Speedy DDR2 Customization Notes

Ray Bittner (raybit@microsoft.com)
Date Written: February 25, 2011
Revision: 1.1

This is a guide to customizing the Speedy DDR2 controller to a particular implementation. It is intended to be succinct and to the point, while still containing all necessary information. A more detailed explanation of the theory and operation can be found in the associated paper, or in the code comments.

The naming convention for used signals indicates whether they are active high or active low. A signal ending in _H is active high, and likewise a _L suffix indicates an active low signal.

1. Change Log

Version 1.0 – July 2, 2009

Version 1.1 – February 25, 2011

- Changed automatic calibration algorithm so that it uses distributed reads to achieve the user specified number of calibrations per second (CALIBRATIONS_PER_SECOND). The new method is more friendly to designs that may be sensitive to long pipeline stalls. Previously, the calibration state machine could stall the user for longer periods if the requisite number of reads did not occur in the user's command stream.
- Fixed a metastability bug in the calibration module where a DQS signal could be sampled as a '1' during the cycle/edge calibration stage and later sampled as a '0' at the same location by the IDELAY calibration stage. With rare combinations of boards/SODIMMs, this could cause initialization to hang. The fix to this bug adds significant guard bands to the calibration window such that the minimum allowable clock rate is now 154.8MHz.
- Fixed a bug in the read to write timing where an added clock cycle was required for proper bus turn around with some DDR2 modules.
- Added code for an arbiter that can be used to multi-port the Speedy DDR2 memory interface (Speedy_Memory_Arbiter_Module). The sample design includes ChipLevel.v files showing use with and without the arbiter.
- Courtesy of Christoph Borchert christoph.borchert@uni-dortmund.de, changed the default UCF file to support compilation for the popular Virtex 5 XUP board (V5LX110T). The original Xilinx ML505 UCF is still included.
- Supplied .bit file and project were compiled using ISE 12.4 for the XUP board (V5LX110T).

2. Housekeeping & Initialization

The following signals interface to user logic clocks and initialization logic.

- `input wire CLK_MEM_INTERFACE` – This is the primary clock that will be used internally and externally for the DDR2 interface. The internal user interface is synchronous to the positive edge of this clock, and this is the clock source that will be used for the external DDR2 interface as well.
- `input wire CLK_MEM_INTERFACE_270` – This clock should be an exact replica of `CLK_MEM_INTERFACE`, except that it should be phase shifted by 270 degrees using a PLL or DCM. It is to skew the DM (Data Mask or byte enable) and DQ (data) signals so that they are center aligned with the DQS (Data Strobe) signals.
- `input wire CLK_200MHz` – This clock must be very close to 200 MHz since it is used as the synchronization source for the IDELAY_CTRL primitive calibration.
- `input wire RESET_L` – An active low global reset signal that should be asserted once at power on, and whenever the input clocks become unstable after that.
- `output reg COMPONENT_READY_H` – This is an active high output signal that indicates when the Speedy DDR2 module is ready for use. It will be deasserted from the time that `RESET_L` is asserted until full calibration is attained.
- `input wire SYSTEM_READY_H` – An active high signal that is asserted by the user to indicate that all high level modules have indicated that they are fully initialized and that the Speedy DDR2 module should expect normal user commands. Note that this cannot be asserted until after the Speedy DDR2 module has asserted `COMPONENT_READY_H` high.
- `output wire [7:0] LEDS` – Debugging outputs from the Speedy DDR2 module that can be left disconnected when not used.

3. Internal User Interface

The following signals should be connected to user logic for the purpose of requesting read and write accesses to the DDR2 RAM.

- `input wire CmdReq_H` – This signal, along with `CmdAck_H`, are used as the flow control handshake between the Speedy DDR2 controller and the user code. A new command is transferred from the user code to the Speedy DDR2 controller on every positive clock edge that both signals are asserted high. `CmdReq_H` is asserted high by the user code whenever a command is to be sent to the Speedy DDR2 controller. The Speedy DDR2 controller will assert `CmdAck_H` high whenever it can accept a new command on that clock cycle. If both `CmdReq_H` and `CmdAck_H` are asserted high for multiple back to back cycles, then multiple commands will be accepted by the Speedy DDR2 controller.
- `output reg CmdAck_H` – See description of `CmdReq_H`.
- `input wire CmdWrite_L` – This indicates whether the new command is a write (logic 0) or a read (logic 1).
- `input wire [31:0] CmdAddress` – This indicates the desired address for the read or write. This is a 32-bit byte resolution address; however, it generally makes sense to supply addresses that are aligned to 32-byte boundaries. This is due to the fact that all memory accesses are performed in 32-byte bursts, and supplying a 32-byte aligned address will reference data in the order expected. If a non-aligned address is supplied, Speedy DDR2 will access the data using sequential 64-bit word addresses within the 32-byte block, with wrap around.
- `input wire [31:0] CmdDataMask_L` – These are the individual byte write enable signals for each byte in the 32-bit burst. Each byte whose write enable is logic 0 will be written, those whose byte enables are logic 1 will not. This is ignored for a read command.

