HW#4 Adding a DRAM Controller

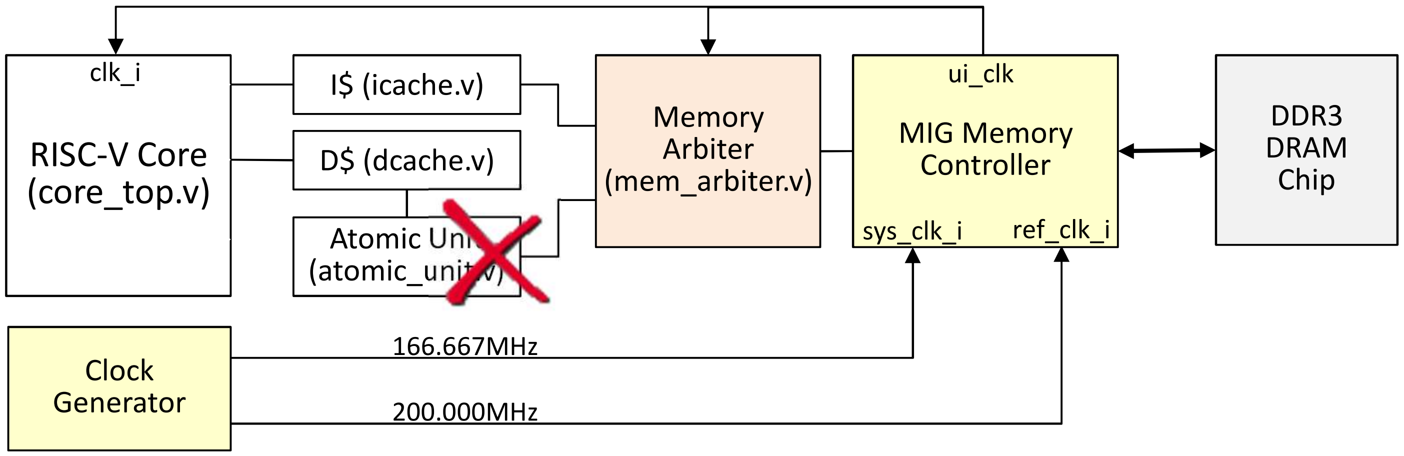
Chun-Jen Tsai
National Chiao Tung University
12/2/2020

Homework Goal

- Starting in this homework, we will connect Aquila to the memory controller to access DRAM:
  - Synthesis of the system will cause timing violation
  - The system still runs correctly since most digital chips can be overclocked a little bit), but this is not a good practice
  - Your job is to fix this timing violation
- For this homework, you should download a new aquila package from E3
- You should upload a report to E3 on how you fix the timing violation by 12/14, 17:00.

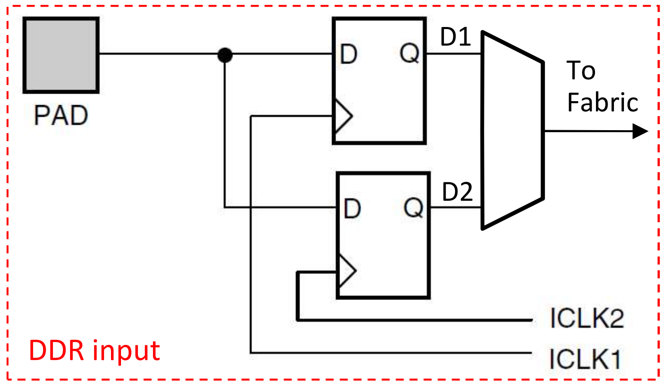
DRAM Specification on Arty

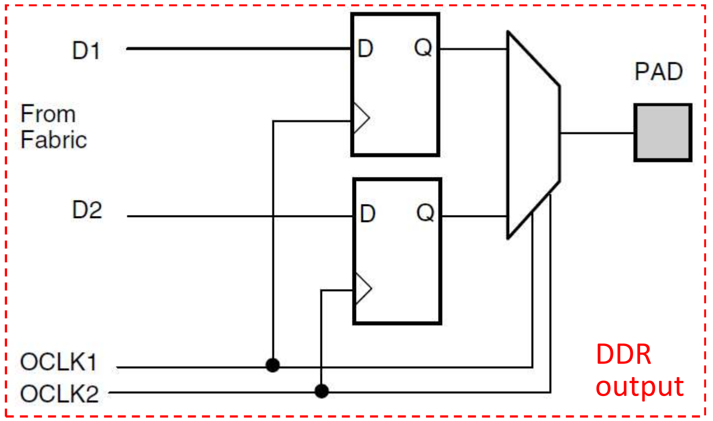
- The DRAM Chip on Arty is a Micron *MT41K128M16JT-125*
  - The chip is a 16-bit DDR3-667 component, clocked at 333MHz
  - To support 333Mhz DRAM clock, the memory controller must run at 333/4 = 83.333 MHz or 333/2 = 166.667 MHz
- Both instruction and data memory shares the same DRAM, so we must add an arbiter to the system:



