FPGA 开源工作室将通过五篇文章来给大家讲解 xilinx FPGA 使用 mig IP 对 DDR3 的读写控制，旨在让大家更快的学习和应用 DDR3。
本实验和工程基于 Digilent 的 Arty Artix-35T FPGA 开发板完成。
软件使用 Vivado 2018.1。
参考工程：ddr3_test。

第五篇：mig 读写时序下板实现

1 顶层文件和约束文件

ddr3_test.v

```
`timescale 1ns / 1ps
`define CMD_WR 3'b000
`define CMD_RD 3'b001
module ddr3_test(
       input clk,//100MHZ
       input reset,
       output init_calib_complete,
      //DDR3 Interface
      // Inouts
      inout [15:0]        ddr3_dq,
      inout [1:0]         ddr3_dqs_n,
      inout [1:0]         ddr3_dqs_p,
      // Outputs
      output [13:0]       ddr3_addr,
      output [2:0]        ddr3_ba,
      output              ddr3_ras_n,
      output              ddr3_cas_n,
      output              ddr3_we_n,
      output              ddr3_reset_n,
      output [0:0]        ddr3_ck_p,
      output [0:0]        ddr3_ck_n,
      output [0:0]        ddr3_cke,
      output [0:0]        ddr3_cs_n,
      output [1:0]        ddr3_dm,
      output [0:0]         ddr3_odt
       );
//*****************************************************************************
       parameter IDLE  = 5'd0,
                 WR1   = 5'd1,
                 WR2   = 5'd2,
                 WR3   = 5'd3,
                 WR4   = 5'd4,
                  WR5    = 5'd5,
                  WR6    = 5'd6,
                  WR7    = 5'd7,
                  WR8    = 5'd8,
                  RD1    = 5'd9,
                  RD2    = 5'd10,
                  RD3    = 5'd11,
                  RD4    = 5'd12,
                  RD5    = 5'd13,
                  RD6    = 5'd14,
                  RD7    = 5'd15,
                  RD8    = 5'd16,
                  DONE   = 5'd17;
           //
              // Single-ended system clock
           wire                                          sys_clk_i;//166.667MHZ

           // Single-ended iodelayctrl clk (reference clock)
           wire                                           clk_ref_i;//200MHZ


             // user interface signals
             wire [27:0]        app_addr;//i
             reg [2:0]         app_cmd;//i
             wire               app_en;//i
             reg [127:0]       app_wdf_data;//i
             wire               app_wdf_end;//i
             wire [15:0]        app_wdf_mask;//i
             wire              app_wdf_wren;//i
             wire [127:0]       app_rd_data;
             wire               app_rd_data_end;
             wire               app_rd_data_valid;
             wire               app_rdy;
             wire               app_wdf_rdy;
             wire               app_sr_req;//i
             wire               app_ref_req;//i
             wire               app_zq_req;//i
             wire               app_sr_active;
             wire               app_ref_ack;
             wire               app_zq_ack;
             wire               ui_clk;
             wire               ui_clk_sync_rst;
```

```
        //wire                  init_calib_complete;
        wire                   sys_rst_n;
        wire                   sys_rst;
        assign sys_rst =     sys_rst_n;
        reg [4:0] cstate,nstate;
        //wire [27:0]         wr_addr;//i  bank row column [2:0] [13:0] [9:0]
        //wire [27:0]         rd_addr;//i  bank row column [2:0] [13:0] [9:0]
        wire wr1_done;
        wire wr2_done;
        wire wr3_done;
        wire wr4_done;
        wire wr5_done;
        wire wr6_done;
        wire wr7_done;
        wire wr8_done;

        wire rd1_done;
        wire rd2_done;
        wire rd3_done;
        wire rd4_done;
        wire rd5_done;
        wire rd6_done;
        wire rd7_done;
        wire rd8_done;

        reg [2:0] bank;
        //reg [13:0] row;
        //reg [9:0] column;
        reg [23:0] addr;
    //assign  wr_addr =(cstate == WR1 ||cstate == WR2 ||cstate == WR3 ||cstate == WR4 ||cstate == WR5 ||cstate == WR6 ||cstate == WR7 ||cstate == WR8)?{1'b0,bank,addr}:28'b0;
    //assign  wr_addr =(cstate == RD1 ||cstate == RD2 ||cstate == RD3 ||cstate == RD4 ||cstate == RD5 ||cstate == RD6 ||cstate == RD7 ||cstate == RD8)?{1'b0,bank,addr}:28'b0;
    assign app_addr ={1'b0,bank,addr};
    assign  app_sr_req = 1'b0;
    assign app_ref_req = 1'b0;
    assign app_zq_req  = 1'b0;
    //assign app_addr = (app_cmd ==`CMD_WR && app_en == 1'b1)?wr_addr:rd_addr;
    assign app_wdf_mask = 16'h0000;
```

```
        assign app_wdf_end = app_wdf_wren;
        assign wr1_done = (app_cmd ==`CMD_WR && addr == 28'd800 && bank == 3'b000) ? 1'b1:1'b0;
        assign wr2_done = (app_cmd ==`CMD_WR && addr == 28'd800 && bank == 3'b001) ? 1'b1:1'b0;
        assign wr3_done = (app_cmd ==`CMD_WR && addr == 28'd800 && bank == 3'b010) ? 1'b1:1'b0;
        assign wr4_done = (app_cmd ==`CMD_WR && addr == 28'd800 && bank == 3'b011) ? 1'b1:1'b0;
        assign wr5_done = (app_cmd ==`CMD_WR && addr == 28'd800 && bank == 3'b100) ? 1'b1:1'b0;
        assign wr6_done = (app_cmd ==`CMD_WR && addr == 28'd800 && bank == 3'b101) ? 1'b1:1'b0;
        assign wr7_done = (app_cmd ==`CMD_WR && addr == 28'd800 && bank == 3'b110) ? 1'b1:1'b0;
        assign wr8_done = (app_cmd ==`CMD_WR && addr == 28'd800 && bank == 3'b111) ? 1'b1:1'b0;

        assign rd1_done = (cstate == RD1 && app_cmd ==`CMD_RD && addr == 28'd800) ? 1'b1:1'b0;
        assign rd2_done = (cstate == RD2 && app_cmd ==`CMD_RD && addr == 28'd800) ? 1'b1:1'b0;
        assign rd3_done = (cstate == RD3 && app_cmd ==`CMD_RD && addr == 28'd800) ? 1'b1:1'b0;
        assign rd4_done = (cstate == RD4 && app_cmd ==`CMD_RD && addr == 28'd800) ? 1'b1:1'b0;
        assign rd5_done = (cstate == RD5 && app_cmd ==`CMD_RD && addr == 28'd800) ? 1'b1:1'b0;
        assign rd6_done = (cstate == RD6 && app_cmd ==`CMD_RD && addr == 28'd800) ? 1'b1:1'b0;
        assign rd7_done = (cstate == RD7 && app_cmd ==`CMD_RD && addr == 28'd800) ? 1'b1:1'b0;
        assign rd8_done = (cstate == RD8 && app_cmd ==`CMD_RD && addr == 28'd800) ? 1'b1:1'b0;

        assign done_flag = (cstate == DONE)?1'b1:1'b0;

        assign app_en =(((cstate == WR1 ||cstate == WR2 ||cstate == WR3 ||cstate == WR4 ||cstate == WR5 ||cstate == WR6 ||cstate == WR7 ||cstate == WR8)&& app_wdf_rdy == 1'b1&&app_rdy == 1'b1)||((cstate == RD1 ||cstate == RD2 ||cstate == RD3 ||cstate == RD4 ||cstate == RD5 ||cstate == RD6 ||cstate == RD7 ||cstate == RD8)&& app_rdy == 1'b1)&&((!wr1_done)||(!wr2_done)||(!wr3_done)||(!wr4_done)||(!wr5_done)||(!wr6_done)||(!wr7_done)||(!wr8_done)||(!rd1_d
one)||(!rd2_done)||(!rd3_done)||(!rd4_done)||(!rd5_done)||(!rd6_done)||(!rd7_done)||(!rd8_done)))?1'b1:1'b0;
        assign app_wdf_wren =(((cstate == WR1 ||cstate == WR2 ||cstate == WR3 ||cstate == WR4 ||cstate == WR5 ||cstate == WR6 ||cstate == WR7 ||cstate == WR8) && app_wdf_rdy == 1'b1&&app_rdy == 1'b1)&&((!wr1_done)||(!wr2_done)||(!wr3_done)||(!wr4_done)||(!wr5_done)||(!wr6_done)||(!wr7_done)||(!wr8_done)))?1'b1:1'b0;
        always @(posedge ui_clk or posedge ui_clk_sync_rst) begin
          if(ui_clk_sync_rst == 1'b1)
            cstate <= IDLE;
          else
            cstate <= nstate;
        end

        always @(*) begin
          nstate = IDLE;
