Timing challenges for serial flash interface


In absence of any standards for serial flash memory, timing requirements are different for each vendor. Timing closure across PVT corners with shrinking technology nodes is a major challenge. Below, we talk about static timing requirements and timing closure challenges without any “data learning” that has to be met for correct operation of the controller with any external serial NOR flash.

The system for any external serial NOR flash consists of a flash controller, external serial flash, and the interfacing path between the two. Figure 1 shows the elements that form a part of the timing analysis for any host and external flash combination. The data forward path (Controller -> Pads -> Board -> Flash) need to take into account the propagation delay from the launching flop to the pad, board delays and the input setup time of the flash. All these have to be checked relative to the flash clock. Since the clock and data are moving in the same direction, it is relatively easier to meet timing in the forward data path. The data return path (Flash-> Board-> Pads -> Controller) would need to take into account flash clock to data output delay, board delay, pad delay, and the propagation delay from the pad to the capturing flop. The return path has the clock and data moving in the opposite direction thereby making the timing budgeting difficult.

![Figure 1: Topology](image)

Timing Modes

The flashes today support various timing modes for example SDR, DDR, DQS. All these modes have their own timing requirements. The Table. 1 below shows the timing requirements from various flash vendors in different modes [1] [2][3][4].
Table 1: Timing requirements by various flashes

<table>
<thead>
<tr>
<th>Flash Vendor (Part number)</th>
<th>SDR</th>
<th>DDR</th>
<th>DQS</th>
<th>Max Frequency supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spansion “S25FL128S and S25FL256S”</td>
<td>Data In setup- 2ns, Data In hold- 2ns, Clock low to Output data valid- 6.5ns (15pF), Output hold- 2ns</td>
<td>Data In setup-1.5ns, Data In hold-1.5ns, Clock low to Output data valid-6.5ns (15pF), Output hold-1.5ns</td>
<td>NA</td>
<td>SDR- 133MHz, DDR- 80MHz</td>
</tr>
<tr>
<td>Macronix “MX25L6465e/ MX25L12865e”</td>
<td>Data In setup- 2ns, Data In hold- 5ns, Clock low to Output data valid-12ns (30pF), Output hold- 2ns</td>
<td>Data In setup-2ns, Data In hold- 5ns, Clock low to Output data valid-9.5ns, Output hold- 2ns</td>
<td>NA</td>
<td>SDR- 104MHz, DDR- 50MHz</td>
</tr>
<tr>
<td>Winbond “W25Q256FV”</td>
<td>Data In setup- 2ns, Data In hold- 3ns, Clock low to Output data valid-7ns, Output hold- 2ns</td>
<td>Data In setup-2ns, Data In hold- 3ns, Clock low to Output data valid-7ns, Output hold- 2ns</td>
<td>NA</td>
<td>SDR- 104MHz, DDR- 50MHz</td>
</tr>
<tr>
<td>Micron “N25Q128A”</td>
<td>Data In setup- 2ns, Data In hold- 3ns, Clock low to Output data valid-5ns (10pF), Output hold- 1ns</td>
<td>Data In setup-2ns, Data In hold- 3ns, Clock low to Output data valid-5ns, Output hold- 1ns</td>
<td>NA</td>
<td>SDR- 108MHz, DDR- 54 MHz</td>
</tr>
<tr>
<td>Spansion’s Hyperflash</td>
<td>NA</td>
<td>TBD</td>
<td>TBD</td>
<td>DDR-166MHz</td>
</tr>
</tbody>
</table>

The Table 2 below summarizes the terms used in the paper further:

Table 2: Terms and their meaning

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_{DVO} )</td>
<td>Clock to data valid delay</td>
</tr>
<tr>
<td>( t_{DO, \text{Board}} )</td>
<td>Board delay in data output path</td>
</tr>
<tr>
<td>( t_{DI, \text{Board}} )</td>
<td>Board delay in data input path</td>
</tr>
<tr>
<td>( t_{SU, \text{SDR}} )</td>
<td>Flash input setup time in SDR mode</td>
</tr>
<tr>
<td>( t_{SU, \text{DDR}} )</td>
<td>Flash input setup time in DDR mode</td>
</tr>
<tr>
<td>( t_{DH, \text{SDR}} )</td>
<td>Flash input data hold time in SDR mode</td>
</tr>
</tbody>
</table>
### SDR Timing

Most of the flashes operate in single data rate (SDR) mode. In SDR mode, the data is transferred only on one edge of the clock signal. The SDR serial flashes sample the incoming data on the rising edge of flash clock and drive the output data on the falling edge of the flash clock. The clock is usually provided by the host controller. Figure 2 shows the delays to be considered for closing the timing in SDR mode.

