Creating HyperLynx DDRx Memory Controller Timing Model

By: Min Maung
Last Modified: April 30, 2009

1.0 Introduction

The DRAM and Controller timing models are required by the HyperLynx 8.0 DDRx Wizard to specify the timing requirements at both ends for the interfacing signals. The DDRx Wizard uses the parameters from the timing models for deriving the final timing margins. These timing parameters include skew, delay, setup and hold time requirements on signals with respect to their associated strobe or clock signal. The DRAM timing models are quite standardized since the timing specifications at the DRAMs are specified by the JEDEC standards. In the case of the controllers, it is very likely that the timing requirements are different from vendor to vendor.

The default controller timing models for each DDRx interface (ddr_ctl.v, ddr2_ctl.v, and ddr3_ctl.v) are included with HyperLynx in the folder such as C:\MentorGraphics\release\SDD_HOME\hyperlynx\Libs. These default timing models can be used to perform simulations using the DDRx Wizard quickly. If your controller’s timing parameters differ significantly from the default model, you will need to create your own model to obtain the accurate timing simulation results. To help you with this process, HyperLynx includes the HyperLynx Timing Model Wizard and the HyperLynx Timing Model Editor. The HyperLynx Timing Model Wizard guides you through the process of creating the timing model. The HyperLynx Timing Model Editor allows you to edit, check for syntax, and view the timing relationships in the form of timing diagrams.

In this application note, firstly the DDRx timing relationships in each signal group are reviewed. Secondly, it defines and explains the timing parameters that are required by the DDRx Wizard at the controller. Thirdly, it provides guidance on how to find the relevant timing parameters from three different controller datasheets and translate them into the required DDRx Wizard controller parameters. Finally, it shows the methodology for creating the controller timing model by entering those parameters in the HyperLynx Timing Model Wizard (GUI based) or in the HyperLynx Timing Model Editor (syntax-based).

2.0 Review on DDRx on Timing Relationships

This section reviews the DDRx specific timing relationships between signals at the controller during read and write operations.

In the DDRx memory interface, there are four main groups of signals:

- Address and Command Signals (A[15:0], BA[2:0], RAS#, CAS# & WE#)
- Control Signals (CS[3:0], CKE[3:0], & ODT[3:0])
- Data, Data Mask, and Strobe signals (DQ[63:0], DM[8:0], CB[7:0], DQS/#DQS[8:0])
- Clock Signals (CK/CK#[5:0])
First, the operating frequency of the clock signal must be determined. The DDRx interface is usually specified by notation DDRx–<speed-grade>, for example DDR-400, DDR2-533 or DDR3-800. The speed-grade unit is in MT/s (mega-transfer per second). In this application note, DDR3-800 will be used as an example throughout. In this case, the speed-grade is 800MT/s, and the operating frequency is half of the speed-grade; 400MHz. The bit period for address/command/control and data signals is needed information.

The address, command and control signals are output only signals but are used for both read and write operations. The address, command and control signals reference an associated clock (CK) signal. 1T or 2T timing is allowed on address and command signals but only 1T timing is allowed on control signals. Typically, the controller outputs these signals approximately aligned with a falling clock edge. To achieve this, the address/command/control signal is launched half the clock period earlier than the clock. This timing is referred to as 1T timing. For 2T timing, the address/command signal is launched one and a half period earlier than the clock so that there is more setup time at the receiver. So, the shortest available setup time for the address signal for 1T timing is one clock period and for 2T timing, it is 2 clock periods. Figure 1 is the timing diagram showing the timing relationships explained above. The gray windows are the uncertainties that need to be accounted for in the real operation.

The data, data mask and strobe signals are bi-directional signals: outputs during the write operation and inputs on the read operation. Each data byte lane references a DQS signal. During the write operation, DQ and DM signals are launched a quarter of the clock period earlier than the DQS signal for each byte lane so that both rising and falling edges of the strobe are centered in the valid data bit window. Since the data and data mask signals can be clocked in at every strobe edge, the bit period for these signals is half of the clock period. Figure 2 shows the above described timing relationship. The gray windows are the uncertainties that need to be accounted for in real operation.
During the write operation, the controller also needs to meet the delay timing relationship between the clock signals (CK) and strobe signals (DQS). This relationship is defined in Figure 3.

**Figure 3: DDRx CK and DQS Timing at the Controller**

During the read operation, the DRAMs send both DQ signals and DQS signals to the controller, edge aligned. At the controller, these signals must meet certain setup and hold requirements. The setup and hold requirements at the controller can be specified in two ways: at the pins or at the internal registers. If the timing requirements are at the pins, DQ/DM signals and DQS signals are edge aligned. If the timing requirements are specified at the internal registers, information on how the controller captures the data internally is needed. Typically, the ideal phase shift is one quarter of the clock cycle. Figure 4 shows the above described timing relationships.

**Figure 4: DDRx DQ/DM Signals Timing at the Controller during Read**

Table 1 shows the summary of the timing relationships in a DDRx interface.