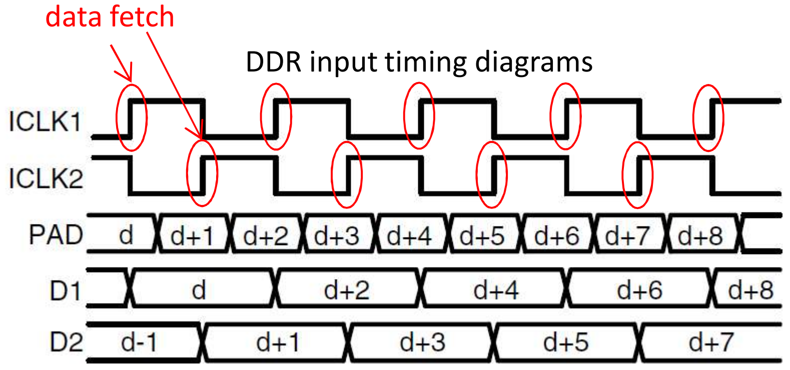
DDRx Memory Controller (1/2)

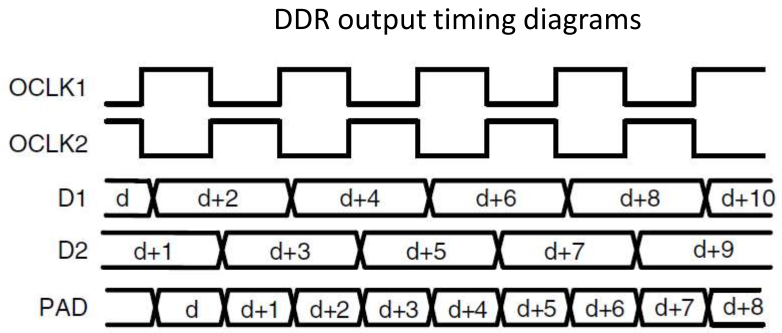
- We can design a low-speed DRAM controller and connect it to a DRAM chip with generic FPGA user pins





CLK1 and CLK2 are 180° phase shifted





DDRx Memory Controller (2/2)

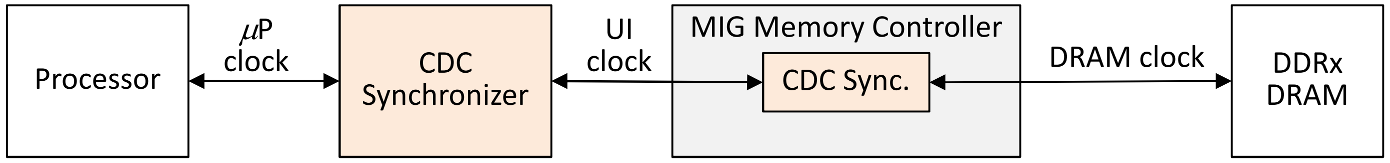
- High-speed DDRx memory controller IPs are complicated and requires some dedicated I/O logic
  - Only certain FPGA I/O banks can be used to connect to the high-speed DRAM chips
  - The memory controller also requires special logic to talk to the DRAM chips
- Xilinx solution for memory controllers
  - Xilinx provides a configurable Memory Interface Generator (MIG) that can be used to generate a memory controller
  - The available DRAM parameters depends on the FPGA family
    - On Kintex devices, DRAM clock up to 800MHz (DDR3-1600)
    - On Artix devices, DRAM clock up to 400MHz (DDR3-800)

MIG Interface on Processor Side

- MIG support two types of processor side interface:
  - AXI interface – easier to use if your processor has AXI-compatible memory ports
  - Native interface – close to the real DRAM chip interface, more efficient to use, but your logic must handle the DRAM burst re-ordering and the large access block issues.
- In Aquila for homework, we choose to use the native MIG interface since:
  - Aquila has I-cache and D-cache so we always access the DRAM on a block basis (128-bit at a time)
  - Burst ordering issue is not hard to handle, we do that in the memory arbiter

