Configuring and Using the DDR2 Memory on the Nexys 4 DDR/Nexys A7 FPGA Trainer Board using Vivado

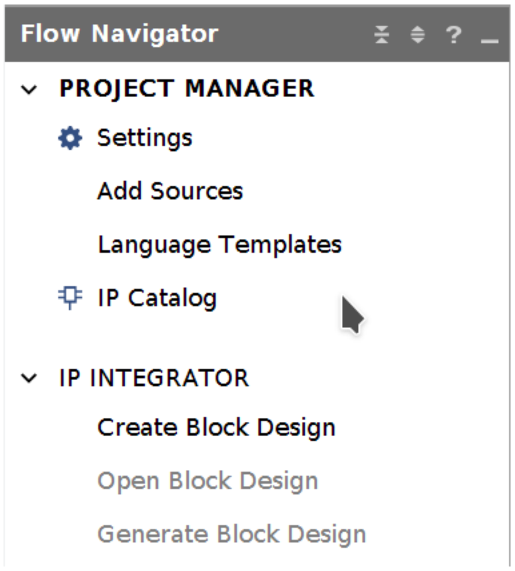
Unlike the block RAM resources internal to the FPGA which can be easily instantiated into a design or implicitly added by the synthesizer, DDR memory generally requires the use of IP to properly manage the refresh, precharge, calibration parameters, etc. required for successful memory transactions.

Xilinx provides such IP for their FPGAs, and this guide shows how to properly configure their Memory Interface Generator IP for the specific DDR2 memory IC on the Nexys board. Additionally, an example module is provided demonstrating the use of Xilinx's MIG IP.

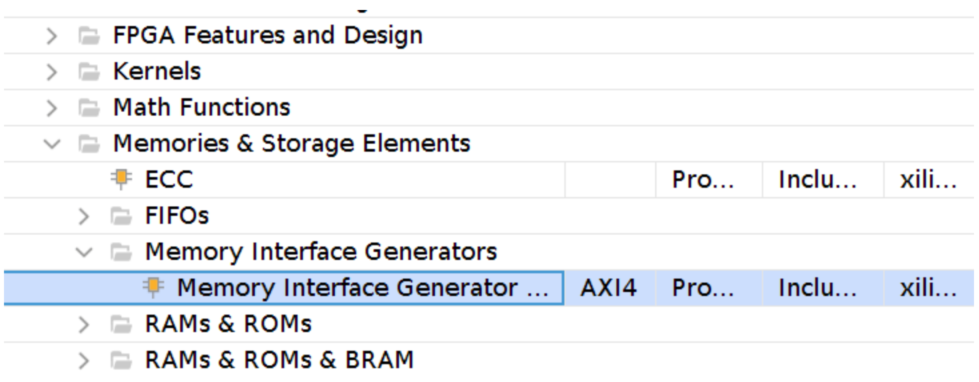
This guide was written against Vivado 2018.3.1 containing MIG IP v4.2. Future versions should be compatible although Xilinx has been known to change the behavior of signals without changing their names/interfaces. In fact, if you need to consult the documentation on the MIG, **be careful that the document matches the exact version of the MIG IP**. This has caused unnecessary confusion and subtle bugs in the past.

Generating the Memory Controller IP

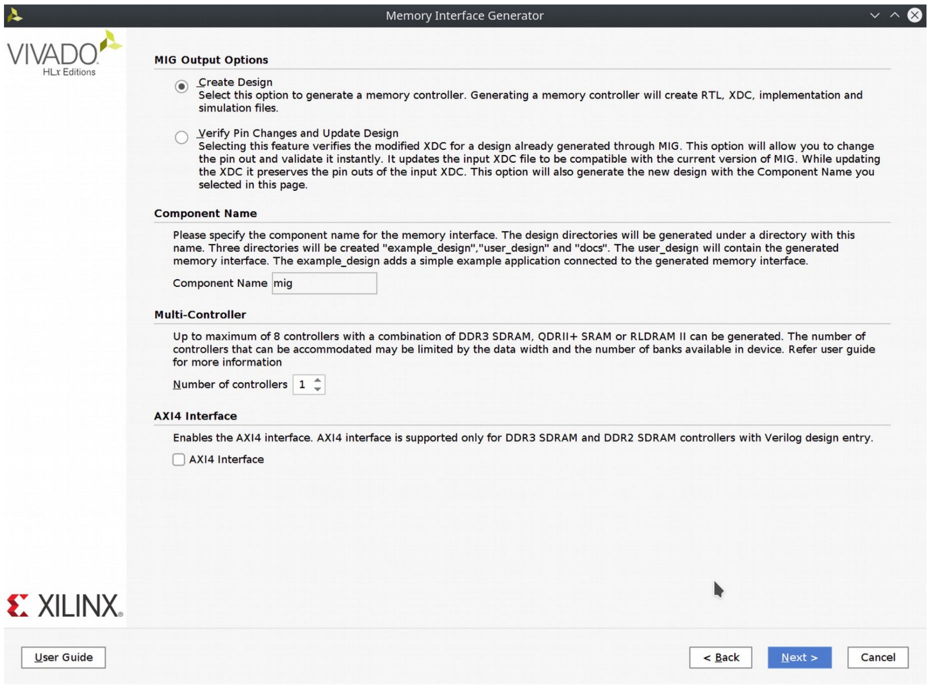
The example project already has a MIG configured for the Nexys board, but if you wish to re-generate it or create one in a difference project, here's a walkthrough. Begin by opening the IP catalog under the Project Manager:



Open the Memory Interface Generator:



Select "Create Design" and give the module a memorable name (this is the name you will use to instantiate the module in your HDL). Additionally, "AXI4 Interface" should be disabled so that we can work directly with the memory controller's "user interface":



MIG Output Options

[✓ Create Design] Select this option to generate a memory controller. Generating a memory controller will create RTL, XDC, implementation and simulation files.

Verify Pin Changes and Update Design Selecting this feature verifies the modified XDC for a design already generated through MIG. This option will allow you to change the pin out and validate it instantly. It updates the input XDC file to be compatible with the current version of MIG. While updating the XDC it preserves the pin outs of the input XDC. This option will also generate the new design with the Component Name you selected in this page.

Component Name

Please specify the component name for the memory interface. The design directories will be generated under a directory with this name. Three directories will be created "example_design","user_design" and "docs". The user_design will contain the generated memory interface. The example_design adds a simple example application connected to the generated memory interface.

Component Name: mig

Multi-Controller

Up to maximum of 8 controllers with a combination of DDR3 SDRAM, QDRII+ SRAM or RLDRAM II can be generated. The number of controllers that can be accommodated may be limited by the data width and the number of banks available in device. Refer user guide for more information