- `input wire [255:0] CmdData` – The 32-byte data that should be written to the associated address during a write command. This is ignored for a read command.
- `output reg ReadDataValid_H` – The read data valid signal. Sometime after a read command is issued, this signal will assert high for exactly one clock cycle to indicate that the associated data is being returned. Note that all read accesses are returned strictly in the order that they were requested by the user code, and that the returned data for a given access is only valid for the single cycle that `ReadDataValid_H` is logic 1. This implies that read requests should not be issued until the user code is prepared to accept the result.
- `output reg [4*DDR2_DQ_WIDTH-1:0] ReadData` – The read data associated with the read access that was previously requested. This is valid only during the clock cycle that ReadDataValid_H is logic 1. Data is returned for a new/different read request on every clock cycle for which `ReadDataValid_H` is logic 1.

4. External DDR2 Interface

The following signals should be connected to the corresponding external pins leading the the DDR2 RAM. There should be very few limitations on which pins may be used for this purpose. They must support the IDDR/ODDR primitives, all of the DQS/DQ signals must support the IDELAY primitive, and they must support the 1.8V voltage standard for DDR2 using the SSTL18_I, SSTL18_II_DCI or DIFF_SSTL18_II_DCI I/O standards as noted in the .UCF file.

- `output [DDR2_CLK_WIDTH-1:0] DDR2_CK` – The positive phase clock outputs.
- `output [DDR2_CLK_WIDTH-1:0] DDR2_CK_N` – The negative phase clock outputs.
- `output wire [DDR2_CKE_WIDTH-1:0] DDR2_CKE` – Clock enable signals. These are always enabled by Speedy DDR2.
- `output wire [DDR2_ODT_WIDTH-1:0] DDR2_ODT` – On Die Termination control. This is always enabled by Speedy DDR2, except during initialization.
- `output wire [DDR2_CS_WIDTH-1:0] DDR2_CS_N` – Chip select signals. Only one chip select is currently supported by Speedy DDR2.
- `output wire DDR2_RAS_N` – Row Address Strobe, active low.
- `output wire DDR2_CAS_N` – Column Address Strobe.
- `output wire DDR2_WE_N` – Write Enable.
- `output wire [DDR2_DM_WIDTH-1:0] DDR2_DM` – Byte write enables.
- `output wire [DDR2_BANK_WIDTH-1:0] DDR2_BA` – Bank address select pins.
- `output wire [DDR2_ROW_WIDTH-1:0] DDR2_A` – Row/Column address select pins.
- `inout wire [DDR2_DQS_WIDTH-1:0] DDR2_DQS` – Data strobe signals, positive phase.
- `inout wire [DDR2_DQS_WIDTH-1:0] DDR2_DQS_N` – Data strobe signals, negative phase. The Speedy DDR2 controller configures the DDR2 to use both the positive and negative phase DQS signals.
- `inout wire [DDR2_DQ_WIDTH-1:0] DDR2_DQ` – Data signals. These are ganged in groups (typically 8 per group) and paired with a single DQS signal pair.

5. Architecture Parameters

These are the parameters that relate to the physical signal connections between the FPGA and the DDR2 RAM.

- DDR2_CLK_WIDTH – The number of clock outputs.
- DDR2_ROW_WIDTH – The number of bits needed to address all rows of the DDR2.
- DDR2_COL_WIDTH – The number of bits needed to address all locations within a given row. This will usually be $\log_2$(RAM Page Size) – 3, because the address only goes down to the burst word level (generally 8 bytes wide).
- DDR2_DQS_WIDTH – The number of DQS signal pairs.
- DDR2_DQ_WIDTH – The number of data bits, also the width of the data bus.
- DDR2_DM_WIDTH – The number of byte write enables. Speedy DDR2 currently requires this to be equal to DDR2_DQS_WIDTH.
- DDR2_BANK_WIDTH – The number of address bits needed to select the bank.
- DDR2_CKE_WIDTH – The number of clock enable outputs.
- DDR2_CS_WIDTH – The number of chip select signals. Currently Speedy DDR2 only supports one.
- DDR2_ODT_WIDTH – The number of On Die Termination control signals.
- DQ_PER_DQS – The number of DQ data bits that are ganged with a single DQS data strobe.
- DDR2_MAX_CS_BITS – Created with the future in mind. This should be set to be equal to DDR2_CS_WIDTH for now.
- DDR2_MAX_BANK_BITS – Created with the future in mind. This should be set to be equal to DDR2_BANK_WIDTH for now.
- DDR2_MAX_ADDR_BITS – Created with the future in mind. This should be set to max(DDR2_ROW_WIDTH, DDR2_COL_WIDTH) for now.