          case(cstate)
            IDLE:begin
              if(init_calib_complete == 1'b1)
                nstate = WR1;
              else
                nstate = IDLE;
            end
            WR1:begin
              if(wr1_done == 1'b1)
                nstate = WR2;
              else
                nstate = WR1;
            end
            WR2:begin
              if(wr2_done == 1'b1)
                nstate = WR3;
              else
                nstate = WR2;
            end
            WR3:begin
              if(wr3_done == 1'b1)
                nstate = WR4;
              else
                nstate = WR3;
            end
            WR4:begin
              if(wr4_done == 1'b1)
```

```
                  nstate = WR5;
                else
                  nstate = WR4;
              end
              WR5:begin
                if(wr5_done == 1'b1)
                  nstate = WR6;
                else
                  nstate = WR5;
              end
              WR6:begin
                if(wr6_done == 1'b1)
                  nstate = WR7;
                else
                  nstate = WR6;
              end
              WR7:begin
                if(wr7_done == 1'b1)
                  nstate = WR8;
                else
                  nstate = WR7;
              end
              WR8:begin
                if(wr8_done == 1'b1)
                  nstate = RD1;
                else
                  nstate = WR8;
              end
              RD1:begin
                if(rd1_done == 1'b1)
                  nstate = RD2;
                else
                  nstate = RD1;
               end
               RD2:begin
                 if(rd2_done == 1'b1)
                   nstate = RD3;
                 else
                   nstate = RD2;
               end
               RD3:begin
                  if(rd3_done == 1'b1)
                    nstate = RD4;
                  else
```

```
            nstate = RD3;
       end
       RD4:begin
          if(rd4_done == 1'b1)
            nstate = RD5;
          else
            nstate = RD4;
       end
       RD5:begin
          if(rd5_done == 1'b1)
            nstate = RD6;
          else
            nstate = RD5;
       end
       RD6:begin
          if(rd6_done == 1'b1)
            nstate = RD7;
          else
            nstate = RD6;
       end
       RD7:begin
          if(rd7_done == 1'b1)
            nstate = RD8;
          else
            nstate = RD7;
       end
       RD8:begin
          if(rd8_done == 1'b1)
            nstate = DONE;
          else
            nstate = RD8;
       end
       DONE:begin
          nstate = WR1;
       end
    endcase
  end

  always @(posedge ui_clk or posedge ui_clk_sync_rst) begin
    if(ui_clk_sync_rst == 1'b1) begin
      //wr_addr <= 28'd0; //bank [] row [] col []
      //rd_addr <= 28'b0;
      app_cmd <= `CMD_WR;
```

```
            app_wdf_data <= 128'b0;

            bank <= 3'b000;
            //row  <= 14'b0;
            //column <= 10'b0;
            addr <=24'b0;
          end
          else
            case(cstate)
              WR1:begin
                app_cmd <= `CMD_WR;
                bank <= 3'b000;
                if(wr1_done == 1'b1) begin
                  addr <=24'b0;
                  app_wdf_data <= 128'b0;
                end
                else if(app_wdf_rdy == 1'b1&&app_rdy == 1'b1) begin
                  //wr_addr <= wr_addr + 8;
                  //row  <= 14'b0;
                  //column <= column+8;
                  addr <= addr +8;
                  app_wdf_data <= app_wdf_data +1;
                end
                else begin
                  addr <= addr;
                  app_wdf_data <= app_wdf_data;
                end
              end
              WR2:begin
                app_cmd <= `CMD_WR;
                bank <= 3'b001;
               if(wr2_done == 1'b1) begin
                  addr <=24'b0;
                  app_wdf_data <= 128'b0;
                end
                else if(app_wdf_rdy == 1'b1&&app_rdy == 1'b1) begin
                  //wr_addr <= wr_addr + 8;
                  //row  <= 14'b0;
                  //column <= column+8;
                  addr <= addr +8;
                  app_wdf_data <= app_wdf_data +1;
                end
```

```
       else begin
        addr <= addr;
        app_wdf_data <= app_wdf_data;
       end
      end
      WR3:begin
       app_cmd <= `CMD_WR;
       bank <= 3'b010;
       if(wr3_done == 1'b1) begin
        addr <=24'b0;
        app_wdf_data <= 128'b0;
       end
       else if(app_wdf_rdy == 1'b1&&app_rdy == 1'b1) begin
        //wr_addr <= wr_addr + 8;
        //row  <= 14'b0;
        //column <= column+8;
        addr <= addr +8;
        app_wdf_data <= app_wdf_data +1;
       end
       else begin
        addr <= addr;
        app_wdf_data <= app_wdf_data;
       end
      end
      WR4:begin
       app_cmd <= `CMD_WR;
       bank <= 3'b011;
       if(wr4_done == 1'b1) begin
        addr <=24'b0;
        app_wdf_data <= 128'b0;
       end
       else if(app_wdf_rdy == 1'b1&&app_rdy == 1'b1) begin
        //wr_addr <= wr_addr + 8;
        //row  <= 14'b0;
        //column <= column+8;
        addr <= addr +8;
        app_wdf_data <= app_wdf_data +1;
       end
       else begin
        addr <= addr;
        app_wdf_data <= app_wdf_data;
       end
```

```
            end
            WR5:begin
              app_cmd <= `CMD_WR;
              bank <= 3'b100;
              if(wr5_done == 1'b1) begin
                addr <=24'b0;
                app_wdf_data <= 128'b0;
              end
              else if(app_wdf_rdy == 1'b1&&app_rdy == 1'b1) begin
                //wr_addr <= wr_addr + 8;
                //row  <= 14'b0;
                //column <= column+8;
                addr <= addr +8;
                app_wdf_data <= app_wdf_data +1;
              end
              else begin
                addr <= addr;
                app_wdf_data <= app_wdf_data;
              end
            end
            WR6:begin
              app_cmd <= `CMD_WR;
              bank <= 3'b101;
             if(wr6_done == 1'b1) begin
                addr <=24'b0;
                app_wdf_data <= 128'b0;
              end
              else if(app_wdf_rdy == 1'b1&&app_rdy == 1'b1) begin
                //wr_addr <= wr_addr + 8;
                //row  <= 14'b0;
                //column <= column+8;
                addr <= addr +8;
                app_wdf_data <= app_wdf_data +1;
              end
              else begin
                addr <= addr;
                app_wdf_data <= app_wdf_data;
              end
            end
            WR7:begin
              app_cmd <= `CMD_WR;
              bank <= 3'b110;
```

```
         if(wr7_done == 1'b1) begin
            addr <=24'b0;
            app_wdf_data <= 128'b0;
          end
          else if(app_wdf_rdy == 1'b1&&app_rdy == 1'b1) begin
            //wr_addr <= wr_addr + 8;
            //row   <= 14'b0;
            //column <= column+8;
            addr <= addr +8;
            app_wdf_data <= app_wdf_data +1;
          end
          else begin
