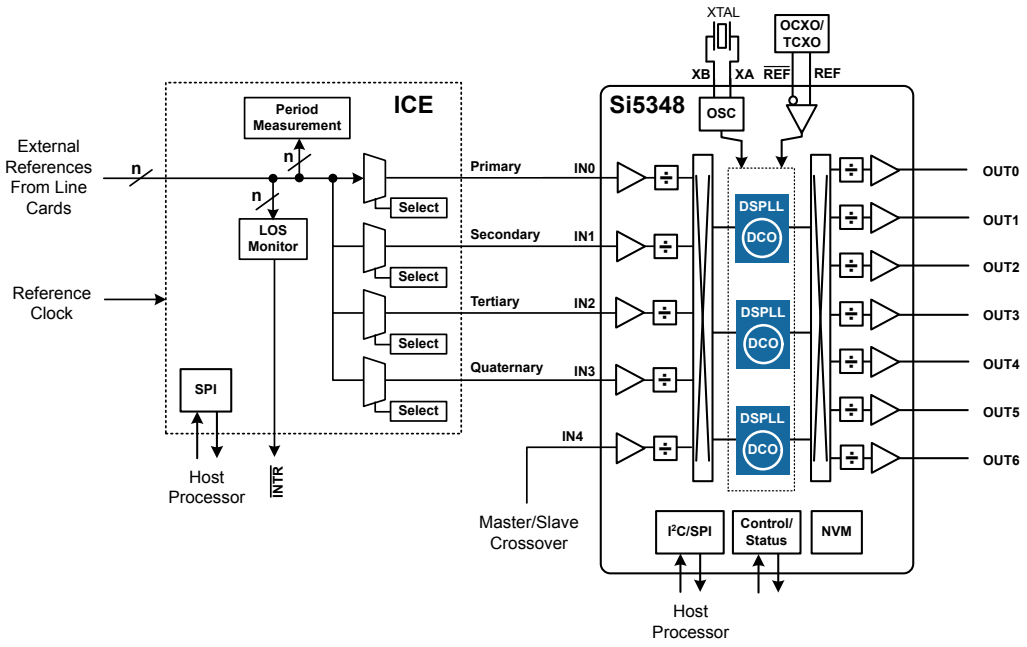
AN1111: DSPLL Input Clock Expander

This application note details an FPGA-based external clock multiplexer that adapts DSPLL devices to applications that require a large number of clock inputs. Because the logic is implemented in an FPGA, the number of selectable inputs and independent multiplexers is bounded only by the available FPGA resources and the register map construction.

A typical master/slave timing configuration found in many telecom products is one application where the Input Clock Expander (ICE) proves useful. In the example shown below, the ICE has been configured to accept *n* clock inputs from system line cards and generates four clock outputs that the Si5348 uses to determine the final source of synchronization. In this case, input selection is accomplished in two stages. The first stage, implemented in the ICE, allows a host processor to select any one of *n* inputs as primary, secondary, tertiary, and quaternary clocks. The host processor uses both loss-of-signal (LOS) and period measurements within the ICE to determine the best inputs to select. The Si5348 provides the second stage of input selection by using an automatic state machine that selects one of the four inputs provided by the ICE. The Si5348 uses its own LOS and out-of-frequency (OOF) monitors, configurable priorities, and revertive/non-revertive properties to automatically select the input used for synchronization. The Si5348 also provides hitless switching between its four inputs. This allows the Si5348 to select from *n* clock inputs while meeting system frequency and phase transients.

KEY FEATURES

- Flexible number of inputs (*n*: 8, 16, 24, or 32) adapt to most applications
- Up to 8 independent output muxes select any one of *n* inputs
- Independent LOS monitor on each input
- Time-sliced period measurements supported on each input with 1 ppb resolution
- Interrupt pin indicating LOS change of status
- 4-wire SPI interface for device control and status monitoring



Input Clock Expander Example

1. Design Overview

The ICE is implemented in a Xilinx Artix-7 FPGA. However, the code provided with this application note could be easily ported to other Xilinx FPGA families and even different FPGA vendors. The only technology-specific features of the code are the clock distribution and the glitchless output muxes. All FPGA vendors have similar primitives for clock distribution, so porting this logic is simply replacing each primitive with its equivalent. The glitchless mux is a little more specialized, but should also be easy to port when using Xilinx devices. When porting to other FPGA vendors, the glitchless mux implementation may require some modifications to align with the capabilities of the selected FPGA vendor. Alternately, the 2:1 glitchless mux used as the final muxing stage could be implemented in FPGA fabric instead of a vendor specific hard macro. The downside to this approach is that a fabric-based implementation is often difficult to properly constrain and the maximum operating frequency may be reduced.

1.1 Inputs

The ICE supports up to thirty-two clock inputs operating at 1 PPS or 8 kHz – 200 MHz. The number of inputs is easily configured by modifying the ‘INPUTS’ parameter located in the top-level module (MUX_TOP). Supported values are: 8, 16, 24, or 32. It’s important to note that 32 is not a hard upper limit and is only bounded by the construction of the device register map. Adding additional pages to the register map would allow more inputs to be supported, limited only by the available FPGA resources. The input signal format is also very flexible and may be set to any input buffer type supported by the FPGA.

1.2 Clock Muxes

The ICE supports up to eight clock output muxes operating at 1 PPS or 8kHz – 200Mhz. Each mux has the ability to select one of the n possible inputs. There is no frequency scaling within the mux logic (i.e. no dividers), so users must ensure clocks with the proper frequency are connected and selected for their application. The number of outputs (i.e. clock muxes) is easily configured by modifying the ‘OUTPUTS’ parameter located in the top-level module (MUX_TOP). Supported values are: 1 – 8. Like the inputs, 8 is not a hard upper limit and is only bounded by the construction of the device register map. Adding additional pages to the register map would allow more outputs to be supported, limited only by the available FPGA resources. The output signal format is also very flexible and may be set to any output buffer type supported by the FPGA.

