
· 在摸索过程中笔者发现，写入了数据之后最快要到发出写命令的第8个系统时钟才能读出所写入的数据，且读操作必须在写操作后经过8的整数倍个时钟后进行。有时将比特流下载到N4DDR上面之后读写的数据有误，但是重启开发板再重新下载即可解决问题，知道个中缘由的朋友欢迎在评论中告知笔者，笔者在此先行谢过。   
· 需要说明的是，此处突发长度（BL）为8，因此app\_addr必须是8对齐的地址。同时，由于前面选择的Data Width为16，因此每次读写数据的长度为8\*16bit==128bit。

· 笔者水平有限，文中难免存在纰漏，望读者在评论中慷慨指出。笔者在此先行谢过。

· 参考文档：   
[Nexys4-DDR\_rm.pdf](http://download.csdn.net/detail/qq_20091945/9725288)   
[ug586\_7Series\_MIS\_v2.4.pdf](http://download.csdn.net/detail/qq_20091945/9725297)

· 模板工程下载链接：   
<http://download.csdn.net/detail/qq_20091945/9728980>

· 测试代码下载链接：   
<http://download.csdn.net/detail/qq_20091945/9725407>   
【注意】由于工程大小超出上限，此处仅提供工程的源码文件和比特流文件。