            addr <= addr;
            app_wdf_data <= app_wdf_data;
          end
        end
        WR8:begin
          app_cmd <= `CMD_WR;
          bank <= 3'b111;
          if(wr8_done == 1'b1) begin
            addr <=24'b0;
            app_wdf_data <= 128'b0;
          end
          else if(app_wdf_rdy == 1'b1&&app_rdy == 1'b1) begin
            //wr_addr <= wr_addr + 8;
            //row   <= 14'b0;
            //column <= column+8;
            addr <= addr +8;
            app_wdf_data <= app_wdf_data +1;
          end
          else begin
            addr <= addr;
            app_wdf_data <= app_wdf_data;
          end
        end
        RD1:begin
          app_cmd <= `CMD_RD;
          bank <= 3'b000;
          if(rd1_done == 1'b1) begin
            addr <= 24'b0;
          end
          else if(app_rdy == 1'b1)begin
```

```
                   addr <= addr+8;
                end
                else begin
                  addr <= addr;
                end
              end
              RD2:begin
                app_cmd <= `CMD_RD;
                bank <= 3'b001;
                if(rd2_done == 1'b1) begin
                  addr <= 24'b0;
                end
                else if(app_rdy == 1'b1)begin
                   addr <= addr+8;
                end
                else begin
                  addr <= addr;
                end
              end
              RD3:begin
                app_cmd <= `CMD_RD;
                bank <= 3'b010;
                if(rd3_done == 1'b1) begin
                  addr <= 24'b0;
                end
                else if(app_rdy == 1'b1)begin
                   addr <= addr+8;
                end
                else begin
                  addr <= addr;
                end
              end
              RD4:begin
                app_cmd <= `CMD_RD;
                bank <= 3'b011;
                if(rd4_done == 1'b1) begin
                  addr <= 24'b0;
                end
                else if(app_rdy == 1'b1)begin
                   addr <= addr+8;
                end
                else begin
                  addr <= addr;
                end
```

```
        end
        RD5:begin
          app_cmd <= `CMD_RD;
          bank <= 3'b100;
          if(rd5_done == 1'b1) begin
            addr <= 24'b0;
          end
          else if(app_rdy == 1'b1)begin
             addr <= addr+8;
          end
          else begin
            addr <= addr;
          end
        end
        RD6:begin
          app_cmd <= `CMD_RD;
          bank <= 3'b101;
          if(rd6_done == 1'b1) begin
            addr <= 24'b0;
          end
          else if(app_rdy == 1'b1)begin
             addr <= addr+8;
          end
          else begin
            addr <= addr;
          end
        end
        RD7:begin
          app_cmd <= `CMD_RD;
          bank <= 3'b110;
          if(rd7_done == 1'b1) begin
            addr <= 24'b0;
          end
          else if(app_rdy == 1'b1)begin
             addr <= addr+8;
          end
          else begin
            addr <= addr;
          end
        end
        RD8:begin
          app_cmd <= `CMD_RD;
          bank <= 3'b111;
          if(rd8_done == 1'b1) begin
```

```
                    addr <= 24'b0;
                  end
                  else if(app_rdy == 1'b1)begin
                     addr <= addr+8;
                  end
                  else begin
                    addr <= addr;
                  end
                end
                DONE:begin
                  //wr_addr <= 28'd0; //bank [] row [] col []
                  //rd_addr <= 28'b0;
                  bank <= 3'b000;
                  //row  <= 14'b0;
                  //column <= 10'b0;
                  addr <=24'b0;
                  app_cmd <= `CMD_WR;
                  app_wdf_data <= 128'b0;
                end
              endcase
        end


clk_wiz_0  U_clk_wiz_0(
        // Clock in ports
         .clk_in1(clk),
         // Clock out ports
         .clk_out1(sys_clk_i),
         .clk_out2(clk_ref_i),
         // Status and control signals
         .reset(reset),
         .locked(sys_rst_n)
        );
       //*******************************************************
********************

ddr3_ip u_ddr3_ip
                 (
       // Memory interface ports
              .ddr3_addr                      (ddr3_addr),
              .ddr3_ba                        (ddr3_ba),
              .ddr3_cas_n                     (ddr3_cas_n),
              .ddr3_ck_n                      (ddr3_ck_n),
              .ddr3_ck_p                      (ddr3_ck_p),
                .ddr3_cke                       (ddr3_cke),
                .ddr3_ras_n                     (ddr3_ras_n),
                .ddr3_we_n                      (ddr3_we_n),
                .ddr3_dq                        (ddr3_dq),
                .ddr3_dqs_n                     (ddr3_dqs_n),
                .ddr3_dqs_p                     (ddr3_dqs_p),
                .ddr3_reset_n                   (ddr3_reset_n),
                .init_calib_complete            (init_calib_complete),

                .ddr3_cs_n                      (ddr3_cs_n),
                .ddr3_dm                        (ddr3_dm),
                .ddr3_odt                       (ddr3_odt),
        // Application interface ports
                .app_addr                       (app_addr),
                .app_cmd                        (app_cmd),
                .app_en                         (app_en),
                .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_wdf_rdy                    (app_wdf_rdy),
                .app_sr_req                     (app_sr_req),
                .app_ref_req                    (app_ref_req),
                .app_zq_req                     (app_zq_req),
                .app_sr_active                  (app_sr_active),
                .app_ref_ack                    (app_ref_ack),
                .app_zq_ack                     (app_zq_ack),
                .ui_clk                         (ui_clk),
                .ui_clk_sync_rst                (ui_clk_sync_rst),


                .app_wdf_mask                   (app_wdf_mask),


        // System Clock Ports
                .sys_clk_i                       (sys_clk_i),
        // Reference Clock Ports
                .clk_ref_i                      (clk_ref_i),

                .sys_rst                         (sys_rst)
                );

ila_0 U_ila(
               .clk(ui_clk),


               .probe0(app_addr),
               .probe1(app_cmd),
               .probe2(app_en),
               .probe3(app_wdf_data),
               .probe4(app_wdf_end),
               .probe5(app_wdf_mask),
               .probe6(app_wdf_wren),
               .probe7(app_rd_data),
               .probe8(app_rd_data_end),
               .probe9(app_rd_data_valid),
               .probe10(app_rdy),
               .probe11(app_wdf_rdy),
               .probe12(app_sr_req),
               .probe13(app_ref_req),
               .probe14(app_zq_req),
               .probe15(app_sr_active)
               );
endmodule
```