6. Timing Related Parameters

These parameters set the various timing constraints imposed by the DDR2 and will be found on the DDR2 datasheet. All times are expressed in terms of picoseconds unless otherwise noted.

- TIME_RP – Single bank precharge time (tRP).
- TIME_WR – Write Recovery Time (tWR).
- TIME_REFI – Average periodic refresh interval (tREFI).
- TIME_RFC – REFRESH to REFRESH or REFRESH to ACTIVE command interval (tRFC).
- CLOCKS_CL – CAS Latency (tCL), expressed in clock cycles.
- CLOCKS_AL – Additive latency (tAL), currently must be set to 0.
- BURST_LENGTH_SELECT – Burst length log2(). Currently must be set to 2.
- BURST_LENGTH – Burst length. Currently must be set to 4.
- TIME_RCD - ACTIVE to READ or WRITE delay (tRCD).
- TIME_RC - ACTIVE to ACTIVE delay (same bank, tRC).
- TIME_RRD - ACTIVE bank a to ACTIVE bank b delay (tRRD).
- TIME_RTP – READ to PRECHARGE delay (tRTP).
- TIME_WTR – WRITE to READ delay (tWTR).
- TIME_RAS_MIN – ACTIVE to PRECHARGE delay (tRAS).
- NUM_IDELAYCTRL - The number of IDELAYCTRLs that are used. This is determined by how the external bus pins are connected, and in which clock domains they reside. Please see the customization instructions below.
- ODDR_CLK_TO_PAD_DELAY - The clock to pad output delay. Used for simulation purposes only in order to test the calibration mechanism.

- TIME_MAX_CALIBRATION_INTERVAL - The maximum interval that may elapse between read calibration locks. If this timer expires without a lock occurring, the state machine will send down a string of reads until a new lock is obtained. Note that this can negatively affect performance to some extent when the user wishes to perform a long string of writes, so it is best not to force calibrations too often.
- CLOCKS_RAS_MAX – This is the maximum amount of time that is allowed to elapse from the time that a row is opened (ACTIVE) until it is closed (PRECHARGE). This is maintained per bank and is expressed in terms of clock cycles because the time in picoseconds is too large. Found as the maximum allowed value for tRAS in the datasheet.
- CLK_PERIOD – The clock period of CLK_MEM_INTERFACE.

7. Notes & Instructions

This is the list of potential "gotchas" that may come up during the process of customizing the Speedy DDR2 controller for a particular implementation.

- The Speedy DDR2 controller enables and uses the DQS pins in differential signaling mode. Therefore, both the positive and negative logic DQS signals must be connected. Note that the DDR2 initialization parameters and input DQS signal receivers could be changed to allow for single ended drive.

- The IDELAYCTRL blocks must be manually placed and constrained because there is no reliable automated way to do this. If they are left unconstrained, the tool assumes that one design module uses all of them and may not behave correctly if they are used by two different modules that make use of them in the same design. To do this, first open floor plan editor to determine which IDELAYCTRL blocks correspond to the pins that you have chosen. There should be one IDELAYCTRL primitive per clock region. Every IDELAYCTRL that exists in the same region as a DQ, DQS or DQS# pin in your design must be constrained. That is, the IDELAYCTRL for a given IO block must be included for any pin that uses an IODELAY in the design.

  Once the set of IDELAYCTRL primitive locations has been identified, change the NUM_IDELAYCTRL parameter in the Speedy_DDR2_Module to reflect the correct number of sites. Then add a constraint to the .UCF file for each instance to provide its exact location, such as:

  ```
  INST "DDR2_Interface/.IDELAYCTRLs[0].IDELAYCTRL_Inst" LOC = "IDELAYCTRL_X0Y0" ;
  INST "DDR2_Interface/.IDELAYCTRLs[1].IDELAYCTRL_Inst" LOC = "IDELAYCTRL_X0Y1" ;
  INST "DDR2_Interface/.IDELAYCTRLs[2].IDELAYCTRL_Inst" LOC = "IDELAYCTRL_X0Y5" ;
  ```

  The order that they are listed does not matter. It can be difficult to identify the exact IDELAYCTRL locations that are required, so it is easiest to include more than necessary, and then search for a warning in the PAR phase indicating which ones are not needed. Such as:

  ```
  WARNING:Place:851 - The delay controller "DDR2_Interface/.IDELAYCTRLs[5].IDELAYCTRL_Inst" has been locked with
  the following location constraint:
  ```

COMP "DDR2_Interface/.IDELAYCTRLs[5].IDELAYCTRL_Inst" LOCATE = SITE "IDELAYCTRL_X0Y3" LEVEL 1
However, none of the delay elements calibrated by this controller are being used. The controller will still use up a global clock resource from the clock region and consume power. Please refer to the usage document to use the controller efficiently.