Two types of muxes are supported in the ICE design: standard and glitchless. Both muxes generate glitchless clock outputs during static selection. The difference between the two implementations is evident only when the clock selection changes. With the standard mux, glitches and/or runt pulses may be generated when the select line changes. This is avoided in the glitchless mux by ensuring all changes occur when the clock lines are low. Specifically, the “from” clock is disabled and the “to” clock is enabled on falling edges. This may result in an extended low period during the switch, but it avoids the generation of glitches and/or runt pulses. Selection of the mux type occurs at build time with compiler directives. If standard muxes are desired, the user builds the code with STANDARD_MUX defined. If glitchless muxes are desired, the user builds the code with GLITCHLESS_MUX defined.

The choice of mux type is application dependent although the use of glitchless muxes is recommended. When using standard muxes, the user should never switch an active clock (i.e. the clock the DSPLL has selected). Based on the description above, one may wonder why the glitchless type is not always used. Well, there is some overhead associated with the glitchless mux implementation. Specifically, it requires more logic (including clock resource), takes longer to complete switches, and may require more host processor involvement.

1.3 Input Selection

Input selection for each clock mux is accomplished through the serial interface by writing to a register. There is an input selection register for each of the output muxes. When using the standard mux, the switch occurs immediately when the register is changed. When using the glitchless mux, a state machine must cycle through several states before the switch completes. In this case, the status of the switch can be monitored by reading the associated busy bit. A '1' indicates that the switch is in progress, and '0' indicates the switch is complete. When complete, the host knows that the input selection register reflects the selected output clock. The time to complete a glitchless switch depends on the input clock mode. Switching between standard inputs clocks will take ~575 µs for the busy bit to clear. Switching between 1 PPS input clocks will take ~4.5 s for the busy bit to clear. In both cases, the output clock may reflect the new input prior to the busy bit clearing.

It is important to note that switching between inputs will not be hitless. In other words, any phase difference between input clocks will be passed on to the outputs. The intent of the ICE is to statically select up to eight out of the *n* inputs. Once this section has been performed, the DSPLL is responsible for protection switching between its inputs.

1.4 Loss-of-Signal (LOS) Monitor

Two types of LOS monitors are supported in the ICE design: standard and alternate. The standard LOS monitor is functionally equivalent to the LOS detector in Skyworks' Si534x/8x devices. The alternate LOS monitor is an interval based detector. As a result, it has less precision than the standard monitor, but also requires much less logic. It is well suited for applications where FPGA resources are tight and gross LOS checks are acceptable. Selection of the LOS monitor type occurs at build time with compiler directives. If Si534x/8x-like detectors are desired, the user builds the code with LOS_DET defined. If the alternate LOS detector is desired, the user builds the code with ALT_LOS_DET defined.

Both types of LOS monitors can be configured to detect phase irregularities or missing clock edges. Each of the input LOS circuits has its own programmable sensitivity which allows ignoring missing edges or intermittent errors. The LOS status for each of the monitors is accessible by reading status registers. The live LOS register always displays the current LOS state and a sticky register stays asserted until cleared indicating an LOS event occurred. An interrupt pin is provided to assert when any of the monitors detect an LOS condition. The interrupt output is generated from the sticky LOS bits. Each bit may be masked to prevent interrupt assertion is desired. An option to disable any of the LOS monitors is also available.

1.5 Period Measurement

The ICE supports on-demand period measurements for each of the *n* inputs with resolutions as fine as 1 ppb. When a measurement is desired, the user (via the host interface) selects the input to be measured, configures the measurement settings based on the desired resolution, and requests a measurement. The request bit is cleared by hardware when the measurement completes signaling to the user that period data is now available. Only one input may be measured at a time. This is because a common circuit is used for all inputs in an effort to keep the gate count at a minimum.

It is important to understand the difference between resolution and accuracy when performing period measurements. Resolution is the smallest change that can be detected in a measurement. Accuracy is the uncertainty with respect to an ideal or absolute standard. As mentioned above, the ICE can obtain resolutions as fine as 1 ppb. However, accuracy depends on the quality of the input reference clock. Since all measurements are made with respect to the reference clock, the more accurate the reference clock, the more accurate the measurements will be. For this reason, a stable and accurate reference (e.g. TCXO or OCXO) should be used for best performance.

1.6 Serial Interface

The host processor communicates with the ICE using a 4-wire SPI interface operating at a maximum frequency of 20 MHz. The device uses a command format similar to Skyworks' Si534x/8x devices.

Table 1.1. SPI Command Format

Instruction	1st Byte[1]	2nd Byte
Set Address	000x xxxx	8-bit Address
Write Data	010x xxxx	8-bit Data
Read Data	100x xxxx	8-bit Data
Write Data + Address Increment	011x xxxx	8-bit Data
Read Data + Address Increment	101x xxxx	8-bit Data

Notes:
1. x = Don't care (1 or 0)

Writing or reading data consist of sending a `Set Address` command followed by a `Write Data` or `Read Data` command. The `Write Data + Address Increment` or `Read Data + Address Increment` commands are available for cases where multiple byte operations in sequential address locations is necessary. The left figure below shows an example of writing three bytes of data using the write commands. The right figure below provides a similar comparison for reading data with the read commands.