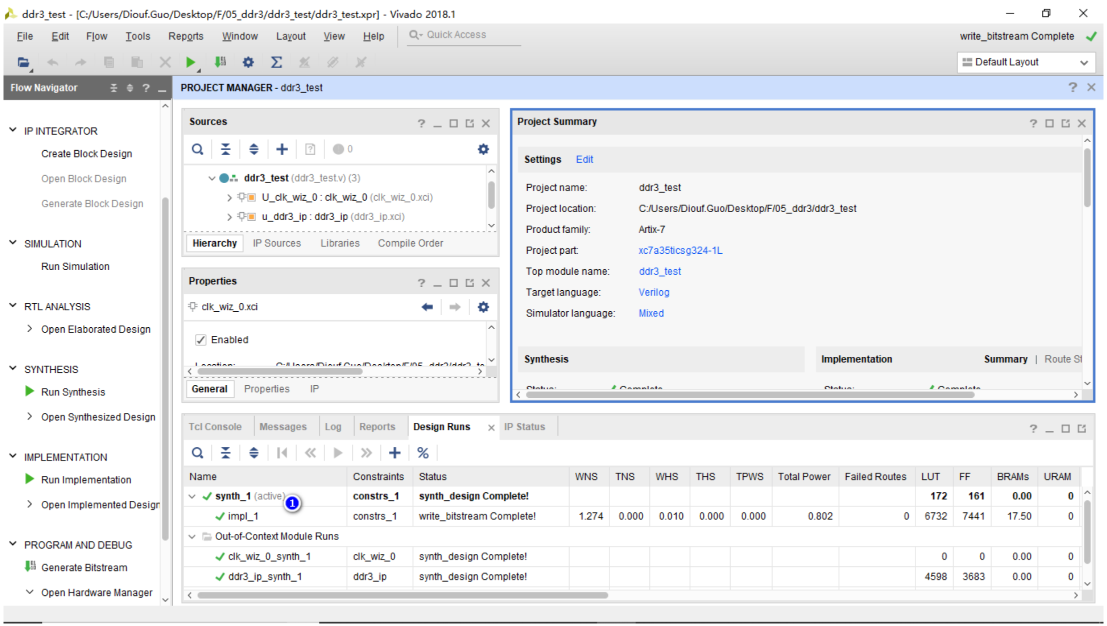
ddr3.xdc

```
set_property PACKAGE_PIN E3 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports clk]
set_property PACKAGE_PIN D9 [get_ports reset]
set_property PACKAGE_PIN E1 [get_ports init_calib_complete]
set_property IOSTANDARD LVCMOS33 [get_ports init_calib_complete]

set_property IOSTANDARD LVCMOS33 [get_ports reset]
```