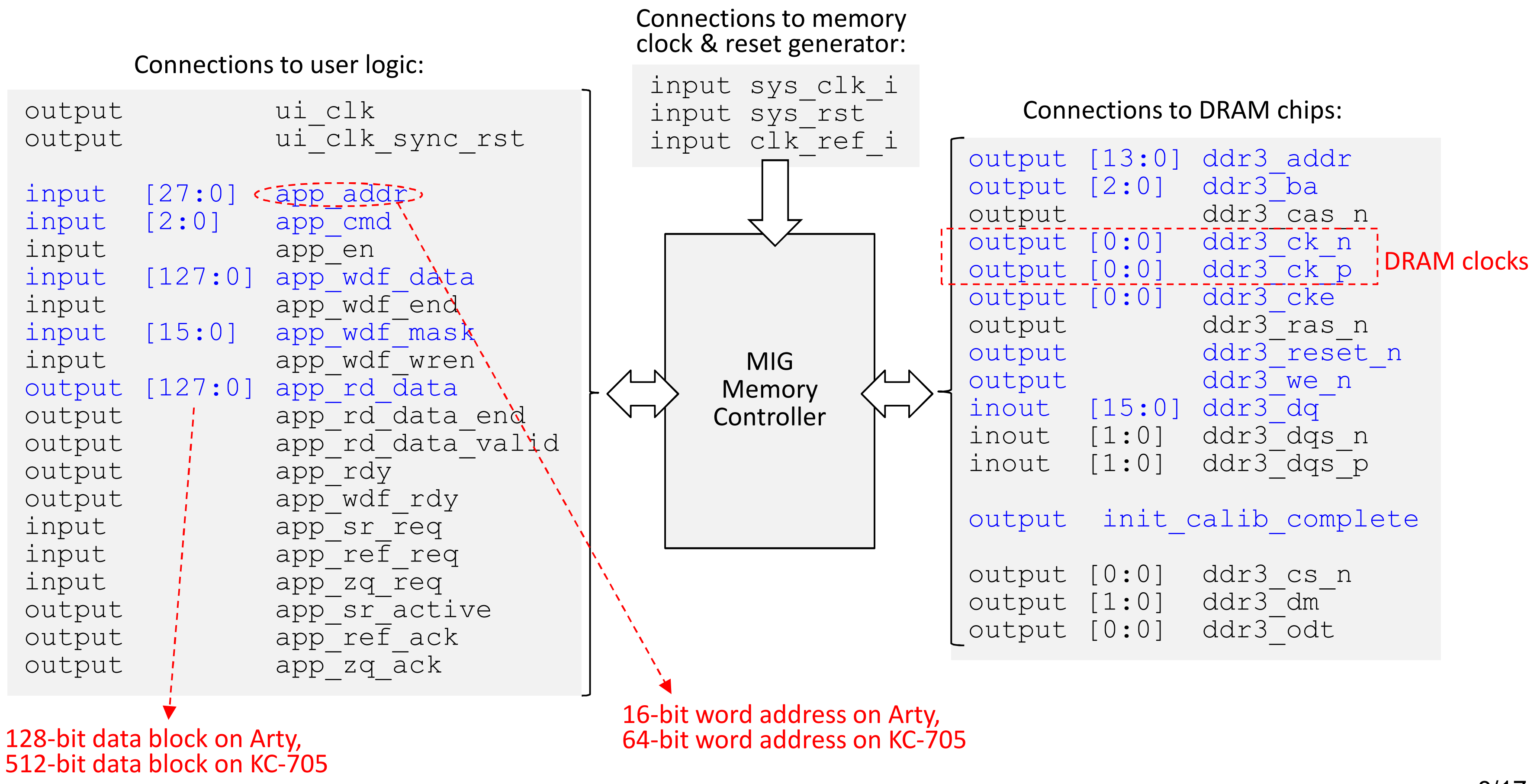
Cross Clock Domain (CDC) Design

- The memory controller generated by MIG is a cross-clock domain IP
  - On DRAM side, it runs at `sys_clk` rate (166.667MHz on Arty)
  - On processor side, it runs at `ui_clk` rate (83.333MHz on Arty)
- If `ui_clk` is too high for the processor, we must produce a slower clock for the processor core
  - In this case, a CDC synchronizer module must be used to connect the processor to the memory controller:



DRAM Native Interface

- ❑ Two clock domains of MIG: DRAM & UI (user interface)


Connections to user logic:
Connections to memory clock & reset generator:
Connections to DRAM chips:
MIG Memory Controller
DRAM clocks
128-bit data block on Arty, 512-bit data block on KC-705
16-bit word address on Arty, 64-bit word address on KC-705

Connections to user logic:

```
output            ui_clk
output            ui_clk_sync_rst

input   [27:0]    app_addr
input   [2:0]     app_cmd
input             app_en
input   [127:0]   app_wdf_data
input             app_wdf_end
input   [15:0]    app_wdf_mask
input             app_wdf_wren
output  [127:0]   app_rd_data
output            app_rd_data_end
output            app_rd_data_valid
output            app_rdy
output            app_wdf_rdy
input             app_sr_req
input             app_ref_req
input             app_zq_req
output            app_sr_active
output            app_ref_ack
output            app_zq_ack
```

Connections to memory clock & reset generator:

```
input sys_clk_i
input sys_rst
input clk_ref_i
```