**Table 1: Summary of DDRx Timing Relationships at the Controller**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Signals</th>
<th>Ref. Signal</th>
<th>Ref. Signal Freq</th>
<th>Clocked Edge at Receiver</th>
<th>Typical Signal Prelaunch Delay Relative to CK or DQS</th>
<th>Timing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read/Write</td>
<td>A/Cmd</td>
<td>CK+/–</td>
<td>0.5*DataRate</td>
<td>Rise</td>
<td>0.5*tCK</td>
<td>1T or 2T</td>
</tr>
<tr>
<td>Read/Write</td>
<td>Ctl</td>
<td>CK+/–</td>
<td>0.5*DataRate</td>
<td>Rise</td>
<td>0.5*tCK</td>
<td>1T</td>
</tr>
<tr>
<td>Write</td>
<td>DQ/DM</td>
<td>DQS+/–</td>
<td>0.5*DataRate</td>
<td>Rise &amp; Fall</td>
<td>0.25*tCK</td>
<td>NA</td>
</tr>
<tr>
<td>Write</td>
<td>DQ/DM</td>
<td>CK+/–</td>
<td>0.5*DataRate</td>
<td>Rise &amp; Fall</td>
<td>0</td>
<td>NA</td>
</tr>
<tr>
<td>Read</td>
<td>DQ/DM</td>
<td>DQS+/–</td>
<td>0.5*DataRate</td>
<td>Rise &amp; Fall</td>
<td>0</td>
<td>NA</td>
</tr>
</tbody>
</table>

**3.0 Controller Timing Parameters Required by the DDRx Wizard**

For any controller, the DDRx Wizard requires the parameters that are described in this section. The same parameters are also required if you want to create a controller model using the HyperLynx Timing Model Wizard.

**3.1 tCKAC(min) and tCKAC(max)**

- Applicable to address (A) and command signals (BA, RAS, CAS, and WE) on both read and write cycles
The minimum and maximum delay between the address and command output signals, transitioning to valid before the rising edge of the output clock (CK)

- Allows 1T (Figure 5) or 2T (Figure 6) timing on these signals
- If the delay between a valid transition of the Address and Command signals and the rising edge of the clock is more than one clock cycle, it is referred to as 2T timing.

Figure 5: tCKAC (min) and tCKAC(max) (1T Timing)

Figure 6: tCKAC (min) and tCKAC(max) (2T Timing)

3.2 tCKCTL(min) and tCKCTL(max)
- Applicable to control signals (CS, CKE, and ODT) on both read and write cycles.
- The minimum and maximum delay between the control output signals transitioning to valid before the rising edge of the output clock.
- Allows only 1T timing (Figure 7) on these signals.

Figure 7: tCKCTL(min) and tCKCTL(max)

3.3 tCKDQS(min) and tCKDQS(max)
- Applicable to CK and DQS signals during the WRITE cycle
- The minimum and maximum skew (Figure 8) between the rising edge of the output data strobe (DQS) and the rising edge of the output clock (CK).

Figure 8: tCKDQS(min) and tCKDQS(max)

3.4 tDQSDQ(min) and tDQSDQ(max)
- Applicable to data(DQ) and data mask (DM) signals and its associated DQS signal during the WRITE cycle
- The minimum and maximum delay between the data and data mask output signals transitioning to valid and the associated output data strobe edge (either rising or falling)

Figure 9: tDQSDQ(min) and tDQSDQ(max)
3.5 tDS and tDH
- Applicable to data(DQ) signals and its associated DQS signal during the READ cycle
- Data setup (tDS) and hold (tDH) window relative to associated DQS
  - During a read cycle, the DRAMs output the transitions of the DQS signal approximately aligned with the transitions of the DQ signals. The controller captures the DQ signals by typically phase shifting the DQS signal internally by about 1/4 clock period.
  - Specify whether the "setup" and "hold" parameters are measured at the controller pins (phase shift=0) (Figure 10) or after an internal 90° phase shift (Figure 11).
  - Setup values are positive if DQ must be valid before DQS (negative if after); Hold values are positive if DQ must remain valid until after DQS (negative if before).

4.0 Interpreting and Deriving Required Timing Parameters from Datasheet
In this section, three examples are shown in deriving the HyperLynx DDRx required parameters from the datasheet parameters.

These derivations are required since each chip vendor specifies the timing parameters for the controller in different ways. Some of the variations are:
- Specified either at the internal registers or at the pin-pad
- Specified as a skew or as a valid window
- Relative to a clock/strobe rising or falling edge

4.1 Example 1: The HyperLynx Default ddr3_ctl Controller Model
In the default controller model, ddr3_ctl.v, the parameters shown in Table 2 were assumed to be defined in the controller datasheet and Figure 12 shows the timing diagram of these parameters.

| Table 2: Default ddr3_ctl Controller Datasheet Parameters Definitions |
|---------------------------------|---------------------------------|
| Parameters                        | Definitions                      |
| tACCSkew                          | Output delay skew from CK falling to Addr/Cmd/Ctl (+/−) |
| tCKDQS                            | Output delay skew from CK rising to DQS rising (+/−) |
| tDQSDQQ                           | Output delay skew from DQS to DQ (+/−) |
| tDS                               | Minimum DQ to DQS setup time, with 1/4 cycle DQS shift |
| tDH                               | Minimum DQS to DQ hold time, with 1/4 cycle DQS shift |
The following sections show the derivations of the DDRx Wizard required parameters from the above default controller’s datasheet parameters.

4.1.1 Deriving tCKAC(min) and tCKAC(max) for DDR3 Default Timing Model
You need tACCSkew to derive tCKAC(min) and tCKAC(max). The timing relationships between those parameters are shown in Figure 13.