2 下板实现读写时序

1>①完成综合和实现



ddr3_test - [C:/Users/Diouf.Guo/Desktop/F/05_ddr3/ddr3_test/ddr3_test.xpr] - Vivado 2018.1

PROJECT MANAGER - ddr3_test

Sources: ddr3_test (ddr3_test.v) (3); U_clk_wiz_0 : clk_wiz_0 (clk_wiz_0.xci); u_ddr3_ip : ddr3_ip (ddr3_ip.xci)

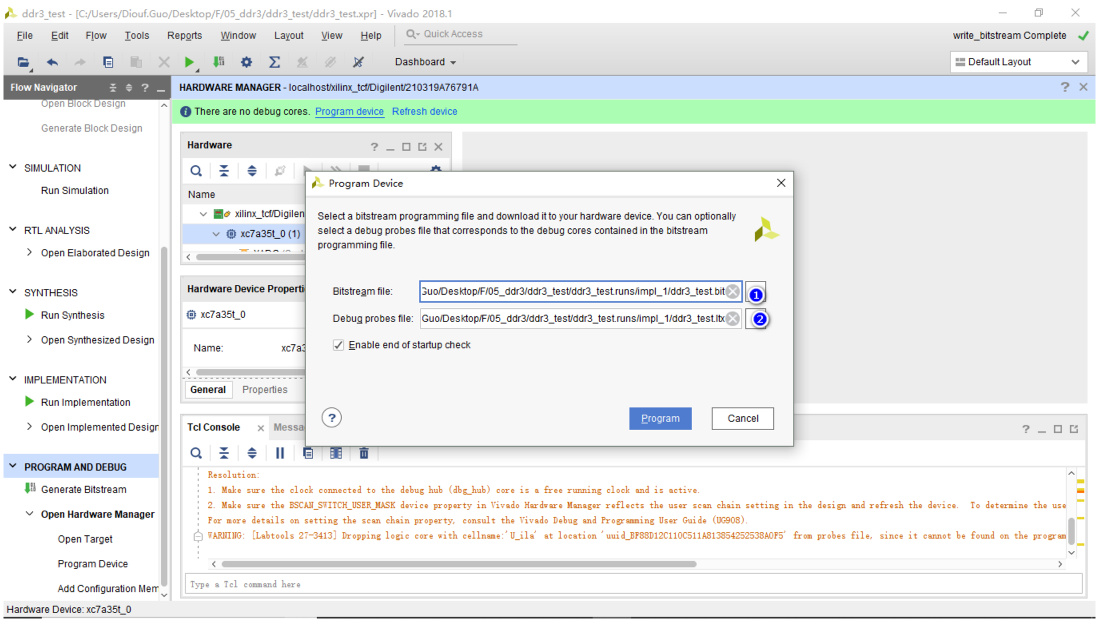
Project Summary

Project name:	ddr3_test
Project location:	C:/Users/Diouf.Guo/Desktop/F/05_ddr3/ddr3_test
Product family:	Artix-7
Project part:	xc7a35ticsg324-1L
Top module name:	ddr3_test
Target language:	Verilog
Simulator language:	Mixed

Design Runs

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMs	URAM
synth_1 (active)	constrs_1	synth_design Complete!								172	161	0.00	0
impl_1	constrs_1	write_bitstream Complete!	1.274	0.000	0.010	0.000	0.000	0.802	0	6732	7441	17.50	0
Out-of-Context Module Runs													
clk_wiz_0_synth_1	clk_wiz_0	synth_design Complete!								0	0	0.00	0
ddr3_ip_synth_1	ddr3_ip	synth_design Complete!								4598	3683	0.00	0

2>下载 bit 文件和 debug 文件。



HARDWARE MANAGER - localhost/xilinx_tcf/Digilent/210319A76791A

There are no debug cores. Program device Refresh device

Program Device

Select a bitstream programming file and download it to your hardware device. You can optionally select a debug probes file that corresponds to the debug cores contained in the bitstream programming file.

Bitstream file: Guo/Desktop/F/05_ddr3/ddr3_test/ddr3_test.runs/impl_1/ddr3_test.bit

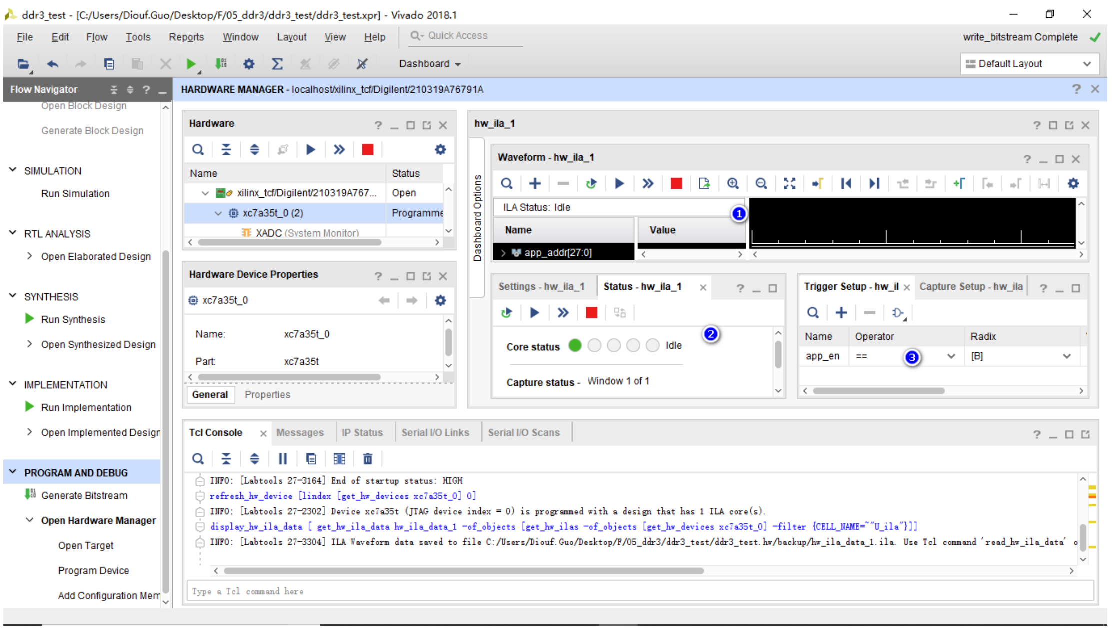
Debug probes file: Guo/Desktop/F/05_ddr3/ddr3_test/ddr3_test.runs/impl_1/ddr3_test.ltx

☑ Enable end of startup check

```
Resolution:
1. Make sure the clock connected to the debug hub (dbg_hub) core is a free running clock and is active.
2. Make sure the BSCAN_SWITCH_USER_MASK device property in Vivado Hardware Manager reflects the user scan chain setting in the design and refresh the device.  To determine the use
For more details on setting the scan chain property, consult the Vivado Debug and Programming User Guide (UG908).
WARNING: [Labtools 27-3413] Dropping logic core with cellname:'U_ila' at location 'uuid_BF88D12C110C511A813854252538A0F5' from probes file, since it cannot be found on the program
```