![SDR Timing](image)

**Figure 2: SDR Timing requirement**

*Describing the terms used in Figure 2-4*

SCK_q is the clock from the controller’s end, SCK_f is the clock when it is received at the flash’s end. The controller drives the data DO_q to the flash, after the propagation delay and the pad delay, the data at the flash interface is DO_f. Flash samples this data and drives data to the controller, DI_q. After the propagation delays, board and pad delays, the data DI_f reaches the controller.

**Timing parameters to be considered in SDR mode for flash input timing taken from reference edge A**-

1. \( t_1 \) (Maximum data delay from the controller to the flash) = \( t_{DVO} + t_{DO,\ Board} \)

2. \( t_2 \) (Time at which data is required at flash interface) = \( \frac{1}{2} \) clock period + clock skew - \( t_{SU,SDR} \)

3. \( t_3 \) (Maximum time for which the data will remain valid at flash) = \( t_1 \) + one clock period

4. \( t_4 \) (Hold requirement of the flash as seen from ref. edge A) = Clock skew + \( \frac{1}{2} \) clock period + \( t_{DH,SDR} \)
Timing parameters to be considered in SDR mode for flash output timing taken from reference edge C’ and capture edge D-

\( t_5 \) (Total data delay for the data released by flash to be captured at the controller interface) = \( t_v + t_{DL, board} \)

\( t_6 \) (Time at which the data is required at controller’s interface) = \( \frac{t_{SC}}{2} - \text{clock skew} - t_{ctrl, setup} \)

\( t_7 \) (Maximum time for which the flash data at the controller will remain valid) = \( \frac{t_{SC}}{2} - \text{clock skew} + t_{HO} \)

\( t_8 \) (Hold requirement of the controller as seen from ref. edge C’) = \( \frac{t_{SC}}{2} - \text{clock skew} + t_{ctrl, hold} \)

For timing closure in SDR mode the following equations should be satisfied,

\[ t_2 > t_1, \text{ this will ensure that the setup requirement of the flash to capture data is met} \]

\[ t_3 > t_4, \text{ this will ensure that the hold requirement of the flash to capture data is met} \]

\[ t_6 > t_5, \text{ this will ensure that the setup requirement of the controller to capture data is met} \]

\[ t_7 > t_8, \text{ this will ensure that the hold requirement of the controller to capture data is met} \]

**DDR Timing**

The increasing requirement of improved throughput has introduced the double data rate (DDR) mode. In DDR mode, the data is transferred on both the rising and falling edges of the clock signal. The DDR serial flashes sample as well as drive the data on both rising and falling edges of flash clock. The path is therefore of only half clock cycle, as a result of which meeting timing in high frequency DDR mode becomes a challenge.

As the maximum data valid time \( (t_v) \) approaches half clock period, closing the static timing analysis becomes a nightmare since most flashes don’t provide a decent output hold time \( (t_{HO}) \). As a result the valid data window for timing closure becomes very small as compared to SDR mode. In absence of any sort of data learning it becomes very difficult to assure timing at maximum frequency for DDR mode across PVT corners. Figure.3 shows the delays to be considered for closing the timing in DDR mode.
The static timing requirements for DDR mode is governed by the equations below:

**Timing parameters to be considered in DDR mode for flash input timing taken from reference edge A:**

- \( t_1 \) (Maximum data delay from the controller to the flash) = \( t_{DO} + t_{DO,\text{Board}} \)
- \( t_2 \) (Time at which data is required at flash interface) = \( \frac{1}{2} \) clock period + clock skew - \( t_{SU,\text{DDR}} \)
- \( t_3 \) (Maximum time for which the data will remain valid at flash) = \( t_1 + \frac{1}{2} \) clock period
- \( t_4 \) (Hold requirement of the flash as seen from ref. edge A) = Clock skew + \( \frac{1}{2} \) clock period + \( t_{DH,\text{DDR}} \)