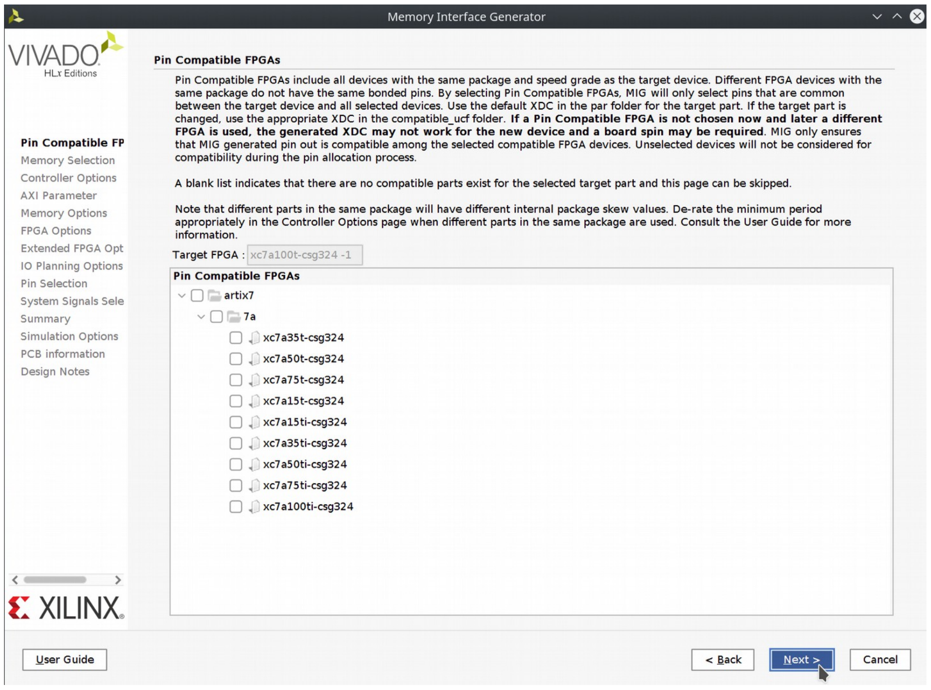
Number of controllers: 1

AXI4 Interface

Enables the AXI4 interface. AXI4 interface is supported only for DDR3 SDRAM and DDR2 SDRAM controllers with Verilog design entry.

☐ AXI4 Interface

The target FPGA should already be selected from the enclosing project (Which is the xc7a100t-csg324-1). No additional pin compatible FPGAs should be selected here. Generally these settings are useful if it's possible your project may need to move to a different FPGA down the line, but as the hardware is fixed, it is unnecessary:



Pin Compatible FPGAs

Pin Compatible FPGAs include all devices with the same package and speed grade as the target device. Different FPGA devices with the same package do not have the same bonded pins. By selecting Pin Compatible FPGAs, MIG will only select pins that are common between the target device and all selected devices. Use the default XDC in the par folder for the target part. If the target part is changed, use the appropriate XDC in the compatible_ucf folder. **If a Pin Compatible FPGA is not chosen now and later a different FPGA is used, the generated XDC may not work for the new device and a board spin may be required**. MIG only ensures that MIG generated pin out is compatible among the selected compatible FPGA devices. Unselected devices will not be considered for compatibility during the pin allocation process.

A blank list indicates that there are no compatible parts exist for the selected target part and this page can be skipped.

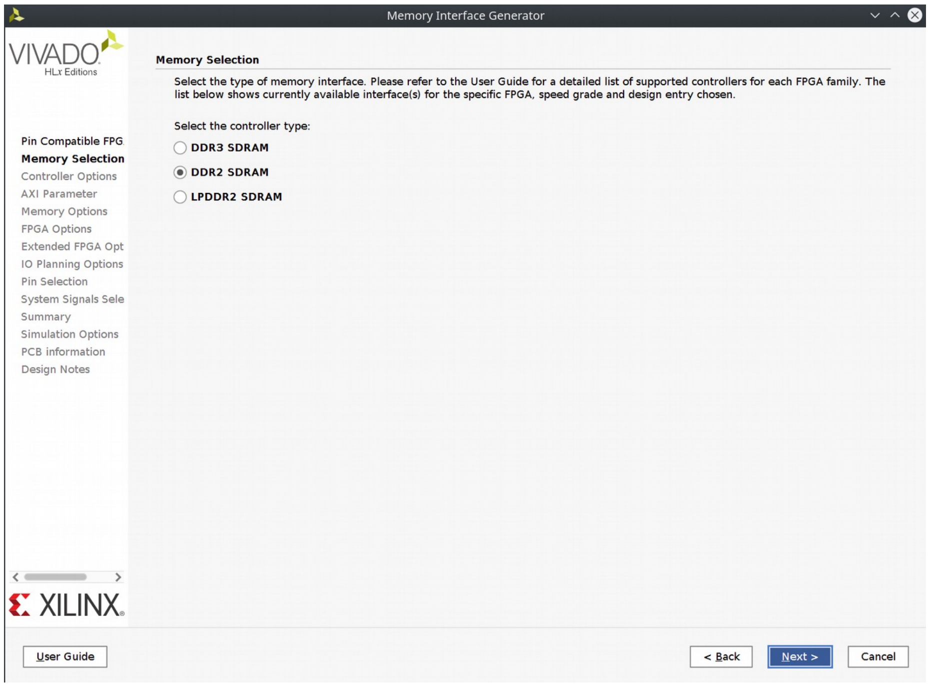
Note that different parts in the same package will have different internal package skew values. De-rate the minimum period appropriately in the Controller Options page when different parts in the same package are used. Consult the User Guide for more information.

Target FPGA : xc7a100t-csg324 -1

Pin Compatible FPGAs

- ☐ artix7
  - ☐ 7a
    - ☐ xc7a35t-csg324
    - ☐ xc7a50t-csg324
    - ☐ xc7a75t-csg324
    - ☐ xc7a15t-csg324
    - ☐ xc7a15ti-csg324
    - ☐ xc7a35ti-csg324
    - ☐ xc7a50ti-csg324
    - ☐ xc7a75ti-csg324
    - ☐ xc7a100ti-csg324

Select DDR2 SDRAM:



Memory Selection

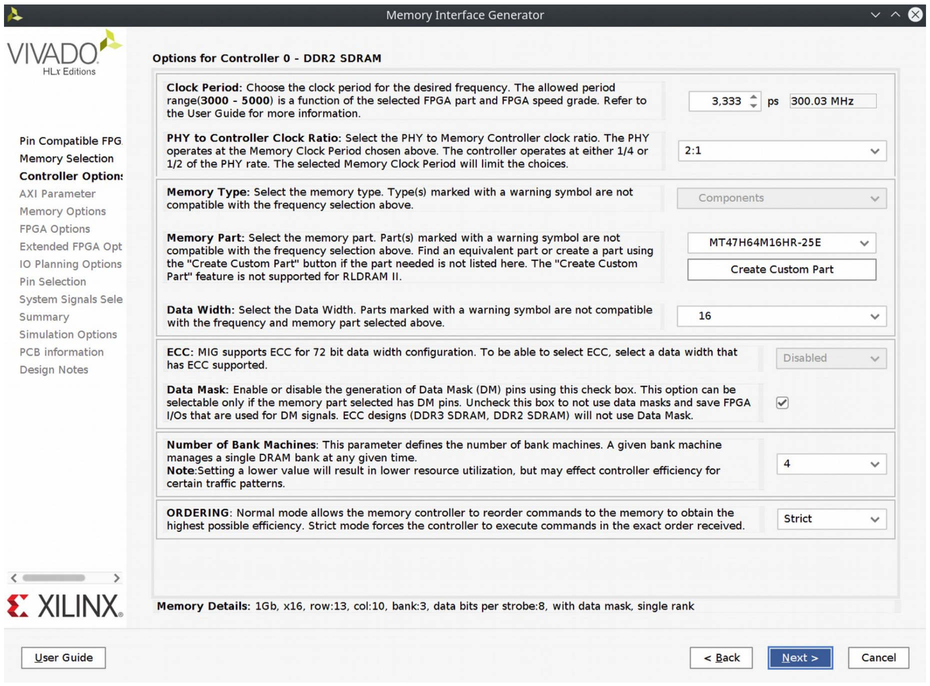
Select the type of memory interface. Please refer to the User Guide for a detailed list of supported controllers for each FPGA family. The list below shows currently available interface(s) for the specific FPGA, speed grade and design entry chosen.