Hardware Device: xc7a35t_0

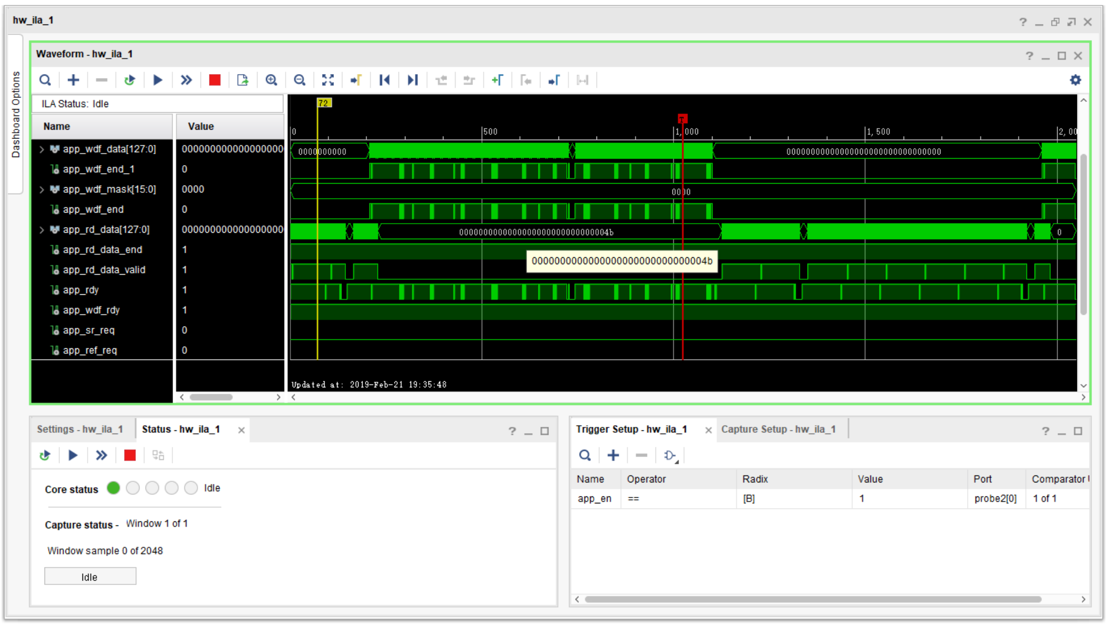
3>下载完成，查看波形。



ddr3_test - [C:/Users/Diouf.Guo/Desktop/F/05_ddr3/ddr3_test/ddr3_test.xpr] - Vivado 2018.1

HARDWARE MANAGER - localhost/xilinx_tcf/Digilent/210319A76791A

```
INFO: [Labtools 27-3164] End of startup status: HIGH
refresh_hw_device [lindex [get_hw_devices xc7a35t_0] 0]
INFO: [Labtools 27-2302] Device xc7a35t (JTAG device index = 0) is programmed with a design that has 1 ILA core(s).
display_hw_ila_data [ get_hw_ila_data hw_ila_data_1 -of_objects [get_hw_ilas -of_objects [get_hw_devices xc7a35t_0] -filter {CELL_NAME=~"U_ila"}]]
INFO: [Labtools 27-3304] ILA Waveform data saved to file C:/Users/Diouf.Guo/Desktop/F/05_ddr3/ddr3_test/ddr3_test.hw/backup/hw_ila_data_1.ila. Use Tcl command 'read_hw_ila_data' o
```

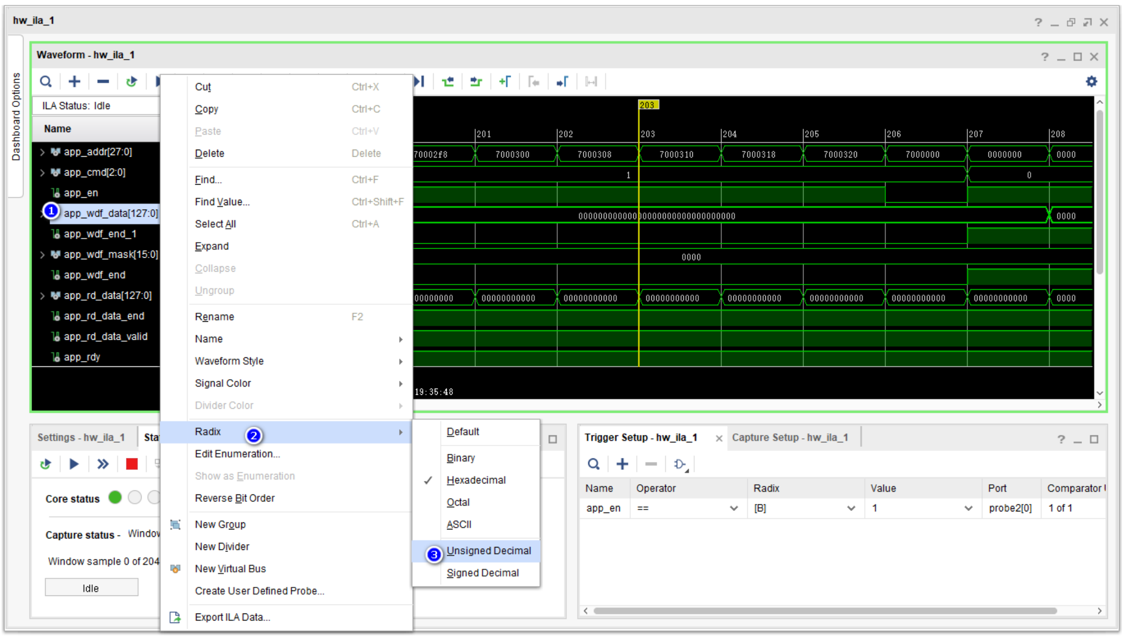


Name	Value
app_wdf_data[127:0]	00000000000000000
app_wdf_end_1	0
app_wdf_mask[15:0]	0000
app_wdf_end	0
app_rd_data[127:0]	00000000000000000
app_rd_data_end	1
app_rd_data_valid	1
app_rdy	1
app_wdf_rdy	1
app_sr_req	0
app_ref_req	0