To derive tCKAC(min) and tCKAC(max) from tACCSkew, ACDlyMin and tACDlyMax parameters are defined as follows and also shown in Figure 13:

\[ t\text{ACDlyMin} = \text{Delay from CK rising out to Addr/Cmd/Ctl invalid (min)} \]

\[ t\text{ACDlyMax} = \text{Delay from CK rising out to Addr/Cmd/Ctl invalid (max)} \]
### 4.1.2 Deriving \(t\text{CKCTL}(\text{min})\) and \(t\text{CKCTL}(\text{max})\) for DDR3 Default Timing Model
The parameters \(t\text{CKCTL}(\text{min})\) and \(t\text{CKCTL}(\text{max})\) are derived exactly the same way as \(t\text{CKAC}(\text{min})\) and \(t\text{CKAC}(\text{max})\) as shown in Figure 13.

### 4.1.3 Deriving \(t\text{CKDQS}(\text{min})\) and \(t\text{CKDQS}(\text{max})\) for DDR3 Default Timing Model
The parameter \(t\text{CKDQS}\) is defined in the default controller datasheet in the exact same way as needed by the DDRx Wizard as shown in Figure 14.

### 4.1.4 Deriving \(t\text{DQSDQ}(\text{min})\) and \(t\text{DQSDQ}(\text{max})\) for DDR3 Default Timing Model
You will need \(t\text{DQSDQQ}\) parameter from the DDR3 Default Timing Model datasheet to derive \(t\text{DQSDQ}(\text{min})\) and \(t\text{DQSDQ}(\text{max})\). The timing relationships between those parameters are shown in Figure 15.
4.1.5 Deriving tDS and tDH for DDR3 Default Timing Model

The parameters tDS and tDH are defined in the default controller datasheet exactly the same way as the DDRx Wizard required timing parameters as shown in Figure 16.

Therefore, you will not need to derive these parameters. Note that tDQDQS(min) and tDQDQS(max) parameters (shown in Figure 16) are derived from tDS and tDH or vice versa. The DDRx Wizard only needs those parameters to be defined using either tDQDQS(min) and tDQDQS(max) or tDS and tDH.

4.1.6 Summary of DDRx Wizard Required Timing Parameters for DDR3 Default Timing Model

Table 8 shows the summary of derived timing parameters which are needed by the DDRx Wizard. These parameters can be entered directly into the HyperLynx Timing Model Wizard which will be covered in the next section.
Table 8: Summary of DDRx Wizard Required Timing Parameters DDR3 Default Timing Model

<table>
<thead>
<tr>
<th>DDRx Wizard Required Parameters</th>
<th>Min (ps)</th>
<th>Max (ps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>tCKAC</td>
<td>–1550</td>
<td>–950</td>
</tr>
<tr>
<td>tCKCTL</td>
<td>–1550</td>
<td>–950</td>
</tr>
<tr>
<td>tCKDQS</td>
<td>–150</td>
<td>150</td>
</tr>
<tr>
<td>tDQSDQ</td>
<td>–775</td>
<td>–475</td>
</tr>
<tr>
<td>tDS</td>
<td>150</td>
<td>–</td>
</tr>
<tr>
<td>tDH</td>
<td>250</td>
<td>–</td>
</tr>
</tbody>
</table>

4.2 Example 2: MPC8544E PowerQUICC™ III Integrated Processor Controller Model

In this example, each of the following section shows the derivation of DDRx Wizard required parameters from the Freescale MPC8544E PowerQUICC™ III Integrated Processor datasheet parameters.

The following diagrams are screenshots of the output timing parameters and the diagrams for Freescale MPC8544E PowerQUICC™ III Integrated Processor:

**Figure 17: MPC8544E Output Timing Diagram**

![MPC8544E Output Timing Diagram](image)

**Figure 18: MPC8544E Output Timing Parameters**

![MPC8544E Output Timing Parameters](image)
4.2.1 Deriving $t_{CKAC}(\text{min})$ and $t_{CKAC}(\text{max})$ for MPC8544E

Figure 17 and Figure 18 shows that the needed timing parameters to derive $t_{CKAC}(\text{min})/(\text{max})$ are $t_{DDKHAS}$ and $t_{DDKHAX}$. In this derivation, the 533MHz case will be used. Note that the 533MHz in this case is not the frequency of the clock but it is rather the speed grade for the DDRx interface. The actual frequency of the clock is half of 533MHz which is equal to 267MHz.

In this derivation, the 533MHz case will be used. Note that the 533MHz in this case is not the frequency of the clock but it is rather the speed grade for the DDRx interface. The actual frequency of the clock is half of 533MHz which is equal to 267MHz.

From the definition of $t_{DDKHAS}$ and $t_{DDKHAX}$, they are specified as output setup and hold parameters. As you will recall from section 3.1, $t_{CKAC}(\text{min})$ and $t_{CKAC}(\text{max})$ parameters are defined as delay between the address and command output signals, transitioning to "valid" before the rising edge of the output clock.