Connections to DRAM chips:

```
output  [13:0]  ddr3_addr
output  [2:0]   ddr3_ba
output          ddr3_cas_n
output  [0:0]   ddr3_ck_n
output  [0:0]   ddr3_ck_p
output  [0:0]   ddr3_cke
output          ddr3_ras_n
output          ddr3_reset_n
output          ddr3_we_n
inout   [15:0]  ddr3_dq
inout   [1:0]   ddr3_dqs_n
inout   [1:0]   ddr3_dqs_p

output   init_calib_complete

output  [0:0]   ddr3_cs_n
output  [1:0]   ddr3_dm
output  [0:0]   ddr3_odt
```

Block-based I/O of Memory Controller

- DRAM chips typically operates on a row basis, each read/write operation will be on a row of memory cells
  - The memory controller will read/write a large block at one time
- On Arty, MIG read/write 128-bit data at a time
  - You specify the 16-bit starting word addresses, the memory controller will read 128-bit data that contains the data in the same row of DRAM cells
  - For writing, a mask can be used to specify the words you want to modify

Data Reordering of Transaction Data

- MIG is hardwired to read/write 8-word burst each time
  - However, DRAM chips output 4-word wrapping burst each time
  - The least significant word contains the `[app_addr]` data
  - For efficiency, a read burst returns data out-of-order
- On Arty, the following logic is used to re-order the data back to normal order (not really necessary for Aquila):

```
always @(posedge clk_i) begin
    if (rst_i) read_data  <= {128{1'b0}};
    else if (read_data_valid_i)
        case(addr_o[2:0])
        3'h0: read_data  <= {word7, word6, word5, word4, word3, word2, word1, word0};
        3'h1: read_data  <= {word6, word5, word4, word7, word2, word1, word0, word3};
        3'h2: read_data  <= {word5, word4, word7, word6, word1, word0, word3, word2};
        3'h3: read_data  <= {word4, word7, word6, word5, word0, word3, word2, word1};
        3'h4: read_data  <= {word3, word2, word1, word0, word7, word6, word5, word4};
        3'h5: read_data  <= {word2, word1, word0, word3, word6, word5, word4, word7};
        3'h6: read_data  <= {word1, word0, word3, word2, word5, word4, word7, word6};
        3'h7: read_data  <= {word0, word3, word2, word1, word4, word7, word6, word5};
        endcase
end
```