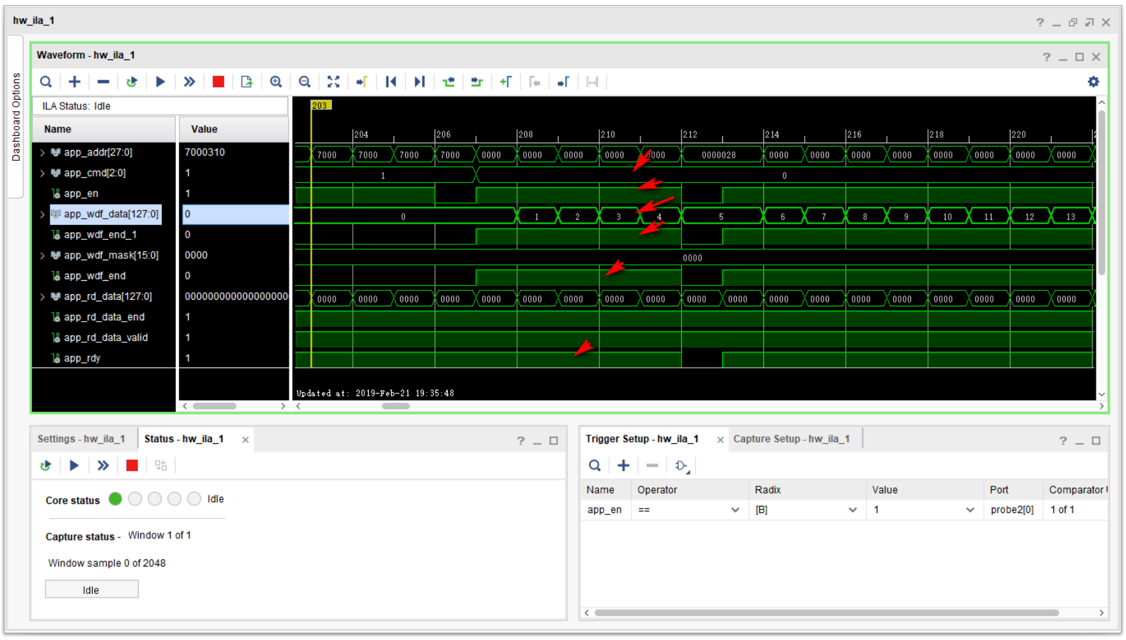
Updated at: 2019-Feb-21 19:35:48

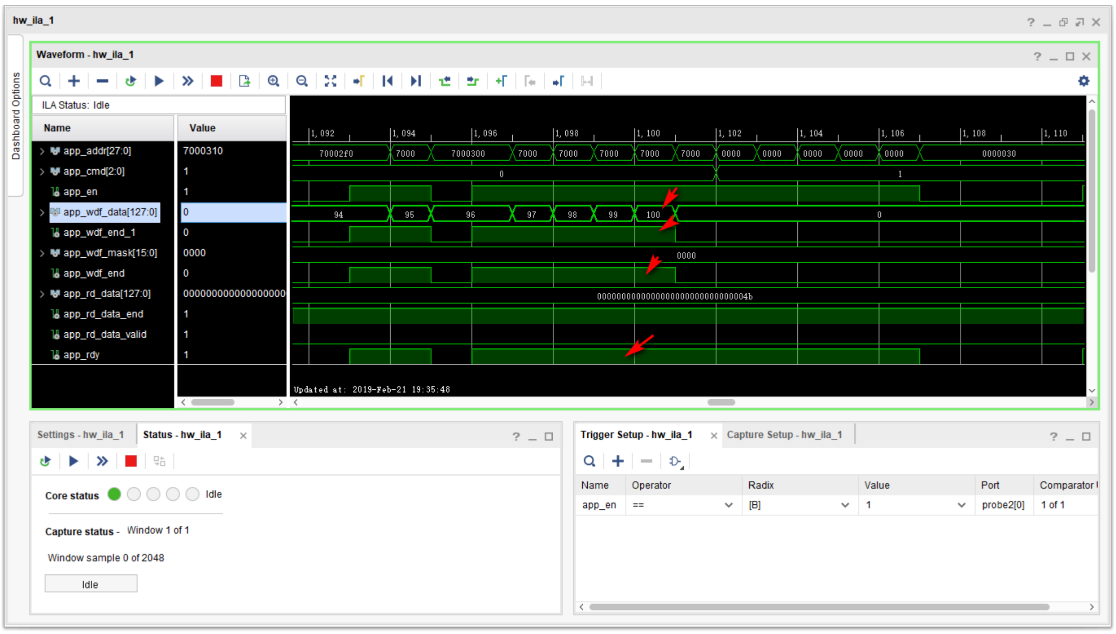
Name	Operator	Radix	Value	Port	Comparator
app_en	==	[B]	1	probe2[0]	1 of 1

4>将 app_wdf_data 数据格式改为 Unsigned Decimal。


hw_ila_1
Waveform - hw_ila_1
ILA Status: Idle
Name
app_addr[27:0]
app_cmd[2:0]
app_en
app_wdf_data[127:0]
app_wdf_end_1
app_wdf_mask[15:0]
app_wdf_end
app_rd_data[127:0]
app_rd_data_end
app_rd_data_valid
app_rdy
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Delete Delete
Find... Ctrl+F
Find Value... Ctrl+Shift+F
Select All Ctrl+A
Expand
Collapse
Ungroup
Rename F2
Name
Waveform Style
Signal Color
Divider Color
Radix
Edit Enumeration...
Show as Enumeration
Reverse Bit Order
New Group
New Divider
New Virtual Bus
Create User Defined Probe...
Export ILA Data...
Default
Binary
Hexadecimal
Octal
ASCII
Unsigned Decimal
Signed Decimal
1
2
3
Settings - hw_ila_1
Core status
Capture status - Window
Window sample 0 of 204
Idle
Trigger Setup - hw_ila_1
Capture Setup - hw_ila_1
Name Operator Radix Value Port Comparator
app_en == [B] 1 probe2[0] 1 of 1
19:35:48

5>查看写时序。


hw_ila_1
Waveform - hw_ila_1
ILA Status: Idle
Name
Value
app_addr[27:0] 7000310
app_cmd[2:0] 1
app_en 1
app_wdf_data[127:0] 0
app_wdf_end_1 0
app_wdf_mask[15:0] 0000
app_wdf_end 0
app_rd_data[127:0] 0000000000000000
app_rd_data_end 1
app_rd_data_valid 1
app_rdy 1
Updated at: 2019-Feb-21 19:35:48
Settings - hw_ila_1
Status - hw_ila_1
Core status Idle
Capture status - Window 1 of 1
Window sample 0 of 2048
Idle
Trigger Setup - hw_ila_1
Capture Setup - hw_ila_1
Name Operator Radix Value Port Comparator
app_en == [B] 1 probe2[0] 1 of 1



hw_ila_1

Waveform - hw_ila_1

ILA Status: Idle

Name	Value
app_addr[27:0]	7000310
app_cmd[2:0]	1
app_en	1
app_wdf_data[127:0]	0
app_wdf_end_1	0
app_wdf_mask[15:0]	0000
app_wdf_end	0
app_rd_data[127:0]	0000000000000000
app_rd_data_end	1
app_rd_data_valid	1
app_rdy	1