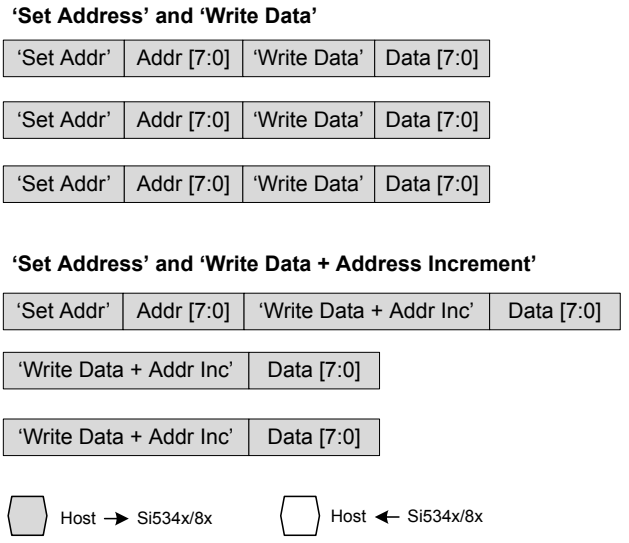


Figure 1.1. Example Writing Three Data Bytes

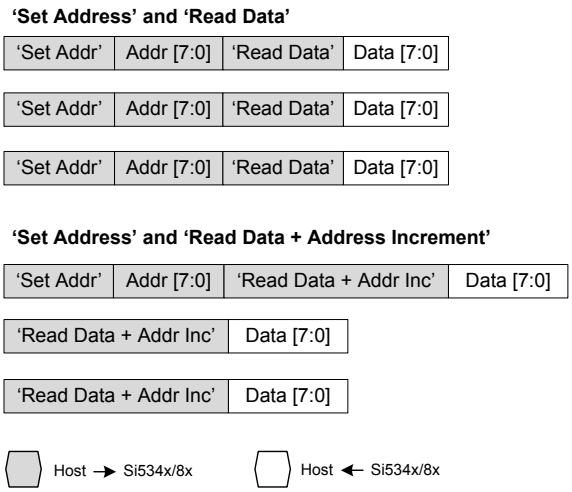


Figure 1.2. Example Reading Three Data Bytes

1.7 Host Processor

The host processor is responsible for configuring the device and selecting the input clocks that will be passed to the DSPLL. The LOS monitors and the period measurement can be used by the host processor to help select the best input. No other tasks are required from the host processor.

2. Design Details

The figure below is the ICE block diagram. The sections that follow describe each of these blocks in detail.

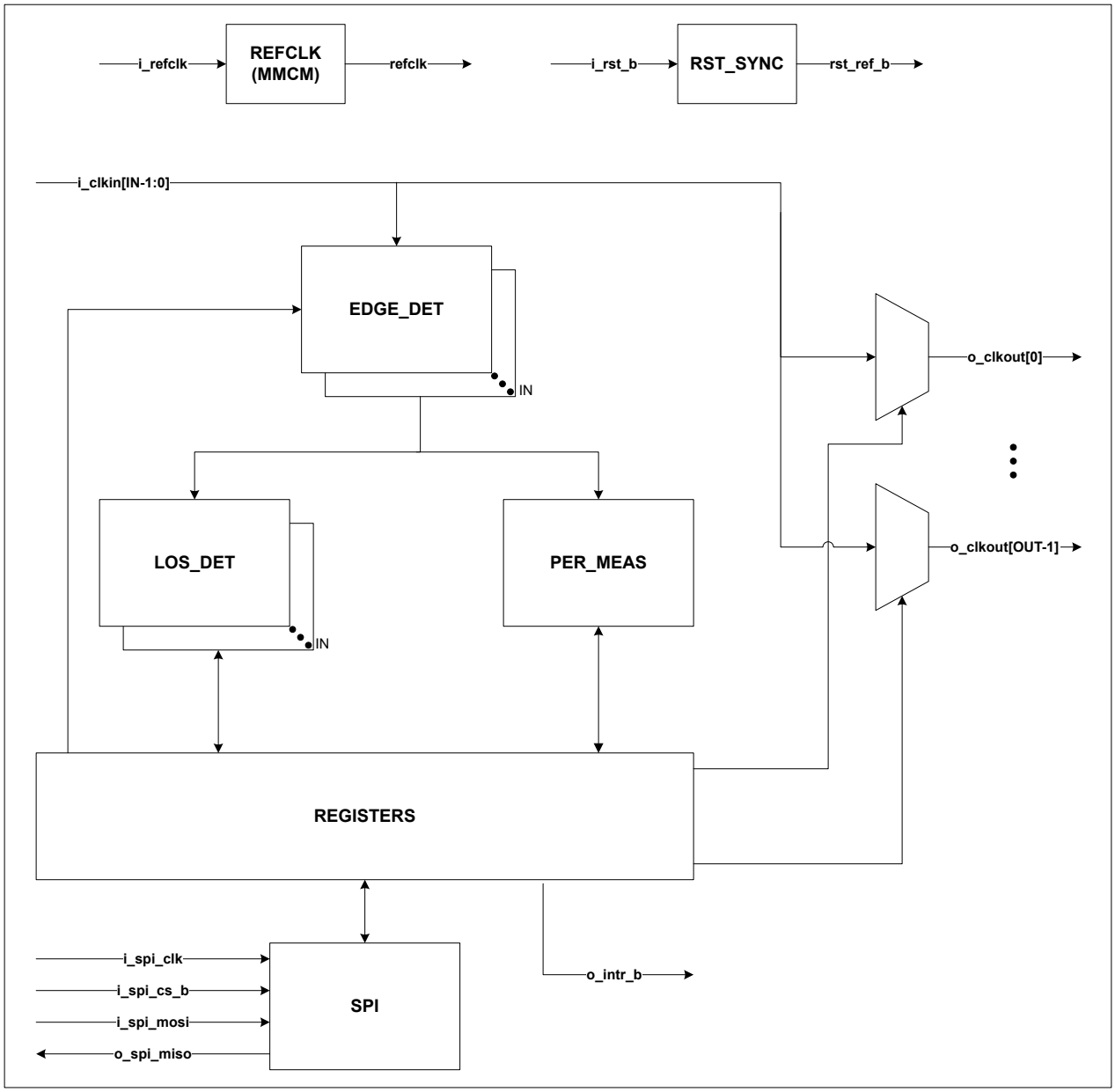


Figure 2.1. Input Clock Expander Block Diagram

2.1 Top-Level (MUX_TOP)

This is the top-level module. It instantiates the logic which implements the Input Clock Expander. The figure below illustrates the design hierarchy.