Figure 19 shows the relationship between the DDRx Wizard needed parameters $t_{CKAC}(\text{min})/(\text{max})$ and the datasheet parameters ($t_{DDKHAS}$ and $t_{DDKHAX}$). Table 9 shows the needed derivations.
Table 9: Deriving tCKAC(min) and tCKAC(max) for MPC8544E

<table>
<thead>
<tr>
<th>SpeedGrade</th>
<th>tCK</th>
<th>tDDKHAS</th>
<th>tDDKHAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDR2-533</td>
<td>1/267MHz = 3750 ps</td>
<td>1480 ps</td>
<td>1480 ps</td>
</tr>
</tbody>
</table>

\[ t_{CKAC(min)} = -t_{CK} + t_{DDKHAX} = -3750 \text{ ps} + 1480 \text{ ps} = -2270 \text{ ps} \]

\[ t_{CKAC(max)} = -t_{DDKHAS} = -1480 \text{ ps} \]

4.2.2 Deriving tCKCTL(min) and tCKCTL(max) for MPC8544E

From the definition of tDDKHCS and tDDKHCX in Figure 18 and Figure 17, they are specified as output setup and hold parameters. As you will recall from section 3.2, tCKCTL(min) and tCKCTL(max) parameters are defined as delay between the control output signals, transitioning to "valid" before the rising edge of the output clock.

Figure 20 and Table 10 show the timing relationships and the derivations.

Table 10: Deriving tCKCTL(min) and tCKCTL(max) for MPC8544E

<table>
<thead>
<tr>
<th>SpeedGrade</th>
<th>tCK</th>
<th>tDDKHCS</th>
<th>tDDKHCX</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDR2-533</td>
<td>1/267MHz = 3750 ps</td>
<td>1480 ps</td>
<td>1480 ps</td>
</tr>
</tbody>
</table>

\[ t_{CKCTL(min)} = -t_{CK} + t_{DDKHCX} = -3750 \text{ ps} + 1480 \text{ ps} = -2270 \text{ ps} \]

\[ t_{CKCTL(max)} = -t_{DDKHCS} = -1480 \text{ ps} \]

4.2.3 Deriving tCKDQS(min) and tCKDQS(max) for MPC8544E

From the definition of tDDKHMH in Figure 18 and Figure 21, tCKDQS(min) is equivalent to tDDKHMH(min) and tCKDQS(max) is equivalent to tDDKHMH(max).

Figure 22 and Table 11 show the timing relationships and derivations of the above parameters.
Table 11: Deriving tCKDQS(min) and tCKDQS(max) for MPC8544E

<table>
<thead>
<tr>
<th>SpeedGrade</th>
<th>tCK</th>
<th>tDDKHMH(min)</th>
<th>tDDKHMH(max)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDR2-533 1/267MHz = 3750 ps</td>
<td>0.5*tCK = 1875 ps</td>
<td>–600 ps</td>
<td>600 ps</td>
</tr>
</tbody>
</table>

4.2.4 Deriving tDQSDQ(min) and tDQSDQ(max) for MPC8544E

As you can see from Figure 17 and Figure 18, the related timing parameters to derive tDQSDQ(min) and tDQSDQ(max) are tDDKHDS/tDDKLDS and tDDKHD/tDDKLDX.

Table 12: Deriving tDQSDQ(min) and tDQSDQ(max) for MPC8544E

<table>
<thead>
<tr>
<th>SpeedGrade</th>
<th>tCK</th>
<th>tDDKLDS/ tDDKHDS</th>
<th>tDDKLDX/ tDDKHD</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDR2-533 1/267MHz = 3750 ps</td>
<td>0.5*tCK = 1875 ps</td>
<td>538 ps</td>
<td>538 ps</td>
</tr>
</tbody>
</table>

4.2.5 Deriving tDS and tDH for MPC8544E

Referring to Figure 24 and Figure 25, the DDRx Wizard needed parameters tDS and tDH can be derived from the tCISKEW(min) and tCISKEW(max).

As you can see in the Notes 2 of Figure 24, tDISKEW = (T/4–abs(tCISKEW)). This gives you a clue that DQS is shifted by tCK/4 with relative to DQ.
Figure 24: MPC8544E Input AC Timing Specifications

Table 18. DDR SDRAM Input AC Timing Specifications

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controller skew for MDQS—MDQ/MECC/MDM</td>
<td>$t_{CISKEW}$</td>
<td>533 MHz</td>
<td>~300</td>
<td>300</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>400 MHz</td>
<td>~365</td>
<td>365</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>333 MHz</td>
<td>~390</td>
<td>390</td>
<td></td>
</tr>
</tbody>
</table>

Notes:
1. $t_{CISKEW}$ represents the total amount of skew consumed by the controller between MDQS[n] and any corresponding bit that will be captured with MDQS[n]. This should be subtracted from the total timing budget.
2. The amount of skew that can be tolerated from MDQS to a corresponding MDQ signal is called $t_{CISKEW}$. This can be determined by the following equation: $t_{CISKEW} = \pm (T/4 - abs(t_{CISKEW}))$, where $T$ is the clock period and $abs(t_{CISKEW})$ is the absolute value of $t_{CISKEW}$, See Figure 3.
3. Maximum DDR1 frequency is 400 MHz.

Figure 25: MPC8544E Input Timing Diagram

Figure 26 and Table 14 show the timing relationships and derivations of $t_{DS}$ and $t_{DH}$ parameters.