Select the controller type:

- DDR3 SDRAM
- [✓ DDR2 SDRAM]
- LPDDR2 SDRAM

Select a clock period of 3333 ps (300MHz), with a PHY to Controller clock ratio of 1:2, a memory part of MT47H64M16HR-25E (the chip on our board), a data width of 16-bits, data mask selected (allowing us to perform less than full-width transfers), and strict ordering.

While most projects will probably tolerate Normal ordering where the IP is allowed to re-order already queued transactions for speed reasons, Strict ordering may be required for certain access patterns and is the safer option without knowing the application:



Options for Controller 0 - DDR2 SDRAM

Clock Period: Choose the clock period for the desired frequency. The allowed period range(**3000 - 5000**) is a function of the selected FPGA part and FPGA speed grade. Refer to the User Guide for more information. 3,333 ps 300.03 MHz

PHY to Controller Clock Ratio: Select the PHY to Memory Controller clock ratio. The PHY operates at the Memory Clock Period chosen above. The controller operates at either 1/4 or 1/2 of the PHY rate. The selected Memory Clock Period will limit the choices. 2:1

Memory Type: Select the memory type. Type(s) marked with a warning symbol are not compatible with the frequency selection above. Components

Memory Part: Select the memory part. Part(s) marked with a warning symbol are not compatible with the frequency selection above. Find an equivalent part or create a part using the "Create Custom Part" button if the part needed is not listed here. The "Create Custom Part" feature is not supported for RLDRAM II. MT47H64M16HR-25E

Create Custom Part

Data Width: Select the Data Width. Parts marked with a warning symbol are not compatible with the frequency and memory part selected above. 16

ECC: MIG supports ECC for 72 bit data width configuration. To be able to select ECC, select a data width that has ECC supported. Disabled

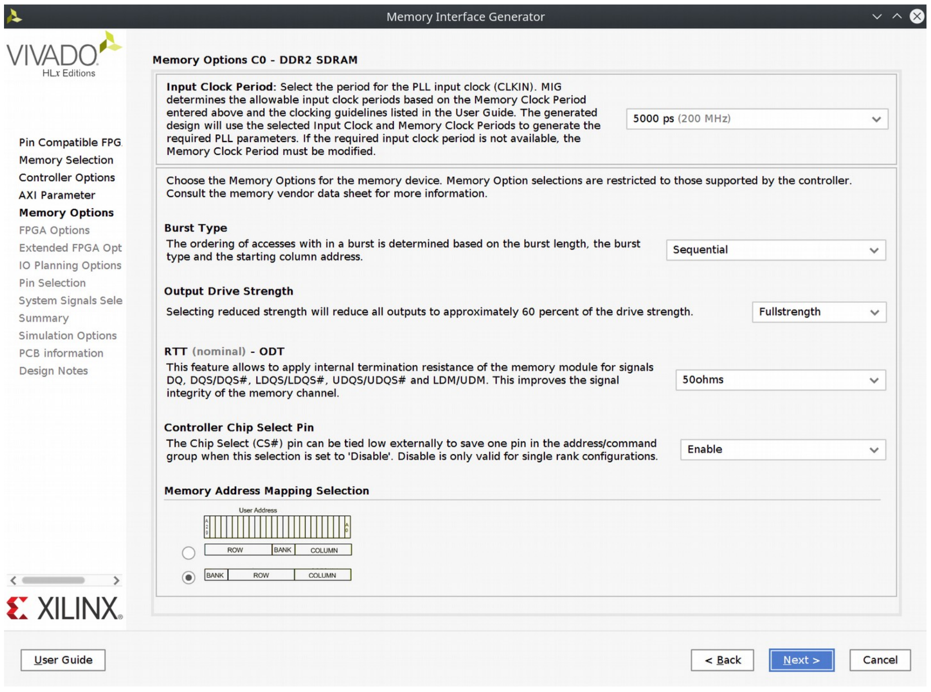
Data Mask: Enable or disable the generation of Data Mask (DM) pins using this check box. This option can be selectable only if the memory part selected has DM pins. Uncheck this box to not use data masks and save FPGA I/Os that are used for DM signals. ECC designs (DDR3 SDRAM, DDR2 SDRAM) will not use Data Mask. ✓

Number of Bank Machines: This parameter defines the number of bank machines. A given bank machine manages a single DRAM bank at any given time.
Note: Setting a lower value will result in lower resource utilization, but may effect controller efficiency for certain traffic patterns. 4

ORDERING: Normal mode allows the memory controller to reorder commands to the memory to obtain the highest possible efficiency. Strict mode forces the controller to execute commands in the exact order received. Strict

Memory Details: 1Gb, x16, row:13, col:10, bank:3, data bits per strobe:8, with data mask, single rank

The MIG will internally instantiate one of the DCM/PLL blocks to generate the memory clock. Here, we need to tell the IP generator what the *input* clock is into that PLL. Other parts of MIG operate directly from a reference clock, which needs to be either externally provided or tied to the input clock at a fixed 200MHz. We will generate a 200MHz input clock elsewhere and later attach it to the reference clock input, so select 5000 ps here. The RTT should be set to 50Ohm, and the other options are configured as seen:



Memory Options C0 - DDR2 SDRAM

Input Clock Period: Select the period for the PLL input clock (CLKIN). MIG determines the allowable input clock periods based on the Memory Clock Period entered above and the clocking guidelines listed in the User Guide. The generated design will use the selected Input Clock and Memory Clock Periods to generate the required PLL parameters. If the required input clock period is not available, the Memory Clock Period must be modified. — 5000 ps (200 MHz)

Choose the Memory Options for the memory device. Memory Option selections are restricted to those supported by the controller. Consult the memory vendor data sheet for more information.

Burst Type
The ordering of accesses with in a burst is determined based on the burst length, the burst type and the starting column address. — Sequential

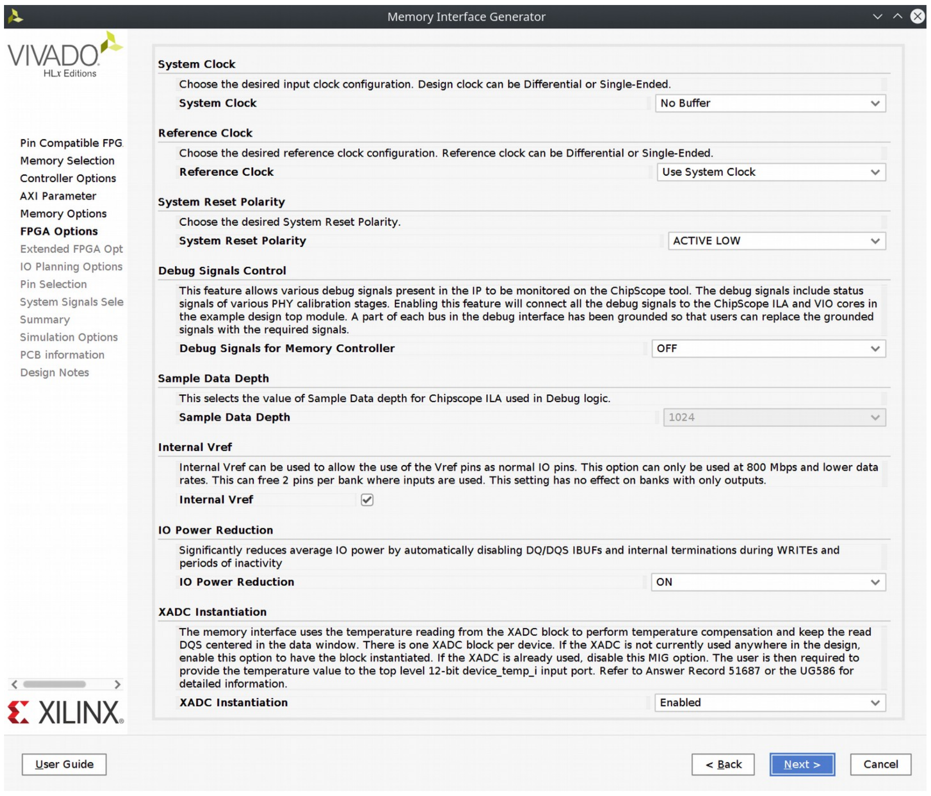
Output Drive Strength
Selecting reduced strength will reduce all outputs to approximately 60 percent of the drive strength. — Fullstrength