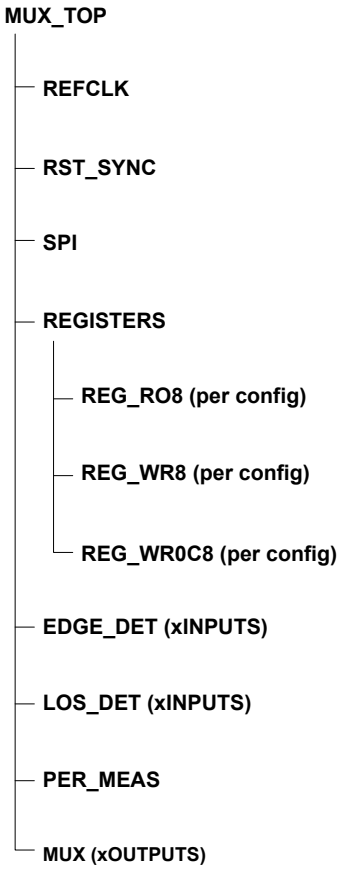


Figure 2.2. Input Clock Expander Design Hierarchy

2.2 Clock Distribution (REFCLK)

This module is generated by Xilinx's Clocking Wizard. It instantiates and configures a mixed-mode clock manager (MMCM) to multiply the input reference clock (i_refclk) up to the 100 MHz required by the ICE design. The expectation is that the user will regenerate this file based on the reference frequency available to them in their application. This MMCM should be generated with the frequency synthesis feature enabled and balanced jitter optimization. Also, an active high reset along with locked, input clock stopped, and feedback clock stopped status signals should be enabled. These status signals are combined into a discrete status pin at the top-level which allows the host to monitor the health of the internal reference clock. The choice of clock resources is left to the discretion of the user.

2.3 Reset Synchronizer (RST_SYNC)

This module provides asynchronous reset assertion and synchronous reset negation. This ensures there will be no timing violations when the reset is negated. Specifically, the negation of reset will not violate the flip-flop recovery window. An added benefit of this circuit is that all flip-flops in the same clock domain will come out of reset on the same clock edge. This is something that can't be guaranteed when using asynchronous reset negation. This module includes an enable input that is connected to the locked output of the MMCM. This ensures the device remains in reset until the MMCM has locked and the output clock is stable.

Only a single reset synchronizer is used in the ICE design and it is clocked off of the 100 MHz internal reference. A separate reset was not created for the SPI clock domain. The SPI master uses a gated clock which is active only during bus transactions. Therefore, a synchronized reset would be negated in the middle of the first transaction corrupting the transfer. Using a reset synchronized to the internal reference clock avoids this issue because the logic is released from reset long before the first bus transaction. SPI reset recovery issues are avoided because the clock is not active when reset is released.

2.4 SPI Interface (SPI)

This module implements a 4-wire SPI slave. As discussed in the Overview section, this module uses a command format similar to the Si534x/8x devices. The only difference is that the `Burst Write Data` command is not supported. This module is implemented based on the following assumptions:

1. The SPI master uses a gated clock which is active only during bus transactions (i.e. when i_spi_cs_b is low).
2. The SPI clock (i_spi_clk) is inactive on reset negation. Therefore, there are no recovery issues with an asynchronous reset negation.
3. Sufficient setup/hold exists on chip-select (i_spi_cs_b) with respect to the SPI clock (i_spi_clk) to support using i_spi_cs_b as an asynchronous reset to the internal bit counter.
4. The SPI master-in slave-out (o_spi_miso) will always be driven to avoid floating inputs on the SPI master.

The SPI slave consists of a bit counter, serial-in parallel-out shift register, clock domain crossing (CDC) logic, register sequencer, and a parallel-in serial-out shift register. The bit counter controls all SPI slave timing. As mentioned above, i_spi_cs_b is used to asynchronously reset the counter when it is negated. While not ideal, this is the only way to synchronize the count to i_spi_cs_b when using a gated SPI clock. The serial-in parallel-out shift register shifts in new data on each rising edge of the clock. The bit counter value determines when a command byte has been received and when a data byte has been received. A separate transfer request is asserted for each of these cases and is used by the CDC logic to latch the command/data in the reference clock domain. With both command and data now in the reference clock domain, the register sequencer generates the signals required to read/write internal registers. It is also responsible for latching and incrementing the register address based on the commands received.

In the output direction, the parallel-in serial out shift register loads read data from the REGISTERS module at the appropriate time (based on the bit count) when a read command has been received. Once loaded, data is shifted out on each clock edge with the old data being replaced with zeros. The shift register operates off of the rising edge of the SPI clock. However, to meet the SPI timing requirements, data must be launched on the falling edge. This re-timing is accomplished by sending the shift register output through a single flip-flop clocked off of the falling edge of the SPI clock.

2.5 Registers (REGISTERS)

This module contains all of the registers required by the ICE design. It operates off of the 100 MHz internal clock and includes multiple instances of the following modules:

- REG_RO8: 8-bit Read Only Register
- REG_RW0C8: 8-bit Read/Write 0 to Clear Register
- REG_RW8: 8-bit Read/Write Register

Register instances are based on compiler directives and match the register map provided in the attached spreadsheet (ice_register_map.xlsx). In addition, this module also implements the interrupt logic and the readback mux. Mux is really a misnomer. In reality, it is just a large OR gate followed by a register latch. The individual 8-bit registers all provide their own address decoding. When selected for a read, they output the register contents. Otherwise, they output all zeros. This allows the OR function to be used instead of a true mux.