Figure 26: tDS & tDH Timing Diagram for MPC8544E

Table 13: Deriving tDS and tDH for MPC8544E

<table>
<thead>
<tr>
<th>SpeedGrade</th>
<th>tCK</th>
<th>$t_{CISKEW(min)}$</th>
<th>$t_{CISKEW(max)}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDR2-533</td>
<td>1/267MHz = 3750 ps</td>
<td>~300 ps</td>
<td>300 ps</td>
</tr>
</tbody>
</table>

$|t_{CISKEW(min)}| = 300 ps $ $|t_{CISKEW(max)}| = 300 ps$
4.2.6 Summary of DDRx Wizard Needed Parameters for MPC8544E

Table 14: Summary of DDRx Wizard Required Timing Parameters for MPC8544E

<table>
<thead>
<tr>
<th>DDRx Wizard Required Parameters</th>
<th>Min (ps)</th>
<th>Max (ps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>tCKAC</td>
<td>– 2270</td>
<td>– 1480</td>
</tr>
<tr>
<td>tCKCTL</td>
<td>– 2270</td>
<td>– 1480</td>
</tr>
<tr>
<td>tCKDQS</td>
<td>– 600</td>
<td>600</td>
</tr>
<tr>
<td>tDQSDQ</td>
<td>– 1337</td>
<td>– 538</td>
</tr>
<tr>
<td>tDS</td>
<td>300</td>
<td>–</td>
</tr>
<tr>
<td>tDH</td>
<td>300</td>
<td>–</td>
</tr>
</tbody>
</table>

4.3 Example 3: Chipset Memory Controller

In this example, the memory controller model is of a chipset from a major vendor. Table 15 defines the parameters that were provided in this chipset controller datasheet and Figure 27 shows the timing diagram of those parameters.

Table 15: Chipset Memory Controller Datasheet Parameters Definitions

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>tCMDvb</td>
<td>minimum time Addr/Cmd (+/–) will become valid before CK rising</td>
</tr>
<tr>
<td>tCMDva</td>
<td>minimum time Addr/Cmd (+/–) will remain valid after CK rising</td>
</tr>
<tr>
<td>tCNTLvb</td>
<td>minimum time Control (+/–) will become valid before CK rising</td>
</tr>
<tr>
<td>tCNTLva</td>
<td>minimum time Control (+/–) will remain valid after CK rising</td>
</tr>
<tr>
<td>tDSSctlr</td>
<td>DQS fall to CK rise (min, write cycles)</td>
</tr>
<tr>
<td>tDSHctr</td>
<td>CK rise to DQS fall (min, write cycles)</td>
</tr>
<tr>
<td>tDVB</td>
<td>Output delay skew from CK rising to DQS rising (+/–)</td>
</tr>
<tr>
<td>tDVA</td>
<td>Output delay skew from DQS to DQ (+/–)</td>
</tr>
<tr>
<td>tSUctlr</td>
<td>Minimum DQ to DQS setup time, with 0 cycle DQS shift</td>
</tr>
<tr>
<td>tHDctlr</td>
<td>Minimum DQS to DQ hold time, with 0 cycle DQS shift</td>
</tr>
</tbody>
</table>

Figure 27: Chipset Memory Controller Datasheet Parameters Timing Diagram

4.3.1 Deriving tCKAC(min) and tCKAC(max) for DDR3 Chipset Memory Controller

The derivations in Table 16 were done by referring to Figure 28. Note that this controller uses 2T timing on the Address/Command signals.
Table 16: Deriving tCKAC(min) and tCKAC(max) for Chipset Memory Controller

<table>
<thead>
<tr>
<th>SpeedGrade</th>
<th>tCK</th>
<th>tCMDvb</th>
<th>tCMDva</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDR3-800</td>
<td>2500 ps</td>
<td>3302 ps</td>
<td>1244 ps</td>
</tr>
</tbody>
</table>

\[
-2*tCK + tCMDva = -5000 + 1244 = -3756 ps \\
tCMDvb = -3302 ps
\]

Figure 28: tCKAC(min) and tCKAC(max) Timing Diagrams for Chipset Memory Controller

4.3.2 Deriving tCKCTL(min) and tCKCTL(max) for Chipset Memory Controller
The derivations in Table 17 were done by referring to Figure 29.

Table 17: Deriving tCKCTL(min) and tCKCTL(max) for Chipset Memory Controller

<table>
<thead>
<tr>
<th>SpeedGrade</th>
<th>tCK</th>
<th>tCNTL_vb</th>
<th>tCNTL_va</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDR3-800</td>
<td>2500 ps</td>
<td>802 ps</td>
<td>1244 ps</td>
</tr>
</tbody>
</table>

\[
-tCK + tCNTL_va = -2500 + 1244 = -1256 ps \\
tCNTL_vb = -802 ps
\]

Figure 29: tCKCTL(min) and tCKCTL Timing Diagrams for Chipset Memory Controller

4.3.3 Deriving tCKDQS(min) and tCKDQS(max) for Chipset Memory Controller
Refer to Figure 30 for the derivations in Table 18.

Table 18: Deriving tCKDQS(min) and tCKDQS(max) for Chipset Memory Controller

<table>
<thead>
<tr>
<th>SpeedGrade</th>
<th>tCK</th>
<th>tDSSctlr</th>
<th>tDSHctlr</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDR3-800</td>
<td>2500 ps</td>
<td>1173 ps</td>
<td>920 ps</td>
</tr>
</tbody>
</table>

\[
tDSHctlr – 0.5*tCK = 920 – 1250 = -330 ps \\
0.5*tCK – tDSSctlr = 1250 – 1173 = 77 ps
\]
4.3.4 Deriving \( t_{DQSDQ}(\text{min}) \) and \( t_{DQSDQ}(\text{max}) \) for Chipset Memory Controller

Refer to Figure 31 for the derivations in Table 19.