RTT (nominal) - ODT
This feature allows to apply internal termination resistance of the memory module for signals DQ, DQS/DQS#, LDQS/LDQS#, UDQS/UDQS# and LDM/UDM. This improves the signal integrity of the memory channel. — 50ohms

Controller Chip Select Pin
The Chip Select (CS#) pin can be tied low externally to save one pin in the address/command group when this selection is set to 'Disable'. Disable is only valid for single rank configurations. — Enable

Memory Address Mapping Selection
○ ROW | BANK | COLUMN
● BANK | ROW | COLUMN

Here we setup the IP for our particular clock and reset style, tying the reference clock to the "system clock", which is the previously mentioned input clock. Be sure to set the other configuration options as well:



System Clock
Choose the desired input clock configuration. Design clock can be Differential or Single-Ended.
System Clock — No Buffer

Reference Clock
Choose the desired reference clock configuration. Reference clock can be Differential or Single-Ended.
Reference Clock — Use System Clock

System Reset Polarity
Choose the desired System Reset Polarity.
System Reset Polarity — ACTIVE LOW

Debug Signals Control
This feature allows various debug signals present in the IP to be monitored on the ChipScope tool. The debug signals include status signals of various PHY calibration stages. Enabling this feature will connect all the debug signals to the ChipScope ILA and VIO cores in the example design top module. A part of each bus in the debug interface has been grounded so that users can replace the grounded signals with the required signals.
Debug Signals for Memory Controller — OFF

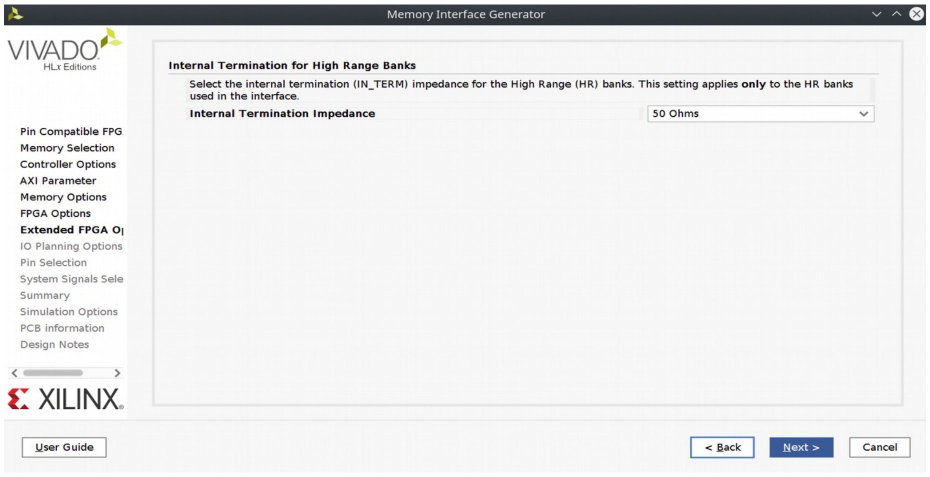
Sample Data Depth
This selects the value of Sample Data depth for Chipscope ILA used in Debug logic.
Sample Data Depth — 1024

Internal Vref
Internal Vref can be used to allow the use of the Vref pins as normal IO pins. This option can only be used at 800 Mbps and lower data rates. This can free 2 pins per bank where inputs are used. This setting has no effect on banks with only outputs.
Internal Vref — ✓

IO Power Reduction
Significantly reduces average IO power by automatically disabling DQ/DQS IBUFs and internal terminations during WRITEs and periods of inactivity
IO Power Reduction — ON

XADC Instantiation
The memory interface uses the temperature reading from the XADC block to perform temperature compensation and keep the read DQS centered in the data window. There is one XADC block per device. If the XADC is not currently used anywhere in the design, enable this option to have the block instantiated. If the XADC is already used, disable this MIG option. The user is then required to provide the temperature value to the top level 12-bit device_temp_i input port. Refer to Answer Record 51687 or the UG586 for detailed information.
XADC Instantiation — Enabled

Termination should be set to 50Ohms:

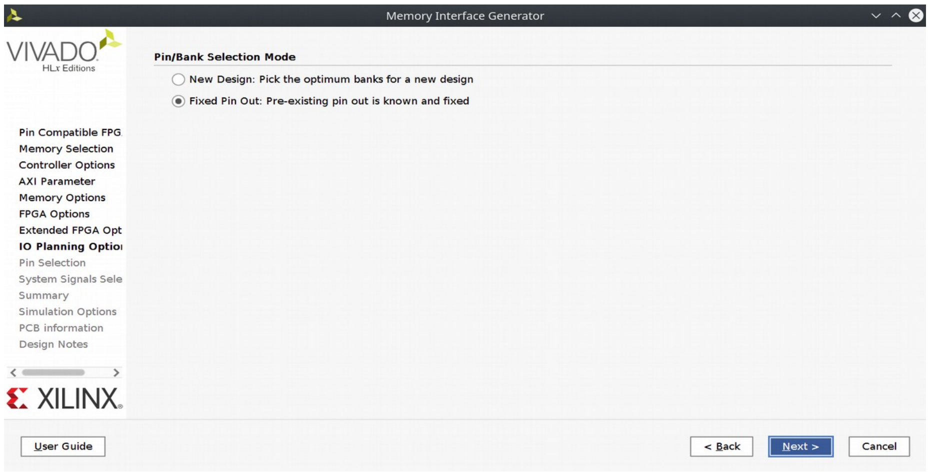


Internal Termination for High Range Banks

Select the internal termination (IN_TERM) impedance for the High Range (HR) banks. This setting applies **only** to the HR banks used in the interface.