2.6 Edge Detection with Prescale (EDGE_DET)

This module serves two purposes. First, it provides prescale dividers (1, 2, 4, or 8) for the input clock. Second, it selects one of the four pre-scaled frequencies based on register configuration and detects the rising edge. Asynchronous edge detection requires that we sample a high and a low each input clock cycle. To satisfy this, the minimum high/low time must be greater than the reference period plus the setup/hold time of the flip-flops. The prescale dividers prior to the edge detection logic are utilized to ensure this condition is met. With a 100 MHz reference clock frequency, the user must ensure the clock feeding the edge detectors is less than or equal to 25 MHz and has a duty cycle no worse than 40/60.

The prescale dividers are implemented with a 3-bit ripple counter using local routing resources. These flops may go metastable due to input clock glitches and/or runt pulse. However, downstream logic is protected with a 3-stage synchronizer used in the edge detector circuit. Should the prescale dividers go metastable, they should resolve themselves shortly after the input clock is completely lost or within a few cycles of the clock returning. Therefore, it is expected to have minimal (if any) impact on LOS or period measurements.

The edge detection synchronizer is fed by a combinatorial mux which select one of four prescaled inputs. The mux may have output glitches when the select line changes, but there should be no glitches on the output when the select lines are static. Generally, glitches should be avoided when feeding a synchronizer, but it is acceptable in this case since the select lines will not change when the LOS or period measurement in active. Any erroneous edge detection generated during this time will be ignored.

2.7 Loss-of-Signal Detection (LOS_DET)

This module includes two versions of the LOS detector: standard and alternate. Selection of the LOS monitor type occurs at build time with compiler directives. If Si534x/8x-like detectors are desired, the user builds the code with LOS_DET defined. If the alternate LOS detector is desired, the user builds the code with ALT_LOS_DET defined.

2.7.1 Standard LOS

This implementation is functionally equivalent to the LOS detector included in the Si534x/8x devices. It utilizes separate trigger and clear thresholds along with a validation timer. The trigger threshold represents the time from the last rising clock edge to LOS assert. The validation timer represents the time from rising clock edge recover to LOS de-assert. Finally, the clear threshold represents the time by which the next rising edge must arrive to prevent restarting the validation timer.

Thresholds are measured in 100 MHz reference clock cycles. In standard mode, it is the register setting * 2^1. In PPS mode, it is the register setting * 2^{14}. Scaling is required to support at least eight cycles of the input clock at the lowest frequencies (Standard: 8 kHz, PPS: 1 Hz) without requiring more than 16-bits per register setting. Validation timer settings are specified in the table below.

Table 2.1. Standard LOS Validation Timers

Setting	Standard Mode	PPS Mode
00	2 ms	1 s
01	100 ms	2.28 s
10	200 ms	4.57 s
11	1 s	9.13 s

The design is realized with an edge counter and a state machine. The edge counter resets each time a rising edge is detected. Otherwise, it is allowed to increment pegging when full. The state machine uses this counter to ascertain the time between rising clock edges. Speaking of the state machine, it consists of three states: LOS, VALIDATE, and NORMAL. The logic starts in the LOS state when the LOS detector is enabled. It stays in this state until a rising clock edge is detected. Once detected, the state machine transition to the VALIDATE state. In this state, the validation timer counts down as long as the clear threshold is not violated. If it reaches the end of the validation timer without any violations, it enters the NORMAL state and LOS clears. If a threshold violation is detected during the validation interval, the state machine transitions back to the LOS state and resets the validation timer, essentially starting the clear process over. Once the logic enters the NORMAL state, it remains in this state as long as the trigger threshold is not violated. If it is, it transitions to the LOS state and asserts the LOS signal.

2.7.2 Alternate LOS

The alternate LOS detector is included in the design to provide a reduced gate option for those applications that may be resource limited and don't require the precise settings of the standard LOS detector. This version is an interval based LOS detector. The user selects (via the register interface) an appropriate interval based on the input clock frequency. LOS is negated if a clock edge is detected during the interval. Likewise, LOS is asserted if a clock edge is not detected in the interval. To avoid LOS chatter, the module also supports an 8-bit holdoff counter configured via the register interface. This sets the number of consecutive intervals with an edge detected before the LOS alarm is cleared. The table below provides the interval options that may be set by the user.

Table 2.2. Alternate LOS Intervals

Setting	Interval	Setting	Interval	Setting	Interval	Setting	Interval
0	20 ns	8	5.12 µs	16	1.31 ms	24	335.54 ms
1	40 ns	9	10.24 µs	17	2.62 ms	25	671.09 ms
2	80 ns	10	20.48 µs	18	5.24 ms	26	1.34 s
3	160 ns	11	40.96 µs	19	10.49 ms	27	2.68 s
4	320 ns	12	81.92 µs	20	20.97 ms	28	5.37 s
5	640 ns	13	163.84 µs	21	41.94 ms	29	10.74 s
6	1.28 µs	14	327.68 µs	22	83.89 ms	30	21.47 s
7	2.56 µs	15	655.36 µs	23	167.77 ms	31	42.95 s

The design is realized with an edge detector and an LOS detector. The edge detector sets a sticky bit each time a rising edge is detected. This sticky bit is cleared (set dominant) when a timing pulse is received corresponding the selected timing interval. The LOS detector operates when enabled and a timing pulse has been received. Regardless of the LOS state, if an edge is not detected during an interval, the LOS status is asserted and the holdoff counter is cleared (i.e. restarted). If an edge is detected during an interval while LOS is asserted, but without the holdoff satisfied, LOS remains asserted and the holdoff counter increments. If an edge is detected during an interval and the holdoff counter is satisfied, LOS is negated.