**Table 19: Deriving \( t_{DQSDQ}(\text{min}) \) and \( t_{DQSDQ}(\text{max}) \) for Chipset Memory Controller**

<table>
<thead>
<tr>
<th>SpeedGrade</th>
<th>( t_{CK} )</th>
<th>( t_{DVB} )</th>
<th>( t_{DVA} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDR3-800</td>
<td>2500 ps</td>
<td>494 ps</td>
<td>494 ps</td>
</tr>
</tbody>
</table>

\[
-0.5 \times t_{CK} + t_{DVA} = -1250 + 494 = -756 \text{ ps} \\
\]

\[ t_{DVB} = -494 \text{ ps} \]

4.3.5 Deriving \( t_{DS} \) and \( t_{DH} \) for Chipset Memory Controller

Refer to Figure 32 for the derivations in Table 20.

**Table 20: Deriving \( t_{DS} \) and \( t_{DH} \) for Chipset Memory Controller**

<table>
<thead>
<tr>
<th>SpeedGrade</th>
<th>( t_{CK} )</th>
<th>( t_{HDctlr} )</th>
<th>( t_{SUctlr} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDR3-800</td>
<td>2500 ps</td>
<td>500 ps</td>
<td>-200 ps</td>
</tr>
</tbody>
</table>

\[
t_{DS} = t_{SUctlr} = -200 \text{ ps} \]

\[ t_{DH} = t_{HDctlr} = 500 \text{ ps} \]
4.3.6 Summary of DDRx Wizard Needed Parameters for Chipset Memory Controller

Table 21: Summary of DDRx Wizard Required Timing Parameters for Chipset Memory Controller

<table>
<thead>
<tr>
<th>DDRx Wizard Required Parameters</th>
<th>Min (ps)</th>
<th>Max (ps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>tCKAC</td>
<td>-3756</td>
<td>-3302</td>
</tr>
<tr>
<td>tCKCTL</td>
<td>-1256</td>
<td>-802</td>
</tr>
<tr>
<td>tCKDQS</td>
<td>-300</td>
<td>74</td>
</tr>
<tr>
<td>tDQS/DQ</td>
<td>-756</td>
<td>-494</td>
</tr>
<tr>
<td>tDS</td>
<td>-200</td>
<td></td>
</tr>
<tr>
<td>tDH</td>
<td>500</td>
<td></td>
</tr>
</tbody>
</table>

5 Creating the DDRx Timing Models (.v)

The required timing parameters derived from the previous section now needs to be implemented into the HyperLynx Timing Model (HLTM) format that is recognized by the DDRx Wizard. This can be done using the following methodologies:

- Using the HyperLynx Timing Model Wizard
  - The parameters entered into the Wizard must be the derived DDRx Wizard required parameters
  - The wizard just creates the most basic model with just one speed grade parameters
  - No syntax expertise required
  - Easy to create and the model is ready for use

- Using the HyperLynx Timing Model Wizard and adding simple syntax in the HyperLynx Timing Model Editor
  - The parameters entered into the Wizard must be the derived DDRx Wizard required parameters
  - Combine different speed-grade timing models created by the Timing Model Wizard to create a multi-speed-grade model
  - Some syntax expertise is required but the Verilog syntax used in these timing models is not that complicated
  - More effort is involved than the previous method but it is a also more robust model

- Entering the required syntax in the HyperLynx Timing Model Editor
  - Define the timing parameters in the model (.v file) as defined in the controller spreadsheet
  - Enter the syntax to derive the DDRx Wizard required timing parameters from the spreadsheet parameters
  - More syntax expertise is required but the Verilog syntax used in these timing models is not that complicated
  - Creates the most robust model and all timing derivations are self-contained for future understanding of the model and reference

5.1 Creating the Single Speed-grade Model

Once you have the required DDRx Wizard timing parameters derived, entering the parameters into the Wizard is quite simple. In this example, the chipset controller parameters from Table 21 are used for the creation process. Note that even though this is the simplest model to create, it doesn’t mean that it’s
less accurate. It simply means that it is the easiest and quickest to create with just one speed grade parameters.

Open the HyperLynx Timing Model Wizard from BoardSim by choosing Models > Run DDRx Controller Timing Model Wizard.

5.1.1 HyperLynx Timing Model Wizard: Introduction page

Set the data rate for the parameters that were derived

5.1.2 HyperLynx Timing Model Wizard: Address/Command Timing

Enter tCKAC (min/earliest) and tCKAC(max/latest) values. These values are entered as positive numbers here.
5.1.3 HyperLynx Timing Model Wizard: Control Timing page

Enter tCKCTL (min/earliest) and tCKCTL (max/latest) values. These values should be entered as positive numbers.

5.1.4 HyperLynx Timing Model Wizard: Write Strobe Timing page

Enter tCKDQS (min/earliest) and tCKDQS (max/latest) values. The earliest value is a negative number and latest value is a positive number.
5.1.5 HyperLynx Timing Model Wizard: Write Data Timing page

Once the model is saved, the .v file with the syntax shown in Figure 33 is created by the Timing Model Wizard. Please refer to the BoardSim Help Manual in Concepts and Reference Guide > File Specifications > HyperLynx Timing Model Format section for more information on the HyperLynx timing model format and syntax. The created model can be opened in the HyperLynx Timing Model Editor to view the timing diagram of the parameters and it is shown in Figure 34. The HyperLynx Timing Model Editor can be opened from BoardSim by choosing Models > Edit DDRx Timing Models (.v).
Note that tCKAC(min)/(max), tCKCTL(min)/(max), and tDQSDQ(min)/(max) values were entered as positive numbers in the DDRx Timing Model Wizard but in Figure 33 and Figure 34, you will notice that those values are displayed as negative numbers. Be aware of this number convention: the DDRx Timing Model Wizard needs these numbers as positive numbers and the DDRx Wizard needs these numbers as negative numbers.
5.2 Creating Multi-Speed-grade Timing Model

This section describes the methodology to create a multi speed-grade model by combining the single speed-grade models created by the Timing Model Wizard from the previous section and adding additional simple syntax to the model.