The `NUM_IDELAYCTRL` parameter can then be updated and the unnecessary constraints can then be removed.

- The timing for the several output pins is block is based on CLK_MEM_INTERFACE_270, while the rest of the design runs on CLK_MEM_INTERFACE. CLK_MEM_INTERFACE_270 lags CLK_MEM_INTERFACE by 270 degrees of phase. Depending on how CLK_MEM_INTERFACE_270 is generated, it may be necessary to define this relationship explicitly so that the tools will have the correct constraint indicating that there is only ¾ of a clock cycle from CLK_MEM_INTERFACE to CLK_MEM_INTERFACE_270. Normally, the tools can derive that constraint from the PLL declaration during clock generation, but you need to check that it does by starting the timing analyzer after the Implementation phase and checking for a derived constraint such as:

  TS_ClockGenerator_CLK_140MHz_270_Raw = PERIOD TIMEGRP
  "ClockGenerator_CLK_140MHz_270_Raw" TS_CLK_33MHz / 4.25 PHASE 5.347 ns HIGH 50%;

  Note the phase term, which is ¾ of a clock period at 140MHz.

- The DQS signals are differential pairs, the DQ signals are single wires. When these DQS signals are driven by the external DDR2 chip during a read, an internal differential receiver is used in the input path that is not present for the DQ signals. This introduces an additional 1.5ns or so to the input delay that is seen on the DQS signals as opposed to the DQ signals. Since the timing of the DQS signals is used to calibrate the capture of the DQ signals, this could be a problem for high clock speeds. At lower clock frequencies, it shouldn't be a problem. For example, at 140MHz, the clock period is 7.143ns, so the valid data window is a little less than 3.571ns. The difference between DQS and DQ induced by the differential receiver is about 1.5ns, so instead of having the capture centered in the data valid window, it will be skewed by 1.5ns, which should still be acceptable when the window is 3.571ns wide. At higher frequencies, this will not be the case, and the adjustment calculation that is done in the `DDR2_Calibration_Module` would need to take the 1.5ns skew into account. The design has been tested at speeds up to 222MHz and this adjustment has not been necessary.

- While attempting to drive the DDR2 clock pins, I have encountered this error from the 10.1 SP1 tools:

  > ERROR:Place:645 - A clock IOB clock component is not placed at an optimal clock IOB site. The clock IOB component <DDR2_CK<0>> is placed at site <AK29>. The clock IO site can use the fast path between the IO and the Clock buffer/GCLK if the IOB is placed in the master Clock IOB Site. If this sub optimal condition is acceptable for this design, you

may use the CLOCK_DEDICATED_ROUTE constraint in the .ucf file to demote this message to a WARNING and allow your design to continue. However, the use of this override is highly discouraged as it may lead to very poor timing results. It is recommended that this error condition be corrected in the design. A list of all the COMP.PINs used in this clock placement rule is listed below. These examples can be used directly in the .ucf file to override this clock rule.

< NET "DDR2_CK<0>" CLOCK_DEDICATED_ROUTE = FALSE; >

When that constraint is used in the .UCF file, the tools throw another error and the only viable solution is to install 9.2i. If 9.2i is used the environment variable XIL_PLACE_ALLOW_LOCAL_BUFG_ROUTING must be set in order to avoid the same error coming up with that version. If version 10.1 SP2 is used, this error does not come up.

- Version 9.2i tools contain a bug where they do not correctly account the output of the IOBUFDS as a driver and fail with an error indicating that there is no driver on the DDR2 clock nets. In order to fix this, install the patch from Xilinx answer record 30035.
  http://www.xilinx.com/support/answers/30035.htm
  This does not happen with version 10.1i.

- The DQS tracking algorithm assumes that the leading edge of a sensed DQS two pulse sequence will arrive no earlier than ¼ clock cycle after the internal clock edge with which it should coincide. This is valid because the ODDR clock to pad time is generally at least ½ of a clock cycle, which means that the DDR2 should not even see the positive clock edge until the negative edge of the FPGA's internal clock. The tDQSS parameter states that the DDR2 can return the associated DQS signal +/- ¼ clock cycle from that clock edge. So, if the calibration search looks back ¼ clock cycle from the negative edge of the FPGA internal clock, an early DQS edge should not be missed. It was not possible to start looking back from the positive edge of the FPGA internal clock because the clock to pad output delay of the ODDR driver was so long that the DQS signal was still appearing to be tri-stated and sampling as 1 for any delay value. This condition would prematurely end the DQS search, so the search now begins from the first internal negative edge of the clock.