2.8 Period Measurement (PER_MEAS)

This module provides on-demand period measurements of the selected input clock. The user selects the input clock to be monitored (post prescale divider), selects the number of cycles to acquire data (2^N), and requests a measurement. The request bit is cleared by hardware once the measurement has completed, providing the necessary handshake to the user. The data returned to the user is the total number of 100 MHz reference clock cycles counted during the measurement. The result is a 32-bit fixed point number with the radix point specified by the PERMEAS_NCYC setting. This module utilizes a 32-bit counter which allows measurements up to 42 s giving the user the ability to perform very accurate period measurements if desired.

The design is realized with an edge counter and a state machine. The edge counter is reset when the request is inactive. Otherwise, it increments each time a selected clock edge is received. The state machine uses this counter to determine when the period measurement should be terminated. The state machine itself consists of three states: IDLE, MEASURE, and ACK. As the name implies, the state machine sits in the IDLE state until a measurement is requested. When a measurement is requested and a clock edge is detected, it transitions to the MEASUREMENT state. While in this state, the period counter increments (pegs when full) until the edge counter indicates the measurement is complete. Once complete, the state machine asserts the acknowledge signal and transitions to the ACK state. It remains in this state until the request line is negated completing the 4-phase handshake. Once the request line is low, the user can read the period count via the register interface. Should the host forcefully negate the request line (i.e. by writing a zero), the state machine will terminate the active measurement and return to the IDLE state. Note that the period counter pegs when full. Since the counter has been sized to handle the advertised measurements, a returned value of all F's would indicate a problem with the measurement. One such error that would cause this is losing a clock during a measurement.

2.9 Clock Multiplexer (MUX)

This module includes two versions of the clock multiplexer: standard and glitchless. Selection of the mux type occurs at build time with compiler directives. If standard muxes are desired, the user builds the code with STANDARD_MUX defined. If glitchless muxes are desired, the user builds the code with GLITCHLESS_MUX defined.

2.9.1 Standard Mux

This is a standard N-to-1 combinatorial mux. There may be output glitches when the select lines change, but there won't be any glitches when the select lines are held static. Put another way, the output clock will be glitch-free once changes to the select lines have had a chance to settle.

2.9.2 Glitchless Mux

This module implements a glitchless clock mux using LUT-based muxes, a state machine, and Xilinx's BUFGCTRL primitive. The BUFGCTRL primitive is the foundation of the design and provides a glitchless 2:1 mux. The state machine and LUT-based muxes merely extend the number of inputs from 2 to N. With this implementation, glitchless switching is guaranteed unless the clocks are intermittent during the clock switch. If both clocks are active or one clock is completely dead, the switch will be free of glitches.

This implementation uses two stages of multiplexing. The first stage includes dual N:1 LUT-based muxes, one that is active and one that is inactive. The second stage is a single BUFGCTRL 2:1 mux. The state machine coordinates switching between these three muxes to achieve glitchless operation. Below are the sequence of events that occur when a switch is commanded:

1. The inactive LUT-based mux selects the destination clock and the busy bit is set.
2. Sufficient time is provided to allow any glitches to flush out.
3. The BUFGCTRL mux selects the inactive mux making it active.
4. Sufficient time is provided to ensure the BUFGCTRL logic gracefully disables the previous clock.
5. The BUFGCTRL is force to switch away from the previous clock by asserting the appropriate IGNORE signal. This ensures a switch will occur even when the "from" clock is dead.
6. Sufficient time is provided to ensure the BUFGCTRL logic gracefully enables the new clock.
7. The clock switch is completed and the busy indicator clears indicating the logic is ready for another switch.

The wait time in step 2 is fixed at 320 ns (16 reference clock cycles) while the wait time in step 4 and 6 depends on the PPS mode settings. When clocks are identified as PPS, the associated wait time for enable/disable is 2.25 s. When the clocks are identified as standard inputs, the associated wait time is 281 µs. Therefore, the maximum time to complete a clock switch is 4.5 s when switching from a PPS input to another PPS input. Likewise, the minimum switching time occurs when switching from a standard input to anther standard input. In this case, the switch time would be 563 µs.

3. Pin Descriptions

Table 3.1. Input Clock Expander Pin List

Pin Name	Pin Type	Function
System Interface		
i_rst_b	Input	**Device Reset (active low).** Asynchronous reset input. The device includes an internal reset synchronizer to provide asynchronous reset assertion and synchronous reset negation. This ensures reset will not be negated during the flip-flop recovery window.
i_refclk	Input	**Reference Clock.** Any input clock frequency may be used as long as it is scaled to 100 MHz internally by the MMCM. The design attached to this application note uses a 12.8 MHz input clock. Note, this reference clock is the timebase used for both LOS and period measurements. For best performance, a stable and accurate reference (e.g. TCXO or OCXO) should be used.
o_mmcm_error	Output	**MMCM Error Indicator.** This output is asserted if the input reference clock stops, the feedback clock stops, or the MMCM loses lock. The host processor should monitor this output to determine the health of the ICE reference clock.
Host Interface		
i_spi_clk	Input	**SPI Clock.** The design supports both free-running and gated clock masters running up to 20 MHz. However, it is recommended that the input clock be inactive when the device is released from reset. This ensures reset recover issues are avoided in the SPI clock domain.
i_spi_cs_b	Input	**SPI Chip Select (active low).** This signal serves as an asynchronous reset to the SPI state counter and should be treated as edge sensitive on the PCB design.
i_spi_mosi	Input	**SPI Master Out Slave In.**
o_spi_miso	Output	**SPI Master In Slave Output.** This signal is always driven to avoid floating inputs on the SPI master.
o_intr_b	Output	**Interrupt (active-low)**. This pin is asserted low when an unmasked LOS event occurs. It should be left unconnected when not in use.
Clock Mux		
i_clkin[INPUTS-1:0]	Input	**Clock Inputs.** The design supports a flexible number of inputs in multiples of eight (*n*: 8, 16, 24, 32). Each input utilizes local routes to minimize FPGA clock resource needs. Frequencies of 1 PPS, 8 kHz - 200 MHz are supported.
o_clkout[OUTPUTS-1:0]	Output	**Clock Outputs.** The design supports up to eight (8) clock outputs. Each output utilizes local routes to minimize FPGA clock resource needs. Frequencies of 1 PPS, 8 kHz - 200 MHz are supported (i.e. there is no clock scaling included in the mux logic).