Internal Termination Impedance 50 Ohms

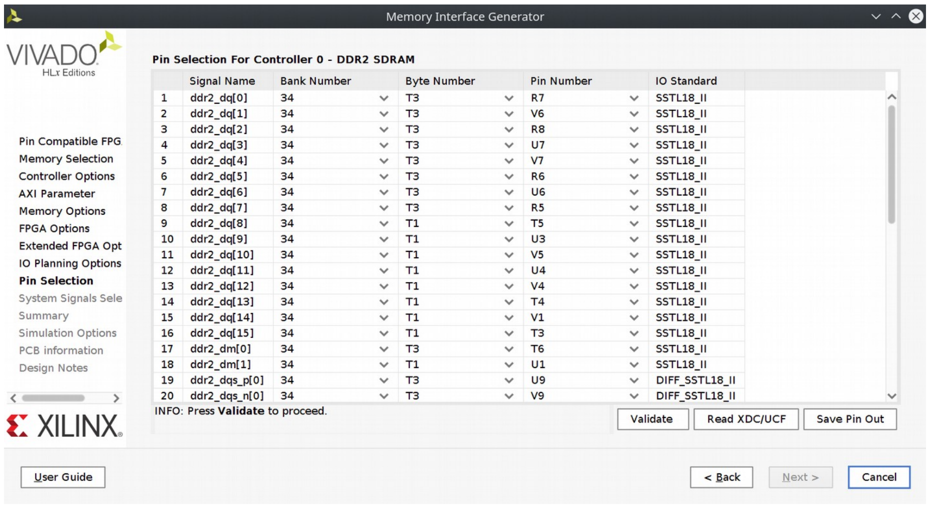
Given that our hardware is already designed and assembled, we need to tell the IP generator exactly where all the various DDR2 pins are located using the Fixed Pin Out option:



Pin/Bank Selection Mode

New Design: Pick the optimum banks for a new design [✓ Fixed Pin Out: Pre-existing pin out is known and fixed]

Now load the provided "mig.ucf" file using the Read "XDC/UCF" button, and press "Validate". The IP tool should respond that the pinout is valid. Since we've loaded the desired pins for the DDR memory controller, we no longer need to specify anything about those signals in our overall project constraints file. The MIG IP itself will generate supplemental constraints containing the location, name, jitter, etc. about those nets automatically:

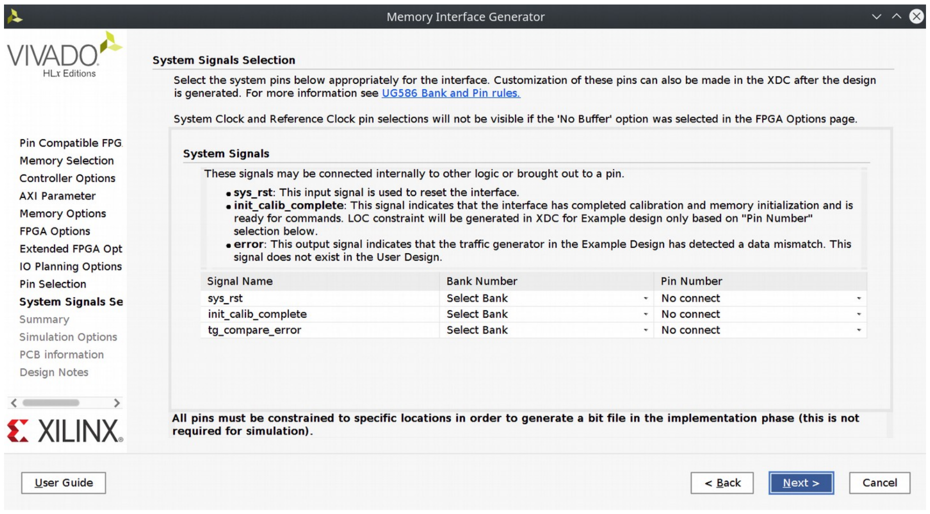


Pin Selection For Controller 0 - DDR2 SDRAM

	Signal Name	Bank Number	Byte Number	Pin Number	IO Standard
1	ddr2_dq[0]	34	T3	R7	SSTL18_II
2	ddr2_dq[1]	34	T3	V6	SSTL18_II
3	ddr2_dq[2]	34	T3	R8	SSTL18_II
4	ddr2_dq[3]	34	T3	U7	SSTL18_II
5	ddr2_dq[4]	34	T3	V7	SSTL18_II
6	ddr2_dq[5]	34	T3	R6	SSTL18_II
7	ddr2_dq[6]	34	T3	U6	SSTL18_II
8	ddr2_dq[7]	34	T3	R5	SSTL18_II
9	ddr2_dq[8]	34	T1	T5	SSTL18_II
10	ddr2_dq[9]	34	T1	U3	SSTL18_II
11	ddr2_dq[10]	34	T1	V5	SSTL18_II
12	ddr2_dq[11]	34	T1	U4	SSTL18_II
13	ddr2_dq[12]	34	T1	V4	SSTL18_II
14	ddr2_dq[13]	34	T1	T4	SSTL18_II
15	ddr2_dq[14]	34	T1	V1	SSTL18_II
16	ddr2_dq[15]	34	T1	T3	SSTL18_II
17	ddr2_dm[0]	34	T3	T6	SSTL18_II
18	ddr2_dm[1]	34	T1	U1	SSTL18_II
19	ddr2_dqs_p[0]	34	T3	U9	DIFF_SSTL18_II
20	ddr2_dqs_n[0]	34	T3	V9	DIFF_SSTL18_II

INFO: Press **Validate** to proceed.

The IP tool will offer options to route status lines to physical pins if desired, but we can set them all to "No connect". They are internally available regardless:



System Signals Selection

Select the system pins below appropriately for the interface. Customization of these pins can also be made in the XDC after the design is generated. For more information see UG586 Bank and Pin rules.

System Clock and Reference Clock pin selections will not be visible if the 'No Buffer' option was selected in the FPGA Options page.

System Signals

These signals may be connected internally to other logic or brought out to a pin.

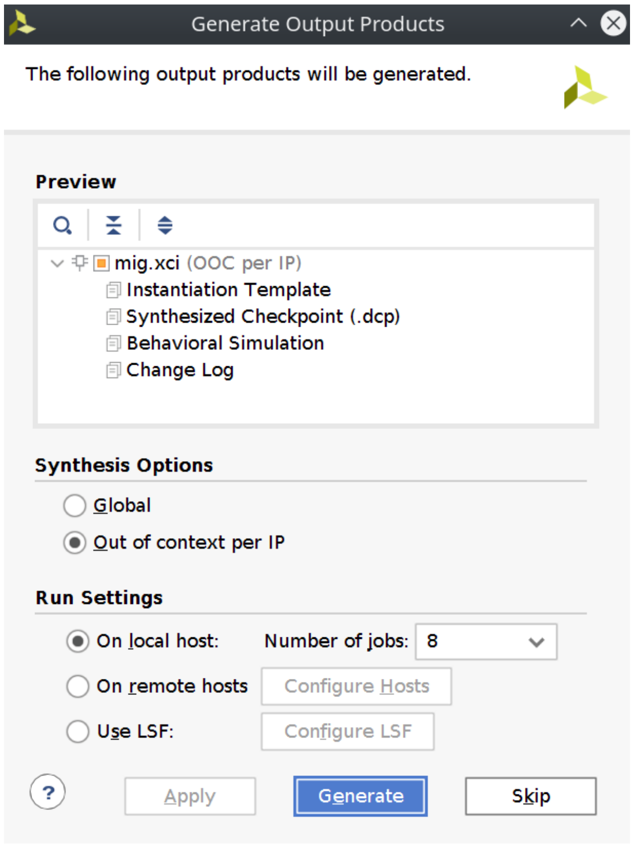
- **sys_rst**: This input signal is used to reset the interface.
- **init_calib_complete**: This signal indicates that the interface has completed calibration and memory initialization and is ready for commands. LOC constraint will be generated in XDC for Example design only based on "Pin Number" selection below.
- **error**: This output signal indicates that the traffic generator in the Example Design has detected a data mismatch. This signal does not exist in the User Design.

Signal Name	Bank Number	Pin Number
sys_rst	Select Bank	No connect
init_calib_complete	Select Bank	No connect
tg_compare_error	Select Bank	No connect

All pins must be constrained to specific locations in order to generate a bit file in the implementation phase (this is not required for simulation).