Figure 35: Syntax for Simple Multi Speed-grade Chipset Controller Model

```plaintext
real Denote_TDH = 0.0; // No hold derating

`; Timing relationships
`*********************************************************************

specify
`ifdef DDR2_667
 // ALU/CBD prelaunch window from next CK (1T or 2T)
 $delay(posedge clk, addr_cmd, -450.0, -450.0);
 // CL prelaunch window from next CK (1T always)
 $delay(posedge clk, ct1, -1500.0, -1652.0);

 // DRAM Write cycles
 $delay(ck, dqa, -33.0, 77.0);
 $delay(posedge dqs, dq, -88.0, -88.0);
 $delay(posedge dqs, da, -88.0, -88.0);

 // DRAM Read cycles
 $setupslop(dqs, dq, -209.0, 500.0);

`elsif DDR3_800
 // ALU/CBD prelaunch window from next CK (1T or 2T)
 $delay(posedge clk, addr_cmd, -776.0, -920.0);
 // CL prelaunch window from next CK (1T always)
 $delay(posedge clk, ct1, -1256.0, -1082.0);

 // DRAM Write cycles
 $delay(ck, dqa, -209.0, 77.0);
 $delay(posedge dqs, dq, -766.0, -494.0);
 $delay(posedge dqs, da, -756.0, -494.0);

 // DRAM Read cycles
 $setupslop(dqs, dq, -209.0, 500.0);

 else `define DDR3_1066
 // ALU/CBD prelaunch window from next CK (1T or 2T)
 $delay(posedge clk, addr_cmd, -2010.0, -2566.0);
 // CL prelaunch window from next CK (1T always)
 $delay(posedge clk, ct1, -943.0, -498.0);

 // DRAM Write cycles
 $delay(ck, dqa, -323.5, 75.5);
 $delay(posedge dqs, dq, -599.6, -139.0);
 $delay(posedge dqs, da, -599.6, -139.0);

 // DRAM Read cycles
 $setupslop(dqs, dq, -209.0, 500.0);

endif

endspecify
endmodule
```

Figure 35 shows the required syntax for that multi-grade model and the methodology to create this model is as follows:

1. Create a model for each speed-grade using the Timing Model Wizard as described in the previous section. In this example, there are three speed grades model files with the names: DDR2_667.v, DDR3_800.v, and DDR3_1066.v.
2. Open the first speed-grade model (DDR2_667.v) in the HyperLynx Timing Model Editor.
3. Save this file as DDRx_multigrade.v.
4. In the line after specify, enter `ifdef DDR2_667
5. At the end of `ifdef section, enter `elsif DDR3_800
6. Open the second speed-grade model (DDR3_800.v), and copy all the lines that are between specify and endspecify.
7. Paste it in the DDRx_multigrade.v file after the line `elsif DDR3_800.
8. At the end of `elsif section enter `else `define DDR3_1066.
9. Open the third speed-grade model (DDR3_1066.v), and copy all the lines that are between specify and endspecify.
10. Paste it in the DDRx_multigrade.v file after the line `else `define DDR3_1066.
11. At the end of `else `define section, type in `endif

12. Choose Build > Compile to check for any syntax errors.


5.3 Creating a Controller Timing Model Syntactically

In this section, the same chipset controller model is created by entering the syntax directly in the HyperLynx Timing model Editor. Since some of the syntax is common to all models, a default controller model can be used as a starting point for this model creation methodology.

4.3.1 Define Top-level Module and HyperLynx Standard Port Names

The top level module and standard HyperLynx port names are defined in this section. The module name can be arbitrary but the port names are reserved for HyperLynx as defined in the embedded table in Figure 36. This section is required in the HyperLynx timing model.

![Figure 36: Syntax for Defining Top-level Module and HyperLynx Standard Port Names](image)

5.3.2 Specify Chipset Controller Datasheet Parameters by Speed-grade

In this section, the parameters of different speed-grades are entered exactly as defined in the datasheet. The speed designators must follow the convention as described in the embedded table (from Help) Figure 37.

![Figure 37: Syntax for Specifying Chipset Controller Datasheet Parameters by Speed-grade](image)
5.3.3 Derive HyperLynx Required Timing Parameters

Table 22 shows the equations needed to derive the HyperLynx required timing parameters from the chipset controller timing parameters. The left column shows the generic timing equations that were used in section 4.3 Example 3: Chipset Memory Controller. The right column show the syntax required in HyperLynx timing model format. Note that tCKAC(min)/(max), tCKCTL(min)/(max), and tDQSDQ(min)/(max) values are defined syntactically as negative numbers.