Updated at: 2019-Feb-21 19:35:48

Settings - hw_ila_1 | Status - hw_ila_1

Core status: Idle

Capture status - Window 1 of 1

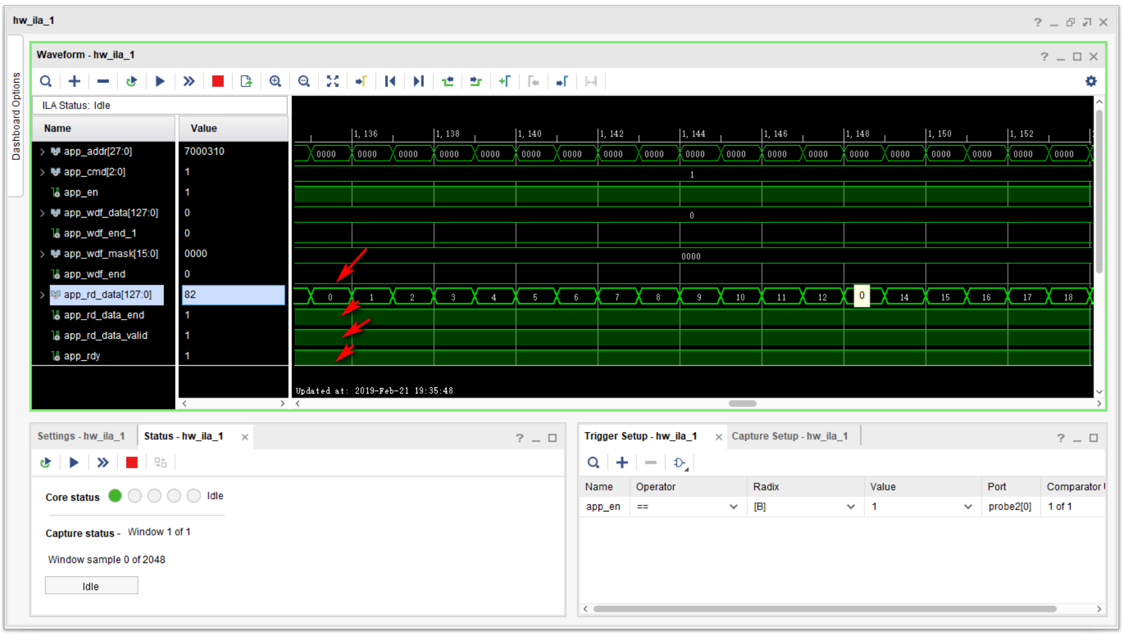
Window sample 0 of 2048

Idle

Trigger Setup - hw_ila_1 | Capture Setup - hw_ila_1

Name	Operator	Radix	Value	Port	Comparator
app_en	==	[B]	1	probe2[0]	1 of 1

6>查看读时序。



hw_ila_1

Waveform - hw_ila_1

ILA Status: Idle

Name	Value
app_addr[27:0]	7000310
app_cmd[2:0]	1
app_en	1
app_wdf_data[127:0]	0
app_wdf_end_1	0
app_wdf_mask[15:0]	0000
app_wdf_end	0
app_rd_data[127:0]	82
app_rd_data_end	1
app_rd_data_valid	1
app_rdy	1

Updated at: 2019-Feb-21 19:35:48

Settings - hw_ila_1 | Status - hw_ila_1

Core status: Idle

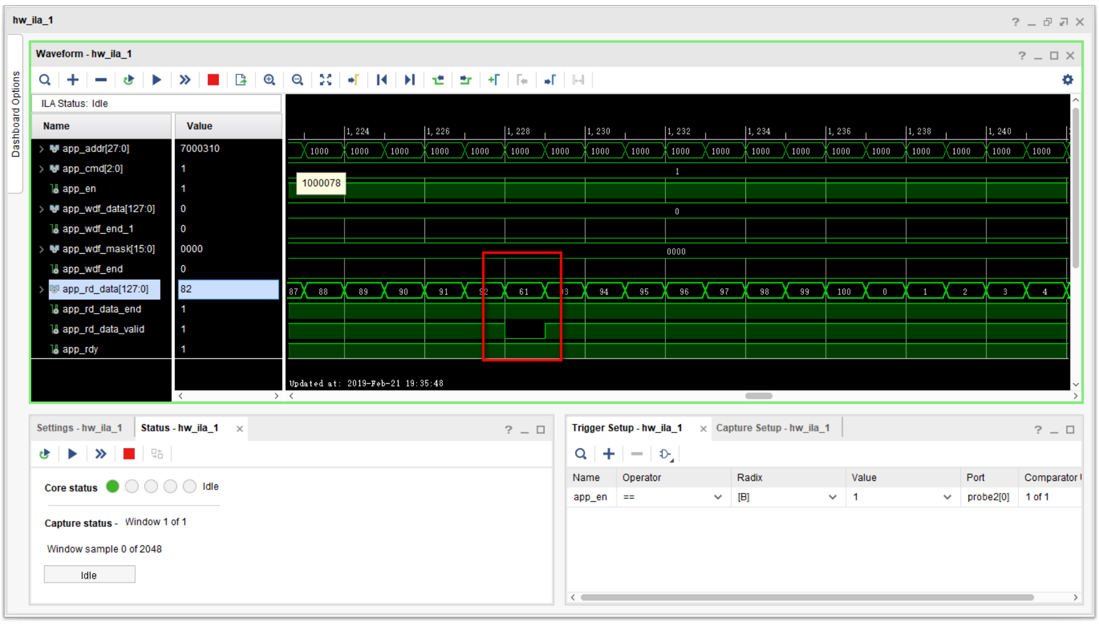
Capture status - Window 1 of 1

Window sample 0 of 2048

Idle

Trigger Setup - hw_ila_1 | Capture Setup - hw_ila_1

Name	Operator	Radix	Value	Port	Comparator
app_en	==	[B]	1	probe2[0]	1 of 1


hw_ila_1

Waveform - hw_ila_1

ILA Status: Idle

Name	Value
app_addr[27:0]	7000310
app_cmd[2:0]	1
app_en	1
app_wdf_data[127:0]	0
app_wdf_end_1	0
app_wdf_mask[15:0]	0000
app_wdf_end	0
app_rd_data[127:0]	82
app_rd_data_end	1
app_rd_data_valid	1
app_rdy	1

Updated at: 2019-Feb-21 19:35:48

Settings - hw_ila_1 | Status - hw_ila_1

Core status Idle

Capture status - Window 1 of 1

Window sample 0 of 2048

Idle

Trigger Setup - hw_ila_1 | Capture Setup - hw_ila_1

Name	Operator	Radix	Value	Port	Comparator
app_en	==	[B]	1	probe2[0]	1 of 1

基于 xilinx mig ip 对 ddr3 读写的验证完成。学会 ddr3 接口的读写控制是每个 FPGA 工程师必须掌握的技能（几乎所有的公司都会要求），暂时没时间学习的同学可以在 FPGA 开源工作室（微信公众号）下回复“DDR3”获取 xilinx mig ip 使用的五篇 pdf 文档，以备不时之需。

FPGA 开源工作室为了大家更好更快的学习 FPGA 目前开通了知识星球（FPGA 自习学院）。FPGA 自习学院将不断更新和总结 FPGA 相关的学习资料，欢迎大家加入，一起学习一起成长。