Finally, a summary is available. It is not necessary to agree to the simulation agreement as the built-in simulation model of our specific DDR2 IC is out-of-date, (a more up-to-date model from the manufacturer is provided as part of this project).

Finishing out the IP tool, Vivado should prompt you to generate output products for the IP block with your configuration, which should be done now:



The following output products will be generated.

Preview

- mig.xci (OOC per IP)
  - Instantiation Template
  - Synthesized Checkpoint (.dcp)
  - Behavioral Simulation
  - Change Log

Synthesis Options

Global [✓ Out of context per IP]

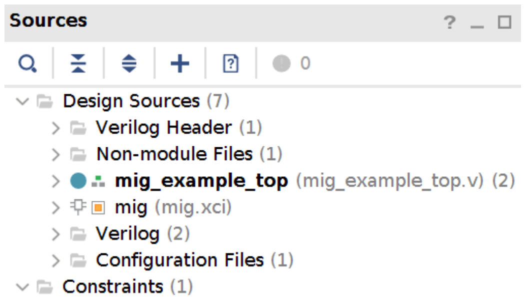
Run Settings

[✓ On local host:] Number of jobs: 8

On remote hosts — Configure Hosts

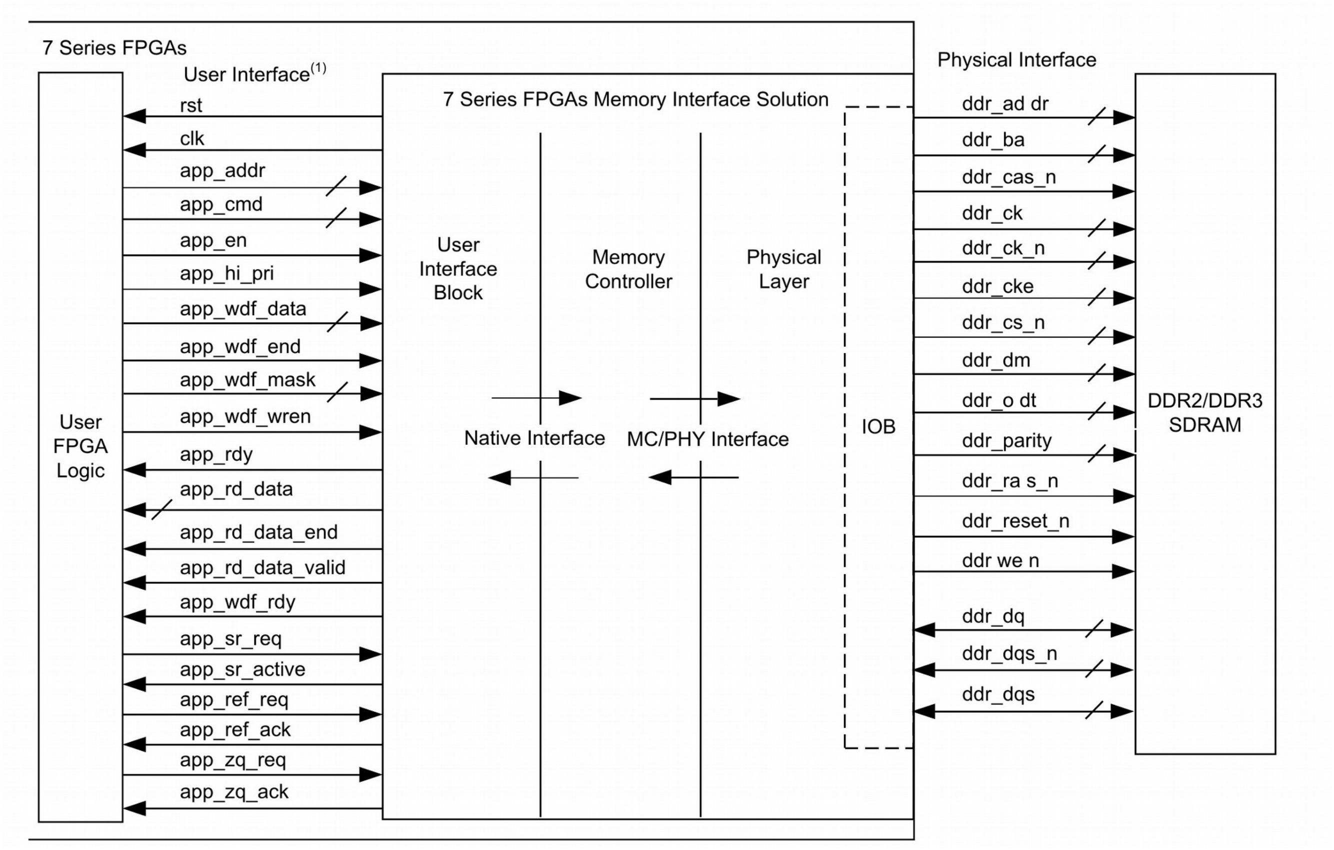
Use LSF: — Configure LSF

After some time the block should appear in the hierarchy with the name you initially gave it:


Sources

- Design Sources (7)
  - Verilog Header (1)
  - Non-module Files (1)
  - **mig_example_top** (mig_example_top.v) (2)
  - mig (mig.xci)
  - Verilog (2)
  - Configuration Files (1)
- Constraints (1)

About the Controller



As noted on this excerpt from the official documentation, the MIG IP block has two sets of signals, physical interface signals connecting to the DDR IC, and those providing the abstract memory interface, facing the rest of the logic in your design.

The physical interface signals need to be passed through the levels of hierarchy to the top file. While not strictly necessary for the physical implementation (the MIG IP can generate its own IO buffers), it is required for simulation. That way, a sim top file can connect the "physical" interface signals to the simulated memory module and allow for end-to-end testing.

Some Notes on the MIG's Interface

The MIG exposes a command queue, accepting or providing data according to its internal state machine, whose status is exposed through the various valid and ready flags. Read and Write transactions are initiated by the user via the *app_cmd* and *app_en* signals. In general, the MIG signals the receipt of input commands/data, and it is the responsibility of the user application to wait until those commands/data are acknowledged as seen in the following excerpt from the docs:

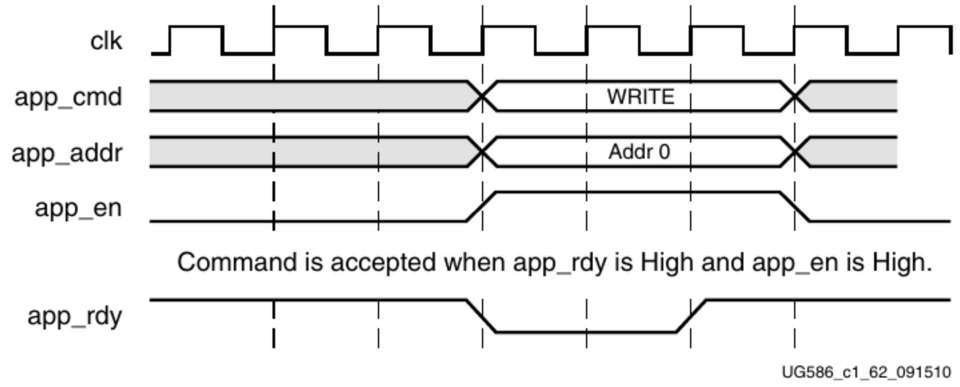

clk
app_cmd
WRITE
app_addr
Addr 0
app_en
Command is accepted when app_rdy is High and app_en is High.
app_rdy
UG586_c1_62_091510

Figure 1-74: **UI Command Timing Diagram with app_rdy Asserted**

This paradigm lends itself to a simple state machine, sending the appropriate read/write command then waiting for readiness before sending the address and optional data words. Another excerpt from the documentation shows an idealized read transaction:

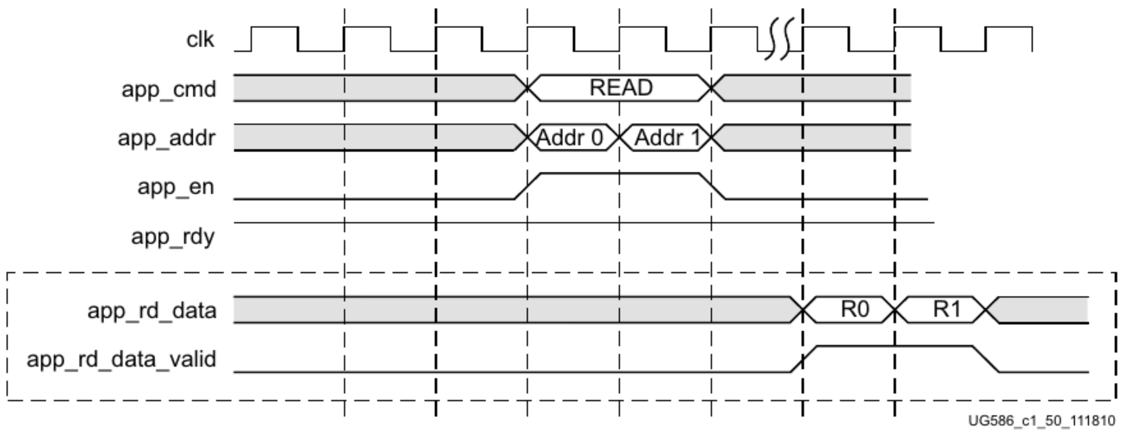


Figure 1-82: **2:1 Mode UI Interface Read Timing Diagram (Memory Burst Type = BL4 or BL8)**

The Xilinx documentation covers this in great detail under the "User Interface" heading of the "Interfacing with the Core" section in Chapter 1.

A More Complete Example

While there is an interface example project accessible by right clicking on the MIG IP module in the hierarchy and selecting "Open IP Example Design", it is somewhat underwhelming and doesn't particularly explain how to use the MIG nor does it handle common use cases. The module's interface is provided in the Verilog and VHDL stubs located at `<project>.srcs/sources_1/ip/<mig_name>/mig_stub.(v|vhdl)`

To accelerate future projects hoping to use the memory interface, and to provide a concrete example, this project (mig_example) was created containing a strobe-oriented SRAM-like interface to the MIG IP in mem_example.v.

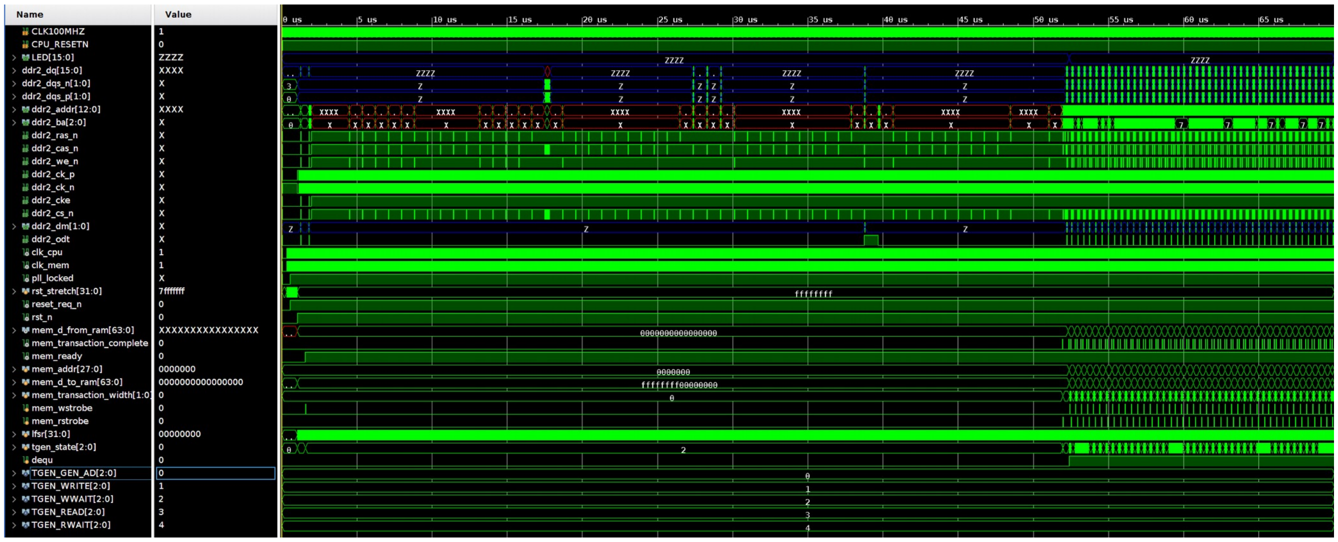
Although the underlying MIG IP supports queuing several reads and writes in a burst, the synchronous access style of mem_example is intended to perform a single read or write as an atomic operation. Additionally, as mem_example was adapted from part of a larger project, it can synchronize signals from the "cpu" clock domain into the memory controller's "mem" clock domain.

This synchronization occurs only on the operation/status strobes and **requires that *addr, data_in* and *width* input signals remain stable from assertion of *rstrobe/wstrobe* until *transaction_complete* is returned**.

As an additional effect, the **"mem" clock must run at least 3 times faster than the "cpu" clock**. While using asynchronous FIFOs would eliminate both of these limitations, in keeping with the general theme, flag synchronizers are used for their simplicity. Again, these are limitations in the mem_example.v module and not the underlying Xilinx MIG IP.

The synthesizer generally does a good job detecting the false timing paths that exist between things like *data_in* changing in the cpu clock domain affecting things in the mem clock domain. That being said, it was still necessary to examine and cut a handful of false paths which can never occur given how we guard against changing those signals using flag synchronizers.

Let's look at our overall simulation:

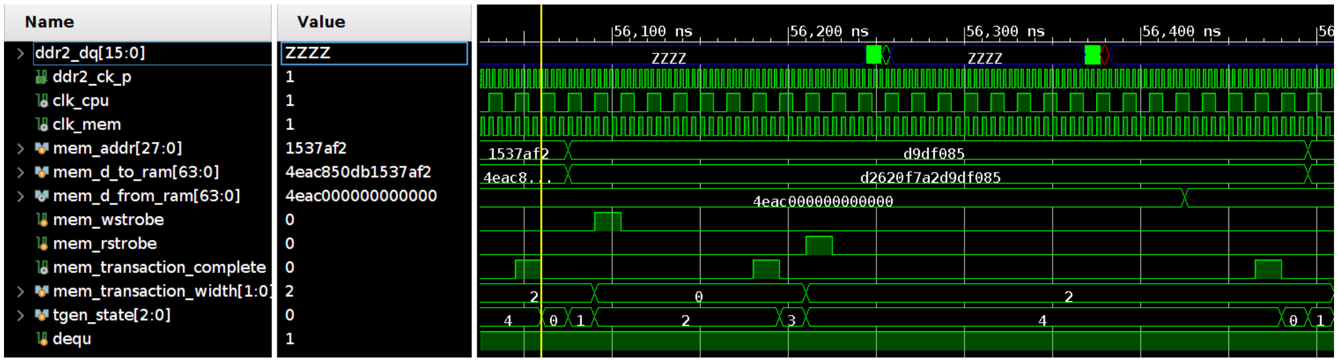


Name	Value
CLK100MHZ	1
CPU_RESETN	0
LED[15:0]	ZZZZ
ddr2_dq[15:0]	XXXX
ddr2_dqs_n[1:0]	X
ddr2_dqs_p[1:0]	X
ddr2_addr[12:0]	XXXX
ddr2_ba[2:0]	X
ddr2_ras_n	X
ddr2_cas_n	X
ddr2_we_n	X
ddr2_ck_p	X
ddr2_ck_n	X
ddr2_cke	X
ddr2_cs_n	X
ddr2_dm[1:0]	X
ddr2_odt	X
clk_cpu	1
clk_mem	1
pll_locked	X
rst_stretch[31:0]	7fffffff
reset_req_n	0
rst_n	0
mem_d_from_ram[63:0]	XXXXXXXXXXXXXXXX
mem_transaction_complete	0
mem_ready	0
mem_addr[27:0]	0000000
mem_d_to_ram[63:0]	0000000000000000
mem_transaction_width[1:0]	0
mem_wstrobe	0
mem_rstrobe	0
lfsr[31:0]	00000000
tgen_state[2:0]	0
dequ	0
TGEN_GEN_AD[2:0]	0
TGEN_WRITE[2:0]	1
TGEN_WWAIT[2:0]	2
TGEN_READ[2:0]	3
TGEN_RWAIT[2:0]	4

Two things you may initially notice: first there is a large delay during the simulation where only occasional memory accesses occur. This is due to the MIG IP's training process which calibrates against the DDR2 model on reset, just like in the physical implementation. Even though this is a simulation, an **abbreviated version of the calibration still occurs and cannot be disabled**. Fortunately this calibration is reduced from the order of several ms in real hardware down to ~65us during sim.

Next the *tgen_state* indicates that we are in a write transaction for an extended period of time. This is entirely expected. We have requested that the memory performs a write, and even through the MIG IP reports ready, it will delay completion of the transaction until the memory becomes available. As a result, the first write is stalled.

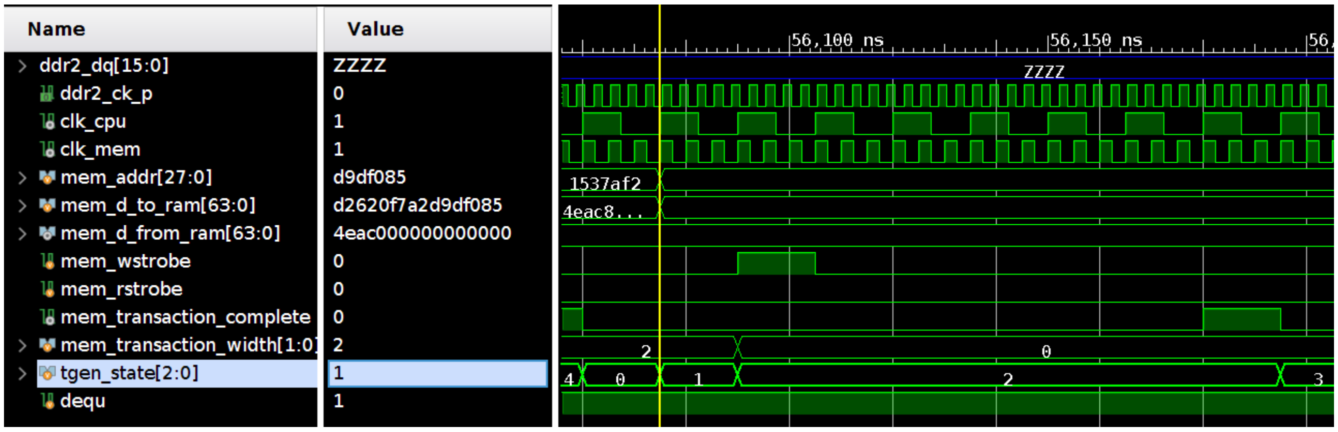
Now to examine one of the memory transactions:



Name	Value
ddr2_dq[15:0]	ZZZZ
ddr2_ck_p	1
clk_cpu	1
clk_mem	1
mem_addr[27:0]	1537af2
mem_d_to_ram[63:0]	4eac850db1537af2
mem_d_from_ram[63:0]	4eac000000000000
mem_wstrobe	0
mem_rstrobe	0
mem_transaction_complete	0
mem_transaction_width[1:0]	2
tgen_state[2:0]	0
dequ	1

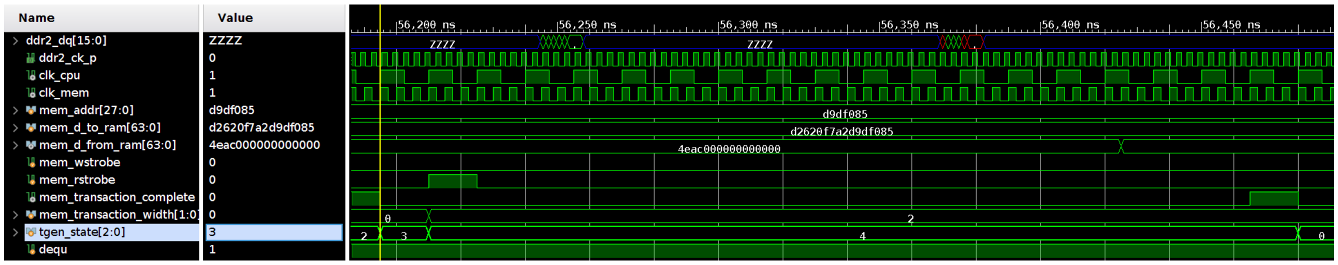
A randomly generated address along with data is generated from an LFSR. These values are captured and fed to the memory interface for a 64-bit write. We wait for the write to be accepted via *transaction_complete* (notice how quickly this occurs, as the MIG IP signals acceptance before it makes it to RAM visible by the relationship of *transaction_complete* to *ddr2_dq*).

Detail of the write:



Name	Value
ddr2_dq[15:0]	ZZZZ
ddr2_ck_p	0
clk_cpu	1
clk_mem	1
mem_addr[27:0]	d9df085
mem_d_to_ram[63:0]	d2620f7a2d9df085
mem_d_from_ram[63:0]	4eac000000000000
mem_wstrobe	0
mem_rstrobe	0
mem_transaction_complete	0
mem_transaction_width[1:0]	2
tgen_state[2:0]	1
dequ	1

Next we make a 16-bit read from the same address:



Name	Value
ddr2_dq[15:0]	ZZZZ
ddr2_ck_p	0
clk_cpu	1
clk_mem	1
mem_addr[27:0]	d9df085
mem_d_to_ram[63:0]	d2620f7a2d9df085
mem_d_from_ram[63:0]	4eac000000000000
mem_wstrobe	0
mem_rstrobe	0
mem_transaction_complete	0
mem_transaction_width[1:0]	0
tgen_state[2:0]	3
dequ	1

Once this read finishes, we compare the data returned with what was originally written. If it matches, LED[0] is lit via the *dequ* signal.

You may notice the odd "left justification" of bits when performing less than 64-bit transactions. The mem_example memory interface was adapted from a big-endian processor design where this arrangement simplified integration with other memory mapped peripherals. It can be changed or just accounted for when using the module. Still, mem_example will allow for byte-oriented addressing into memory with 8, 16, 32, or 64 bit reads and writes.

If all goes well during synthesis, running the example on a real board will result in LED[0] being lit almost immediately after reset. The LED will stay lit as long as the results from the DDR2 read match what was written.