



White Paper

**Choosing the best pin multiplexing method
for your Multiple-FPGA partition**

S2C Inc.
1735 Technology Drive, Suite 620
San Jose, CA 95110, USA
Tel: +1 408 213 8818
Fax: +1 408 549 9948
www.s2cinc.com

Choosing the best pin multiplexing method for your Multiple-FPGA partition

Introduction

Using multiple FPGAs to prototype a large design requires solving a classic problem: the number of signals that must pass between devices is greater than the number of I/Os pins on an FPGA. The classic solution is to use a TDM (Time Domain Multiplexing) scheme that muxes two or more signals over a single wire or pin.

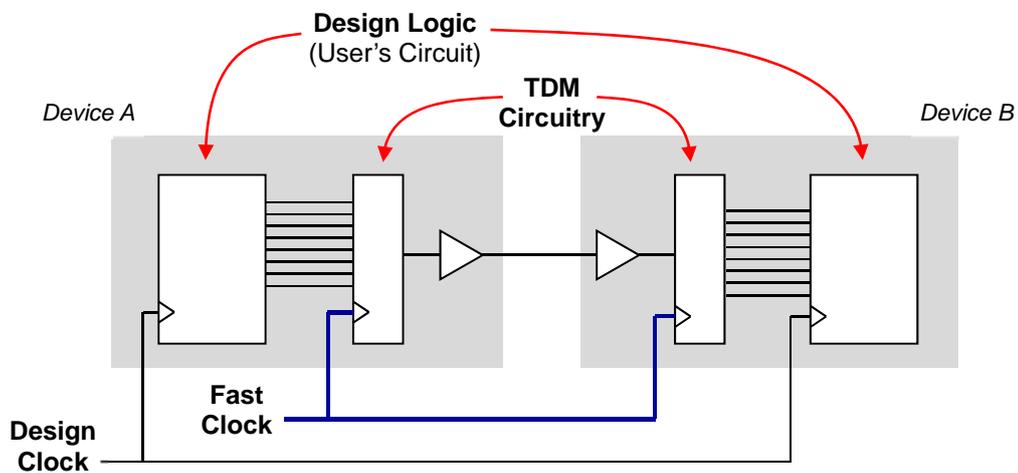


Figure 1 Signals Multiplexed with a Fast Clock

This solution is still widely employed, and coupled with the advances in FPGAs, the obstacles to constructing a multi-device prototype are greatly reduced. The latest FPGAs offer advantages such as a very high number of industry-standard I/O, integrated high-speed transceivers, and LVDS signaling.

Flavors of TDM

There are two distinct types of TDM implementations: synchronous and asynchronous. In **synchronous** TDM the multiplexing circuitry is driven by a fast clock that is synchronous with the (user's) design clock.

Synchronous mode is sufficient for many TDM implementations, but there are limitations. There must be no feed-through nets between FPGAs before inserting TDM (signals that pass through an FPGA without terminating at a register).

In addition, the difference between the fast clock and the design clock can introduce issues. The timing diagram below shows an example of this where event A is the sampling time for

the fast clock, and event B is the sampling time for the design clock – the setup time for both needs to be the same as a single period of the fast clock.

And the interface between the two clock domains could contain a critical path, especially when the TDM ratio is quite large. (This is true even where all inter-FPGA nets are registered input/output). This path is often routed poorly inside the FPGA, and usually suffers from timing violations due to limited FPGA routing resources. This in turn significantly decreases the speed of the fast clock, which decreases the speed of the design.