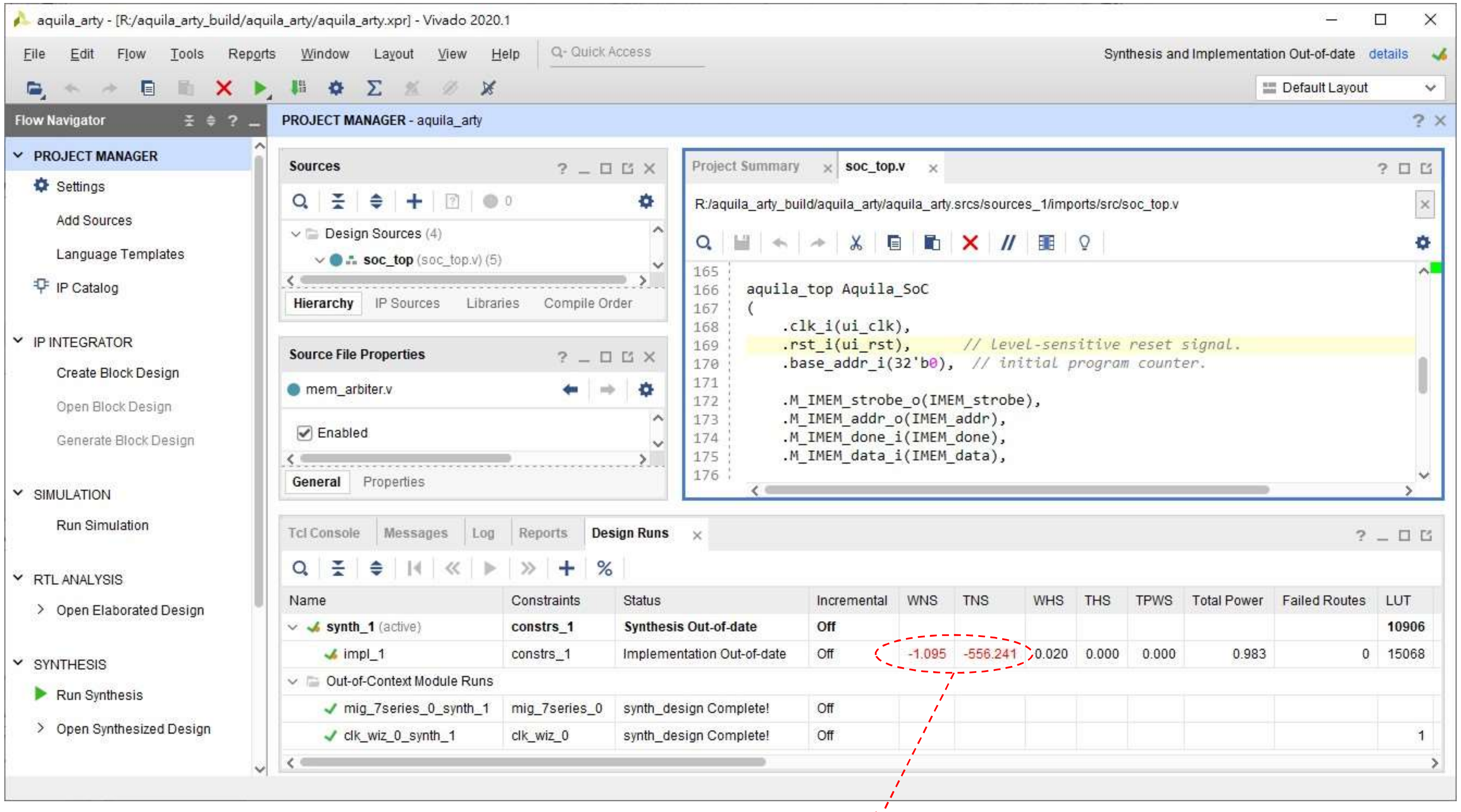
2nd 4-word wrapping burst

1st 4-word wrapping burst

2-to-1 Memory Arbitration

- Since Aquila has two memory ports (I-Mem & D-Mem) that accesses the DRAM, an 2-to-1 multiplexor must be used to share the memory controller port
- For Aquila, instruction fetch has higher priority over data accesses

Ooops, We Have a Timing Violation



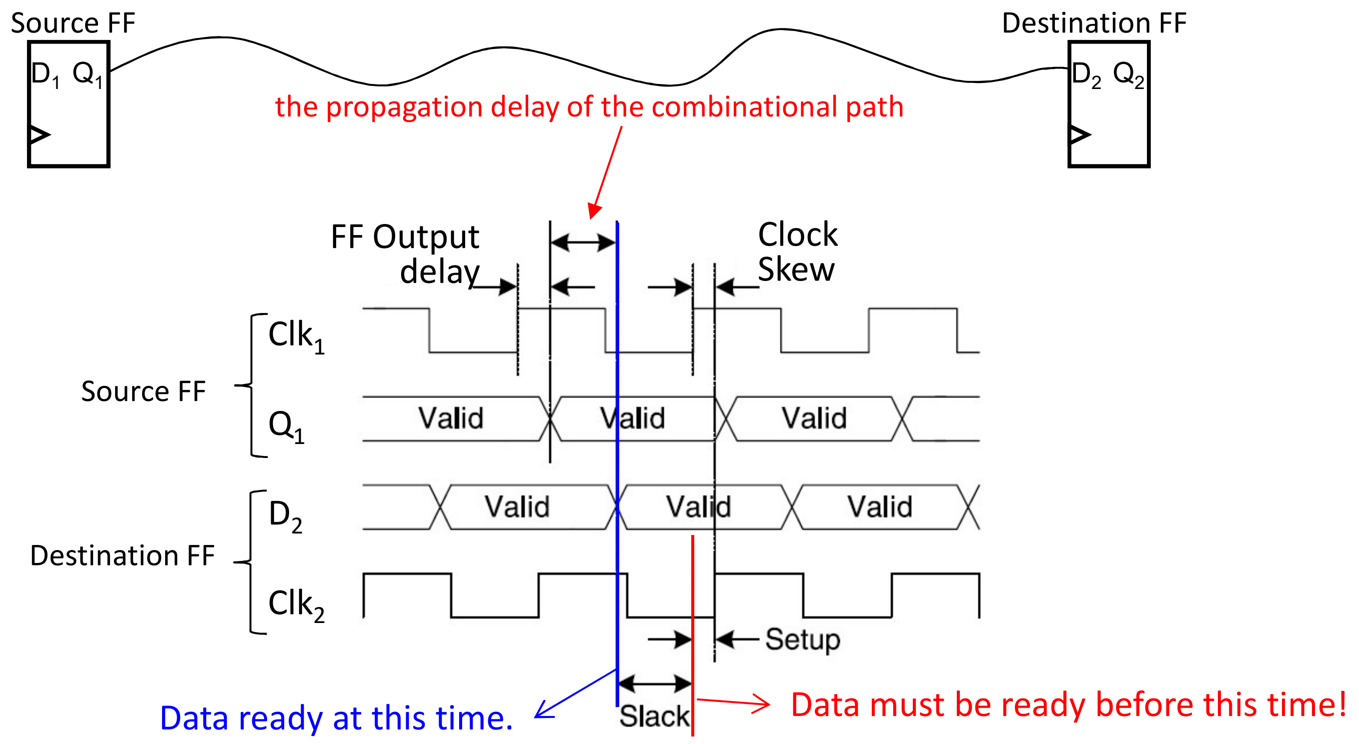
```
aquila_top Aquila_SoC
(
    .clk_i(ui_clk),
    .rst_i(ui_rst),      // level-sensitive reset signal.
    .base_addr_i(32'b0),  // initial program counter.

    .M_IMEM_strobe_o(IMEM_strobe),
    .M_IMEM_addr_o(IMEM_addr),
    .M_IMEM_done_i(IMEM_done),
    .M_IMEM_data_i(IMEM_data),
```