4. Utilization

The table below provides utilization data for all four permutations of LOS detectors (standard and alternate) and output multiplexers (standard and glitchless) when targeting a Xilinx Artix-7 device. In each case, the design has been configured for four outputs. It is important to note that utilization will vary based on routing congestion, technology, and build settings. Therefore, it is highly recommended that each user build the code provided with this application note to obtain the most accurate utilization data for their specific application.

Table 4.1. Input Clock Expander Utilization

Inputs	LUTs	Flip-Flops	BUFGCTRL	MMCM
LOS_DET / STANDARD_MUX				
8	2291	1124	2	1
16	4431	2012	2	1
24	6538	2903	2	1
32	8961	3809	2	1
LOS_DET / GLITCHLESS_MUX				
8	2753	1300	6	1
16	5024	2202	6	1
24	7310	3123	6	1
32	9547	4016	6	1
ALT_LOS_DET / STANDARD_MUX				
8	907	620	2	1
16	1226	940	2	1
24	1891	1260	2	1
32	2277	1580	2	1
ALT_LOS_DET / GLITCHLESS_MUX				
8	1227	796	6	1
16	1785	1128	6	1
24	2281	1460	6	1
32	2950	1780	6	1

5. Design Files

5.1 Source Code

The Verilog source code is associated with this application note on the Skyworks website. To achieve the desired functionality and performance, the code includes both Xilinx synthesis attributes (DONT_TOUCH and ASYNC_REG) and instantiations of Xilinx primitives (MMCME2_ADV, IBUF, IBUFG, BUFG, and BUFGCTRL). Any ports of this code must address these technology specific features where appropriate. The table below lists the Verilog files included with this application note along with a brief description. Not included in this application note is the MMCM file generated by Xilinx's clock wizard. The expectation is that the user will regenerate this file based on the reference frequency available to them in their application.

Table 5.1. Input Clock Expander Source Files

File	Description
MUX_TOP.v	Top-level. This module instantiates the logic which forms the ICE.
RST_SYNC.v	Reset Synchronizer. This module provides asynchronous reset assertion and synchronous reset negation.
SPI.v	SPI Interface. This module implements a 4-wire SPI slave with the necessary clock domain transfer.
REGISTERS.v	Device Registers. This module instantiates the control, status, and command registers required by the ICE design.
REG_RO8.v	8-bit Read-Only Register.
REG_RW0C8.v	8-bit Read/Write 0 to Clear Register.
REG_RW.8	8-bit Read/Write Register.
EDGE_DET.v	Edge Detector with Prescale Dividers. This module divides down the input clock based on the configuration and detects the rising edge
LOS_DET.v	Loss-of-Signal Detector. Based on compiler directives, this module implements one of the two possible LOS detectors: standard or alternate.
PER_MEAS.v	Period Measurement. This module implements the on-demand period measurement logic.
MUX.v	Clock Mux. Based on compiler directives, this module implements one of two possible clock muxes: standard or glitchless.
MATH.v	Math Functions. This module includes a log2 function.

5.2 Constraints

The Xilinx XDC constraint file (MUX_TOP_PAR.xdc) is associated with this application note on the Skyworks website. This file includes the following constraints for a 16-input, 4-output implementation.

- Clock constraints
- I/O Timing Constraints
- Timing Exceptions
- Placement Constraints
- I/O Physical Constraints
- Configuration Settings

Note: These constraints were written for a specific Xilinx Artix-7 (xc7a50tftg256-1) evaluation board implementation. Users will need to modify these constraints as necessary when porting this design to another device and/or board.

5.3 Register Map

The ICE register map is captured as an excel spreadsheet (ice_register_map.xlsx) and is attached to this application note. It is divided into several pages to facilitate easier navigation. A summary of each of these pages is included below.

- Register Map: A high-level overview of the register map partitioning.
- Device Registers: Configuration and status registers that apply to the whole device.
- Standard Mux Registers: Configuration and status registers that apply to the standard multiplexer.
- Glitchless Mux Registers: Configuration and status registers that apply to the glitchless multiplexer.
- LOS Registers: Configuration and status registers that apply to the standard LOS detector.
- Alternate LOS Registers: Configuration and status register that apply to the alternate LOS detector.
- Period Measurement Registers: Configuration, status, and command registers that apply to the period measurement.

6. License Agreement

```
END-USER LICENSE AGREEMENT
IMPORTANT: READ CAREFULLY BEFORE AGREEING TO TERMS.
```

THIS PRODUCT CONTAINS CERTAIN COMPUTER PROGRAMS AND OTHER THIRD PARTY PROPRIETARY MATERIAL ("LICENSED PRODUCT"), THE USE OF WHICH IS SUBJECT TO THIS END-USER LICENSE AGREEMENT. INDICATING YOUR AGREEMENT CONSTITUTES YOUR AND (IF APPLICABLE) YOUR COMPANY'S ASSENT TO AND ACCEPTANCE OF THIS END-USER LICENSE AGREEMENT (THE "LICENSE" OR "AGREEMENT"). IF YOU DO NOT AGREE WITH ALL OF THE TERMS, YOU MUST NOT USE THIS PRODUCT. WRITTEN APPROVAL IS NOT A PREREQUISITE TO THE VALIDITY OR ENFORCEABILITY OF THIS AGREEMENT, AND NO SOLICITATION OF SUCH WRITTEN APPROVAL BY OR ON BEHALF OF SKYWORKS SOLUTIONS, INC. ("Skyworks") SHALL BE CONSTRUED AS AN INFERENCE TO THE CONTRARY. IF THESE TERMS ARE CONSIDERED AN OFFER BY Skyworks, ACCEPTANCE IS EXPRESSLY LIMITED TO THESE TERMS.