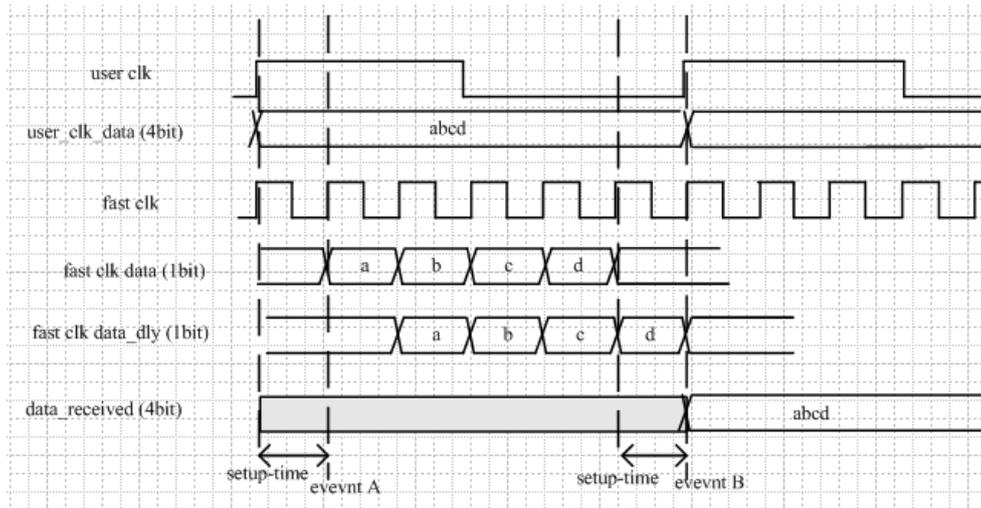


Figure 2 Synchronous TDM Timing

Finally, synchronous TDM typically supports only one clock per one set of pins. Usually this requires stricter timing constraints which can be hard to meet when there are a lot of pins, and therefore is difficult to automate.

In **asynchronous** mode, the TDM fast clock runs completely independent of the design clocks. Although asynchronous mode is slower, it supports multiple clocks and the timing constraints are easier to meet.

Asynchronous TDM addresses the timing violations caused by synchronous mode, and does not require a timing constraint on the datapath between clock domains (usually equal to one-cycle of the fast clock.) In fact, the fast clock can always run at its maximum speed. (For LVDS TDM, this is 1 Gbps for Virtex 7 and 1.6 Gbps for Virtex UltraScale.) This means the design clock speed won't be affected by potential a reduction of the fast clock, as in synchronous mode.

An additional benefit is that asynchronous TDM is not sensitive to feed-through nets, so these can be used with an asynchronous scheme. However, the designer should be aware that feed-through nets transmitted over asynchronous TDM can impact system performance.

Single-cycle and Multi-cycle clocks

The majority of designs utilized a single-cycle clock. The bottleneck for pin multiplexing frequency becomes the latency, rather than of how fast signals can be transmitted between devices. Since LVDS has a longer latency, LVDS can actually be slower than single-ended

signals when the TDM pin ratio is low. However, when the TDM pin ratio is high, the LVDS latency becomes less of a factor and therefore runs faster than single-ended signals.

As for designs that use multiple clock cycles, they can run at full transmission speed. However since the data doesn't get to the destination in 1 design clock cycle, the designer must manually insure this is okay for their design. This issue is design dependent, and as result, can't be automated.

Single-ended Signals vs. LVDS

Single-ended TDM uses a single-ended signal which can transmit physical signals at a speed up to 290 MHz (Virtex UltraScale). This is determined by dividing the TDM ratio (or signal multiplexing ratio) and taking into account setup, synchronization and board delays.

With a TDM ratio of 4:1, the system clock speed will be around 17.8 MHz. If the TDM ratio is increased to 16:1, the system clock speed will drop to less than 10 MHz. From this we can see that as the TDM ratio increases, the performance drop linearly.