Name	Constraints	Status	Incremental	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT
synth_1 (active)	constrs_1	Synthesis Out-of-date	Off								10906
impl_1	constrs_1	Implementation Out-of-date	Off	-1.095	-556.241	0.020	0.000	0.000	0.983	0	15068
Out-of-Context Module Runs											
mig_7series_0_synth_1	mig_7series_0	synth_design Complete!	Off								
clk_wiz_0_synth_1	clk_wiz_0	synth_design Complete!	Off								1

We got negative slack time in our design!

Meeting Timing Constraint on FPGA

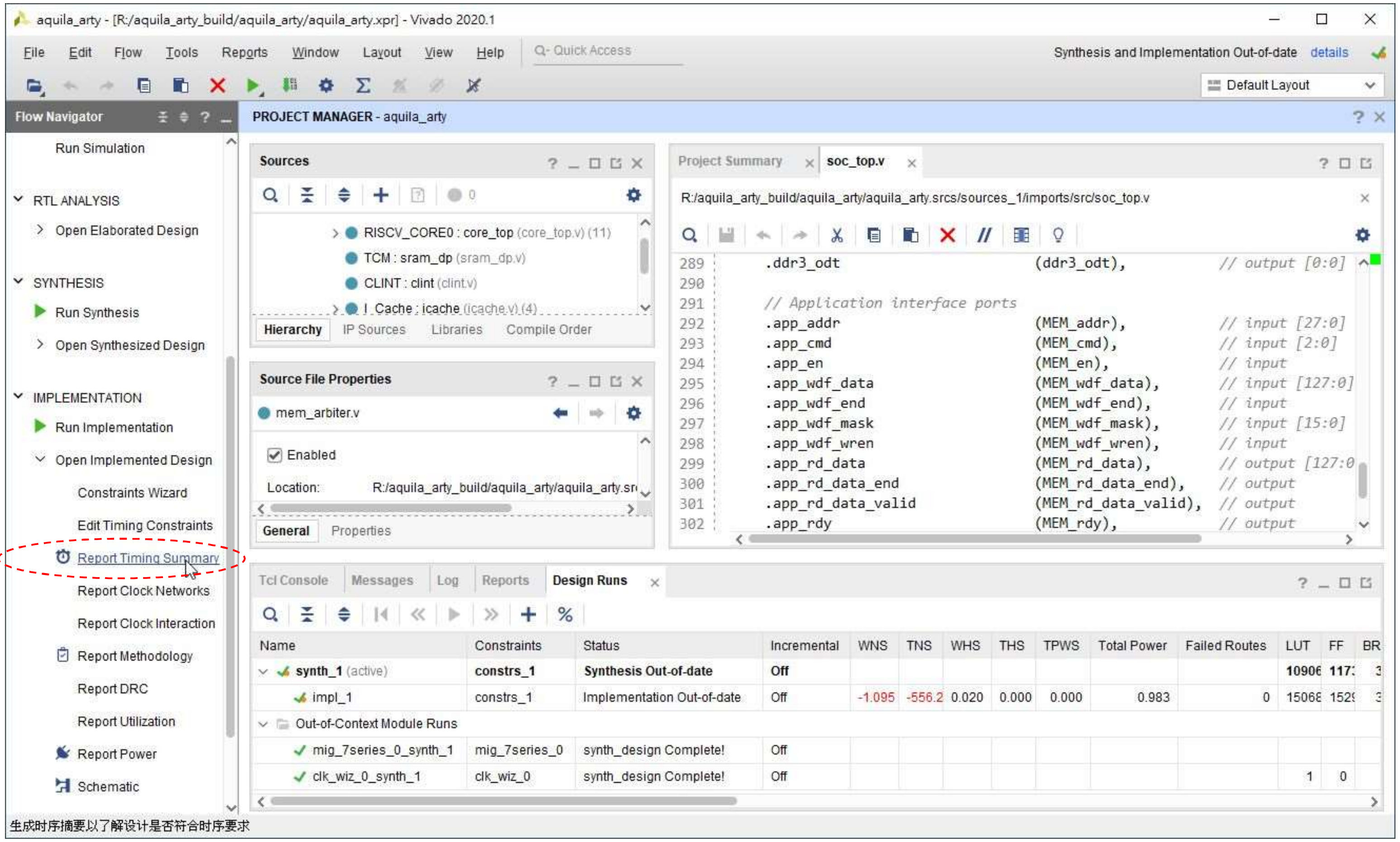
- The critical path must has positive slack time to meet the timing constraint