LICENSE AND WARRANTY: The Licensed Product and the embedded Software which is made the subject of this License is either the property of Skyworks or a third party from whom Skyworks has the authorization to distribute to you subject to the terms of this Agreement. This Licensed Product is protected by state, federal, and international copyright law. Although Skyworks continues to own the Licensed Product and the right to distribute the embedded third party Software, you will have certain rights to use the Licensed Product and the embedded Software after your acceptance of this License. Except as may be modified by a license addendum which accompanies this License, your rights and obligations with respect to the use of this Product and the embedded software are as follows:

1. AS APPROPRIATE WITH RESPECT TO THE LICENSED PRODUCT, YOU MAY: Use, copy, distribute and make derivative works of the Software for any purpose, including commercial applications, subject to the following restrictions: (i) The origin of this software must not be misrepresented; (ii) you must not claim that you wrote the original software; (iii) altered source versions must be plainly marked as such, and must not be misrepresented as being the original software; and (iv) any notices contained in the Software may not be removed or altered, including notices in source code versions.

2. YOU MAY NOT: (A) Sublicense, assign, rent or lease any portion of the Licensed Product or the embedded Software; or (B) Remove any product identification, copyright or other notices that appear on the Licensed Product or embedded Software.

3. Limited Use: Use of any of the Software is strictly limited to use in systems containing one or more Skyworks products when the Software is enabled to be functional. Any unauthorized use is expressly prohibited and will constitute a breach of this Agreement.

4. Warranty: Skyworks does not warrant that the Licensed Product or embedded Software will meet your requirements or that operation of the Licensed Product will be uninterrupted or that the embedded Software will be error-free. You agree that the Licensed Product is provided "AS IS" and that Skyworks makes no warranty as to the Licensed Product or embedded Software. Skyworks DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT, RELATED TO THE SOFTWARE, ITS USE OR ANY INABILITY TO USE IT, THE RESULTS OF ITS USE AND THIS AGREEMENT.

YOU MAY HAVE OTHER RIGHTS, WHICH VARY FROM STATE TO STATE.

5. Disclaimer of Damages: IN NO EVENT WILL Skyworks BE LIABLE TO YOU FOR ANY SPECIAL, CONSEQUENTIAL, INDIRECT, OR SIMILAR DAMAGES, INCLUDING ANY LOST PROFITS OR LOST DATA ARISING OUT OF THE USE OR INABILITY TO USE THE LICENSED PRODUCT EVEN IF Skyworks HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

SOME STATES DO NOT ALLOW THE LIMITATION OR EXCLUSION OF LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES. SO THE ABOVE LIMITATION OR EXCLUSION MAY NOT APPLY TO YOU.

IN NO CASE SHALL Skyworks' LIABILITY EXCEED THE PURCHASE PRICE FOR THE LICENSED PRODUCT. The disclaimers and limitations set forth above will apply regardless of whether you accept the Licensed Software.

6. Term and Termination: The term of this Agreement and the License granted herein shall begin upon use of the Licensed Product and continue in perpetuity unless you breach any of the obligations set out under this Agreement. Upon your breach of this Agreement by you, the license granted hereunder shall terminate immediately and you shall cease all use of the Licensed Products and return same as well as any copies of the Licensed Product and/or embedded Software to Skyworks immediately. Termination of this License upon your breach is only one remedy available to SKYWORKS SOLUTIONS, INC. In addition to termination of this Agreement upon your breach, Skyworks shall be entitled to seek any and all other available remedies, at law or at equity, arising from your breach.

7. Export: You shall comply with all applicable federal, provincial, state and local laws, regulations and ordinances including but not limited to applicable U.S. Export Administration Laws and Regulations. You shall not export or re-export, or allow the export or re-export of the Licensed Product, any component of the Licensed Product, or any copy of the embedded Software in violation of any such restrictions, laws or regulations, or to Cuba, Libya, North Korea, Iran, Iraq, or Rwanda or to any Group D:1 or E:2 country (or any national of such country) specified in the then current Supplement No. 1 to Part 740, or, in violation of the embargo provisions in Part 746, of the U.S. Export Administration Regulations (or any successor regulations or supplement), except in compliance with and with all licenses and approvals required under applicable export laws and regulations, including without limitation, those of the U.S. Department of Commerce.

8. General: This Agreement will be governed by the laws of the State of Texas and any applicable federal laws or regulations. The waiver by either Party of any default or breach of this Agreement shall not constitute a waiver of any other or subsequent default or breach. This Agreement constitutes the complete and exclusive statement of the mutual understanding between you and Skyworks with respect to this subject matter herein. This Agreement may only be modified by a written addendum, which has been signed by both you and Skyworks. Should you have any questions concerning this Agreement, or if you desire to contact Skyworks for any reason, please write:

```
Skyworks Solutions, Inc.
5260 California Ave.
Irvine, CA 92617, U.S.A.
```

SKYWORKS®

ClockBuilder Pro

Customize Skyworks clock generators, jitter attenuators and network synchronizers with a single tool. With CBPro you can control evaluation boards, access documentation, request a custom part number, export for in-system programming and more!

www.skyworksinc.com/CBPro

Portfolio
www.skyworksinc.com/ia/timing

SW/HW
www.skyworksinc.com/CBPro

Quality
www.skyworksinc.com/quality

Support & Resources
www.skyworksinc.com/support

Skyworks Solutions, Inc. | Nasdaq: SWKS | sales@skyworksinc.com | www.skyworksinc.com
USA: 781-376-3000 | Asia: 886-2-2735 0399 | Europe: 33 (0)1 43548540 |