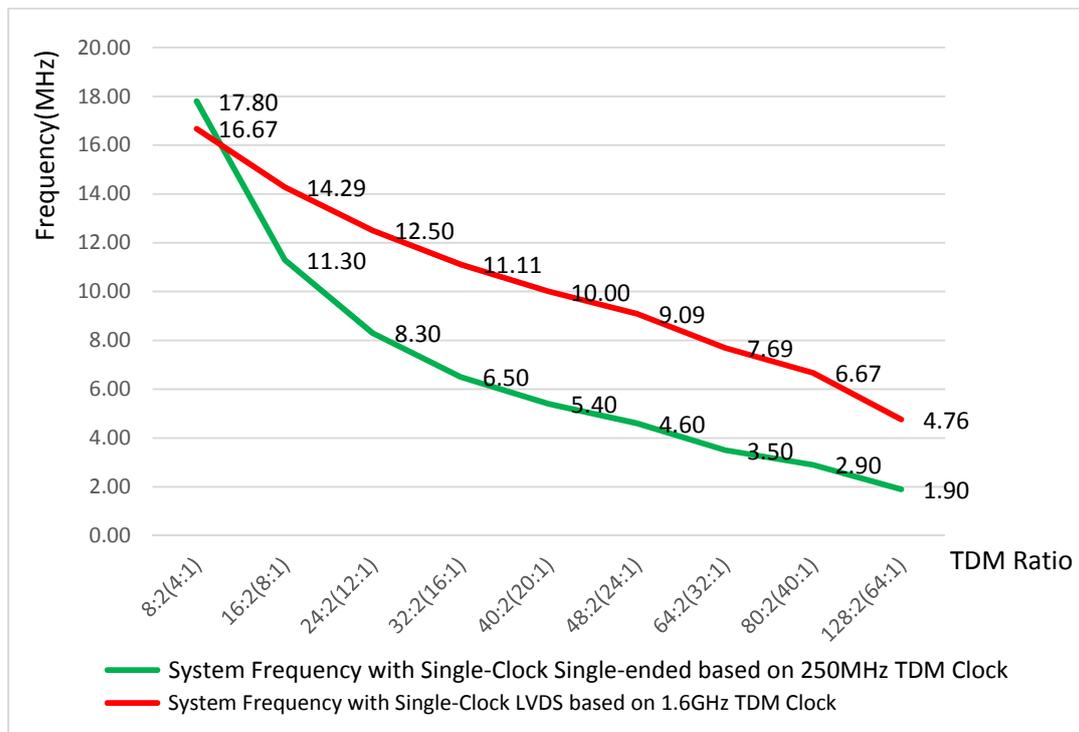


Figure 3 Single-Ended TDM and LVDS TDM performance with Asynchronous mode

However, using the LVDS (Low Voltage Differential Signaling) I/O standard supported by Xilinx FPGAs, the physical transmission data rate between FPGAs can achieve up to 1.6 Gbps. This offers tremendous advantages over single-ended transmission, even when considering that a single LVDS signal requires a pair of single-ended pins.

However, using the LVDS (Low Voltage Differential Signaling) I/O standard supported by Xilinx FPGAs, the physical transmission data rate between FPGAs can achieve up to 1.6

Gbps. This offers tremendous advantages over single-ended transmission, even when considering that a single LVDS signal requires a pair of single-ended pins.

shows a comparison between Single-Ended TDM and LVDS TDM using Xilinx UltraScale devices. (Note: performance for different FPGA families vary.) Performance of TDM implemented with LVDS is better than Single-Ended TDM, especially for higher TDM ratios.

The chart below is another comparison of Single-ended TDM and LVDS TDM. It shows the number of physical I/O needed to accommodate a given number of virtual I/O, assuming a system speed of 11 MHz:

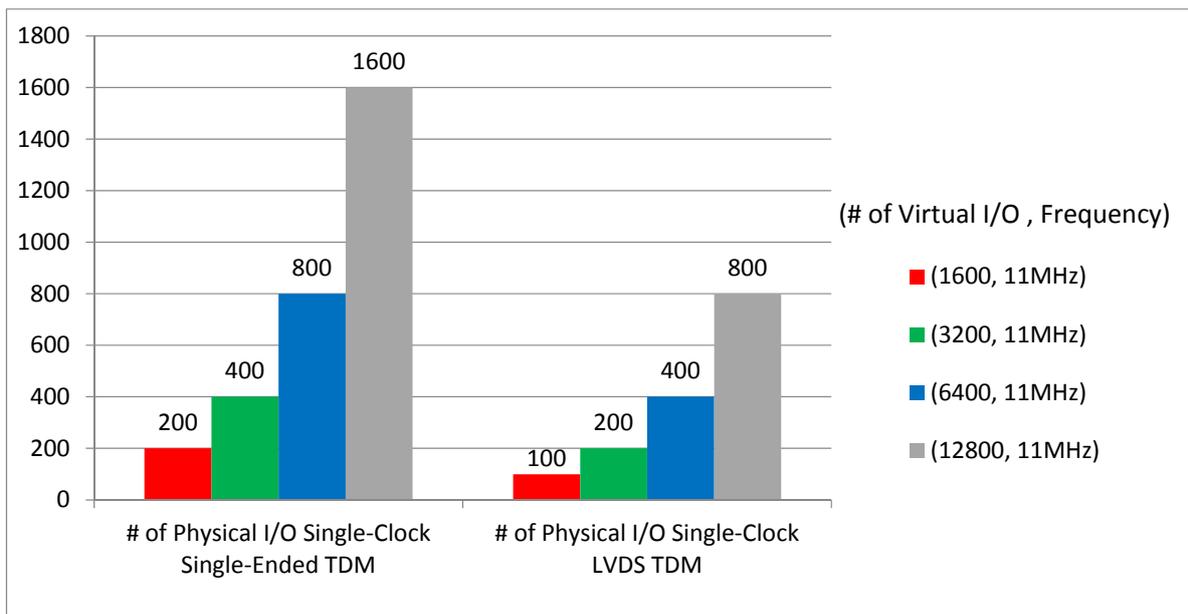


Figure 4 Number of physical interconnections needed for a system running at 11MHz

This shows that for a system with a clock speed of 11 MHz, if 12800 virtual connections are needed, single-ended TDM consumes 1600 physical I/O. With LVDS TDM, this number is cut in half to 800.

Given the physical I/O limitation of FPGAs, partitioning becomes easier if less physical interconnections are needed. LVDS TDM has clear advantages over traditional Single-Ended TDM.

Partitioning and Automatic TDM Insertion

Combining the technique of using asynchronous LVDS TDM with a single clock cycle design, it's possible to create a tool that can partition a design and perform automatic TDM insertion. Ideally, such a tool would be able to:

- Optimizes buses and match the LVDS resources in each bank considering such factors as trace lengths, matching impedances, and impedance continuity.

- Avoid consuming FPGA design resources for the TDM circuitry by taking advantage of built-in reference clocks (e.g.: IODELAY) to drive TDM clocks and resets

S2C's Prodigy Play Pro is a tool that provides design partitioning across multiple FPGAs, and offers automatic TDM insertion based on an asynchronous TDM using LVDS.

Appendix

All TDM speed data is based on S2C's reference design running on a Prodigy UltraScale series Logic Module.

Table 1 Speed Comparison Table

Asynchronous mode				Synchronous mode					
LVDS mode@1.6Gbps		Single-ended mode@250Mbps		LVDS mode@1.6Gbps			Single-ended mode@250Mbps		
Pin ratio	system speed	Pin ratio	system speed	Pin ratio	system speed	Multiple-cycle speed	Pin ratio	system speed	Multiple-cycle speed
8:2	16.7M	4:1	17.8M	8:2	33.3 M	200M	4:1	41.6M	62.5M
16:2	14.3M	8:1	11.3 M	16:2	28.6 M	100M	8:1	25.0M	31.2M
24:2	12.5M	12:1	8.3M	24:2	25.0 M	66.7M	12:1	17.8M	20.8M
32:2	11.1M	16:1	6.5M	32:2	N/A	50.0M	16:1	13.9M	15.6M
40:2	10.0M	20:1	5.4M	40:2	N/A	40.0M	20:1	11.3M	12.5M
48:2	9.1M	24:1	4.6M	48:2	N/A	33.3M	24:1	9.6M	10.4M
64:2	7.7M	32:1	3.5M	64:2	N/A	25.0M	32:1	N/A	7.8M
80:2	6.7M	40:1	2.9M	80:2	N/A	20.0M	40:1	N/A	6.2M
128:2	4.8M	64:1	1.9M	128:2	N/A	12.5M	64:1	N/A	3.9M