* slack_time = target_clock_period - (FF_output_delay + prop._delay + setup_time – clock_skew)

Locating Timing Violation

- To locate the critical path that cause timing violation, check the implementation timing report of Vivado

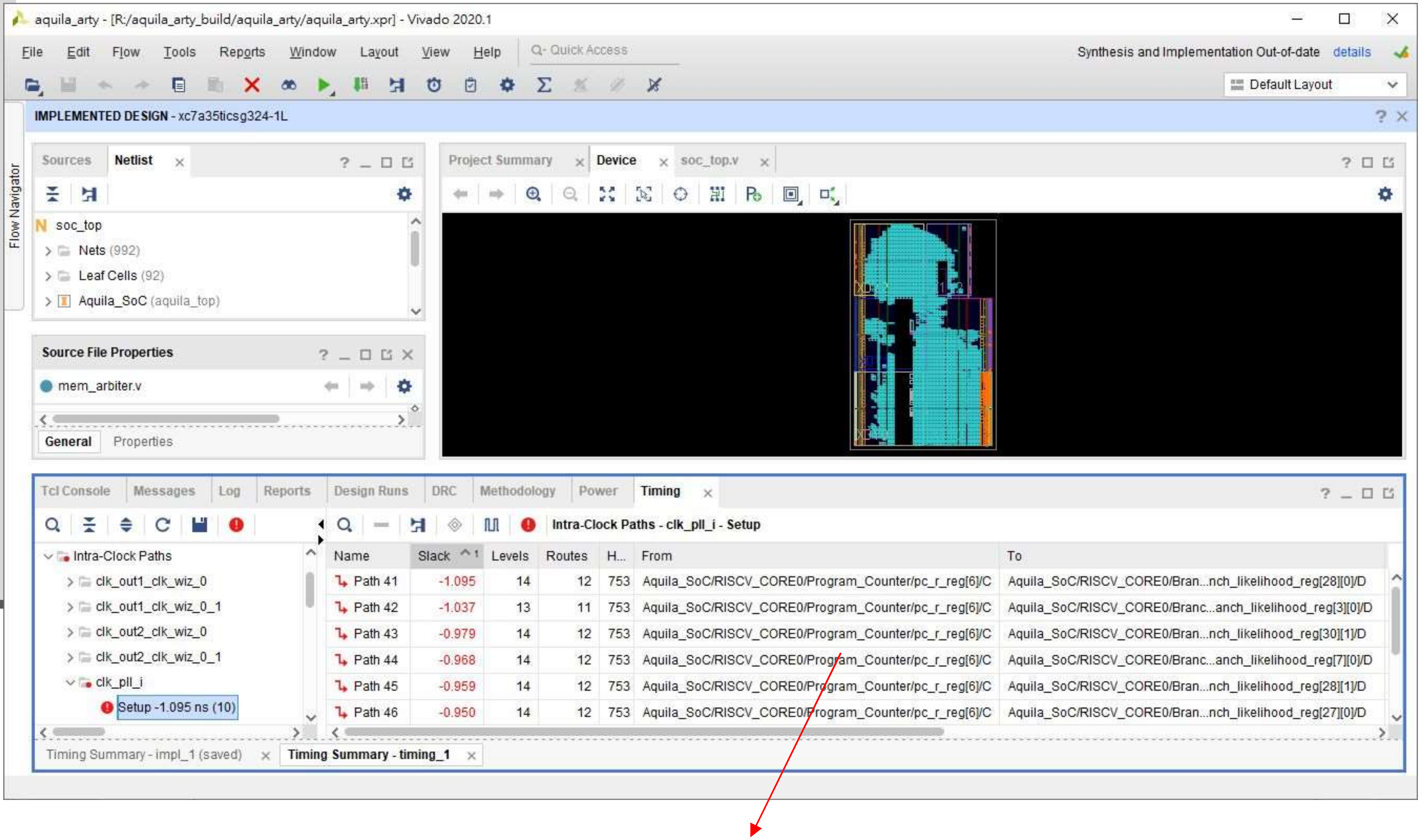

aquila_arty - [R:/aquila_arty_build/aquila_arty/aquila_arty.xpr] - Vivado 2020.1
File Edit Flow Tools Reports Window Layout View Help
Quick Access
Synthesis and Implementation Out-of-date details
Default Layout
Flow Navigator
PROJECT MANAGER - aquila_arty
Run Simulation
RTL ANALYSIS
Open Elaborated Design
SYNTHESIS
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
Run Implementation
Open Implemented Design
Constraints Wizard
Edit Timing Constraints
Report Timing Summary
Report Clock Networks
Report Clock Interaction
Report Methodology
Report DRC
Report Utilization
Report Power
Schematic
Sources
RISCV_CORE0 : core_top (core_top.v) (11)
TCM : sram_dp (sram_dp.v)
CLINT : clint (clint.v)
I_Cache : icache (icache.v) (4)
Hierarchy IP Sources Libraries Compile Order
Source File Properties
mem_arbiter.v
Enabled
Location: R:/aquila_arty_build/aquila_arty/aquila_arty.sr
General Properties
Project Summary
soc_top.v
R:/aquila_arty_build/aquila_arty/aquila_arty.srcs/sources_1/imports/src/soc_top.v
Tcl Console Messages Log Reports Design Runs

```
    .ddr3_odt                     (ddr3_odt),          // output [0:0]

    // Application interface ports
    .app_addr                     (MEM_addr),          // input [27:0]
    .app_cmd                      (MEM_cmd),           // input [2:0]
    .app_en                       (MEM_en),            // input
    .app_wdf_data                 (MEM_wdf_data),      // input [127:0]
    .app_wdf_end                  (MEM_wdf_end),       // input
    .app_wdf_mask                 (MEM_wdf_mask),      // input [15:0]
    .app_wdf_wren                 (MEM_wdf_wren),      // input
    .app_rd_data                  (MEM_rd_data),       // output [127:0
    .app_rd_data_end              (MEM_rd_data_end),   // output
    .app_rd_data_valid            (MEM_rd_data_valid), // output
    .app_rdy                      (MEM_rdy),           // output
```

Name	Constraints	Status	Incremental	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BR
synth_1 (active)	constrs_1	Synthesis Out-of-date	Off								10906	117	3
impl_1	constrs_1	Implementation Out-of-date	Off	-1.095	-556.2	0.020	0.000	0.000	0.983	0	15068	152	3