**Timing parameters to be considered in DDR mode for flash output timing taken from reference edge C’:**

- \( t_5 \) (Total data delay for the data released by flash to be captured at the controller interface) = \( t_{V} + t_{DI,\text{Board}} \)
- \( t_6 \) (Time at which the data is required at controller’s interface) = \( t_{SC}/2 - \text{clock skew} - t_{\text{ctrl,setup}} \)
- \( t_7 \) (Maximum time for which the flash data at the controller will remain valid) = \( t_{SC}/2 - \text{clock skew} + t_{HO} \)
- \( t_8 \) (Hold requirement of the controller as seen from ref. edge C’) = \( t_{SC}/2 - \text{clock skew} + t_{\text{ctrl,hold}} \)

For timing closure in DDR mode the following equations should be satisfied,

- \( t_2 > t_1 \), this will ensure that the setup requirement of the flash to capture data is met
- \( t_3 > t_4 \), this will ensure that the hold requirement of the flash to capture data is met
$t_6 > t_5$, this will ensure that the setup requirement of the controller to capture data is met

$t_7 > t_6$, this will ensure that the hold requirement of the controller to capture data is met

### DQS Timing

Recently, Spansion introduced its Hyperflash into the market which supports a data strobe signal (RDS) similar to the DQS in DRAM interface [5]. This signal eases the timing closure in the DDR mode by allowing the timing of the input data w.r.t to the RDS/DQS signal provided by the flash instead of the conventional clock out data in timing.

In a flash supporting DQS mode, the data strobe signal is an output from the flash device that indicates when data is being transferred from the flash to the host. The data is then captured by the controller on both rising/falling edge of the DQS signal. This data strobe signal is used by the flash only for read and not write. For write, the timing requirement has to be met similar as in the conventional DDR mode.

Here both DQS and the data are sent by the flash, both of these being in the same direction it is easier to just match the skew between the two (in case DQS is centre aligned with the data) for timing closure. In case DQS is not center aligned with the data, the DQS has to be delayed to match the skews for proper timing closure in all PVT corners. Figure 4 shows the delays to be considered for closing the timing in DQS mode.

![Figure 4: DQS timing requirement](image)

**Timing parameters to be considered in DQS mode for flash output timing taken from reference edge D’-**

$t_6$ (Total data delay for the data released by flash to be captured at the controller interface) = $t_v + t_{DL, Board}$

$t_6$ (Time at which the data is required at controller’s interface) = Total clock DQS delay - $t_{ctrl, setup}$

$t_7$ (Maximum time for which the flash data at the controller will remain valid) = $t_{SC}/2$ - clock skew +
\[ t_{\text{ho}} \]
\[ t_6 (\text{Hold requirement of the controller as seen from ref. edge D'}) = \text{Total clock DQS delay} + t_{\text{ctrl,hold}} \]

For read timing closure in DQS mode the following equations should be satisfied,

\[ t_6 > t_5, \text{ this will ensure that the setup requirement of the controller to capture data is met} \]
\[ t_7 > t_8, \text{ this will ensure that the hold requirement of the controller to capture data is met} \]

Thus, even with shrinking data valid window in case of high frequency DDR mode, the presence of this data strobe signal has eased the timing for reads in DDR mode.

**Conclusion**

This paper not only extensively covers the timing challenges in every mode (SDR, DDR, DQS) of the external serial nor flashes, but also talks about the specific timing requirements to be used in closing the static timing in all these modes independent of flash vendor. Meeting timing requirements over various PVT corners is difficult, as the variation from best to worst corners is significant. Also, with increasing use of DDR mode for high performance and shrinking data valid window, “data learning” and “dqs mode” are becoming important. The inclusion of such techniques will ease the timing closure for the host controller, making the device highly efficient for high performance use-cases, giving maximum throughput.

- **References**