<table>
<thead>
<tr>
<th>Generic Timing Equations</th>
<th>HyperLynx Timing Model Syntax (HLTM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>From Table 16:</td>
<td></td>
</tr>
<tr>
<td>tCKAC(min) = -2*tCK + tCMDva</td>
<td>parameter tCKACMin = -(2*tCK) + tCMD_va;</td>
</tr>
<tr>
<td>tCKAC(max) = -tCMDvb</td>
<td>tCKACMax = -tCMD_vb;</td>
</tr>
<tr>
<td>From Table 17</td>
<td></td>
</tr>
<tr>
<td>tCKCTL(min) = -tCK - tCNTL_va</td>
<td>parameter tCKCTLMin = tCNTL_va - tCK;</td>
</tr>
<tr>
<td>tCKCTL(max) = -tCK - tDSSctlr</td>
<td>parameter tCKCTLMax = -tCNTL_vb;</td>
</tr>
<tr>
<td>From Table 18</td>
<td></td>
</tr>
<tr>
<td>tCKDQS(min) = tDSHctlr - 0.5*tCK</td>
<td>parameter tCKDQSMin = tDSHctlr - tDQBit;</td>
</tr>
<tr>
<td>tCKDQS(max) = 0.5*tCK - tDSSctlr</td>
<td>parameter tCKDQSmax = tDQBit - tDSSctlr;</td>
</tr>
<tr>
<td>From Table 19</td>
<td></td>
</tr>
<tr>
<td>tDQSDQ(min) = -0.5*tCK + tDVA</td>
<td>parameter tDQSDQMin = -tDQBit + tDVA;</td>
</tr>
<tr>
<td>tDQSDQ(max) = -tDVB</td>
<td>parameter tDQSDQmax = -tDVA;</td>
</tr>
<tr>
<td>From Table 20</td>
<td></td>
</tr>
<tr>
<td>tDS = tSUctrl</td>
<td>parameter tDS = tSUctrl;</td>
</tr>
<tr>
<td>tDH = tHDctrlr</td>
<td>parameter tHD = tHDctrlr;</td>
</tr>
</tbody>
</table>

Figure 38: HyperLynx Timing Model Syntax to Derive the Required HyperLynx Timing Parameters

```verbatim
// HyperLynx Required Timing Parameters Derivations

// ADDR/CMD Parameter Derivations
parameter tCKACMin = -(2*tCK) + tCMD_va; // CK rising out to earliest valid ADDR/CMD (CE-2)
parameter tCKACMax = -tCMD_vb; // CK rising out to latest valid ADDR/CMD (CE-1)

// CTRL Parameter Derivations
parameter tCKCTLMin = tCNTL_va - tCK; // CK rising out to earliest valid CTRL (CK=1)
parameter tCKCTLMax = -tCNTL_vb; // CK rising out to latest valid CTRL (CK=1)

// DRAM Write Parameter Derivations
parameter tDSHctlr = tCK / 2.; // Data bit time
parameter tDQBSmin = -tDQBit + tDVA;
parameter tDQBSmax = -tDVA;
parameter tDSHctrl = tDQBit - tDSSctlr;
parameter tDSHctrl = tDVA;
parameter tDDctrl = tDSSctlr;

// DRAM READ Parameter Derivations
parameter tDS = tDDctrl;
parameter tDH = tDDctrl;

<-----------------------------Internal DQS shift (read cycles)----------------------------->
real DQSReadShift = 0;
```

5.3.4 Define Timing Relationships in the Model’s Specify Block

In this section, the timing relationships are defined. This section is responsible for providing the min and max delay values, and setup and hold values between ports to the DDRx Wizard. All controller timing models must include this section and Figure 39 shows the required syntax for each parameter.

The $delay function is a non-Verilog construct added to the HyperLynx timing model syntax. The following is the syntax to specify delay relationship between two module ports:

```verbatim
$delay(fromport_event, toport_event, min_delay, max_delay);
```

For example, in Figure 39, the syntax $delay(posedge ck, addr_cmd, tCKACMin, tCKACMax); sends the min and max delay values (tCKACMin and tCKACMax) between the ck and addr_cmd ports to the DDRx Wizard. Remember that the values tCKACMin and tCKACMax were defined in
the previous section. The parameter names are arbitrary but it is recommended that the standard parameter names (as required by the DDRx Wizard) are used for purpose of clarity.

Note that parameter tDQDQS and parameters tDS/tDH are complementary methods of describing the DQ-to-DQS timings during read cycles; only one of the two methods should be specified in the timing model file. Please refer to HyperLynx Help for more information.

Figure 39: Syntax to Define Timing Relationships

```
! Timing relationships
*******************************************************************************
 specify
 // ADDR<CH> prelaunch window from next CK (27)
 $delay(posedge ck, addr_ch, tPRECHMin, tPRECHMax); 
 // CTK prelaunch window from next CK (1F clocks)
 $delay(posedge ck, tctl, tCRTLMin, tCRTLMax); 
 // DQM Write cycles
 $delay(dq, dq, tCQDSMin, tCQDSMax);
 $delay(dq, dqs, tDQDSMin, tDQDSMax);
 // DQM Read cycles
 $setup(dqs, dq, tDQS, tDQSDelay);
 endspecify
 endmodule
```

This chipset controller timing model can be downloaded from the link in the following tech note:


6 Summary

This application note provided necessary information needed to create a controller HyperLynx timing model from a vendor’s datasheet. Three examples of different controller types were used to describe the model creation process. In order to create these models, it is important that the datasheet parameters are translated into the HyperLynx required parameters correctly. Once the parameters are translated correctly, the model can be created using the HyperLynx Timing Model Wizard (GUI-based) or using the HyperLynx Timing Editor (Syntax-based). A less robust controller timing model can be created quickly and easily using the Timing model wizard or a more robust model can be created syntactically.