Out-of-Context Module Runs													
mig_7series_0_synth_1	mig_7series_0	synth_design Complete!	Off										
clk_wiz_0_synth_1	clk_wiz_0	synth_design Complete!	Off								1	0	

生成时序摘要以了解设计是否符合时序要求

Fixing Critical Paths



aquila_arty - [R:/aquila_arty_build/aquila_arty/aquila_arty.xpr] - Vivado 2020.1

Synthesis and Implementation Out-of-date details

IMPLEMENTED DESIGN - xc7a35ticsg324-1L

- soc_top
  - Nets (992)
  - Leaf Cells (92)
  - Aquila_SoC (aquila_top)

Source File Properties: mem_arbiter.v

Intra-Clock Paths - clk_pll_i - Setup

- Intra-Clock Paths
  - clk_out1_clk_wiz_0
  - clk_out1_clk_wiz_0_1
  - clk_out2_clk_wiz_0
  - clk_out2_clk_wiz_0_1
  - clk_pll_i
    - Setup -1.095 ns (10)

Name	Slack	Levels	Routes	H...	From	To
Path 41	-1.095	14	12	753	Aquila_SoC/RISCV_CORE0/Program_Counter/pc_r_reg[6]/C	Aquila_SoC/RISCV_CORE0/Bran...nch_likelihood_reg[28][0]/D
Path 42	-1.037	13	11	753	Aquila_SoC/RISCV_CORE0/Program_Counter/pc_r_reg[6]/C	Aquila_SoC/RISCV_CORE0/Branc...anch_likelihood_reg[3][0]/D
Path 43	-0.979	14	12	753	Aquila_SoC/RISCV_CORE0/Program_Counter/pc_r_reg[6]/C	Aquila_SoC/RISCV_CORE0/Bran...nch_likelihood_reg[30][1]/D
Path 44	-0.968	14	12	753	Aquila_SoC/RISCV_CORE0/Program_Counter/pc_r_reg[6]/C	Aquila_SoC/RISCV_CORE0/Branc...anch_likelihood_reg[7][0]/D
Path 45	-0.959	14	12	753	Aquila_SoC/RISCV_CORE0/Program_Counter/pc_r_reg[6]/C	Aquila_SoC/RISCV_CORE0/Bran...nch_likelihood_reg[28][1]/D
Path 46	-0.950	14	12	753	Aquila_SoC/RISCV_CORE0/Program_Counter/pc_r_reg[6]/C	Aquila_SoC/RISCV_CORE0/Bran...nch_likelihood_reg[27][0]/D

We have a critical path from the `pc_r` of the PCU to the `likelihood_reg` of the BPU!

Warning on Using ILA

- For this homework, you should use ILA for debugging because it is difficult to simulate DDR memory at system level
- Unfortunately, ILA uses on-chip memory to capture data. If you need to capture a lot of data, you may have to reduce the cache sizes of Aquila to save more BRAMs for ILA

Your Homework

- Fix the timing violation of Aquila with memory controller
- There are many different ways to cut a critical path into two shorter paths. You should try to find a minimal clean solution (i.e., don’t change the code significantly)

- Write a report:
  - Describe the critical paths that cause the timing violation
  - Describe how you modify the critical paths to meet the timing constraint