FPGA Design Considerations for HardWire™ Designs

Preliminary

Product Specification

HardWire Array Design Considerations

It is important to observe good design practices when using Xilinx FPGAs. It is possible that the FPGA device can “hide” some less than obvious design shortcomings. However, these problems can affect the performance of the application when the design is converted to a HardWire Array.

For example, a small glitch generated from unstable inputs can be filtered by the pass transistors used to control the routing of long nets in an FPGA. However, in the HardWire Array, the pass transistors are replaced by a short connection, and the unfiltered glitch propagates through the system.

Another example of an important design consideration has to do with FPGA designs which have not been fully simulated. The customer logic may have an exceptionally long net delay in a synchronous path which could be so long as to miss the setup and hold time requirements for the flip-flop. The simulation may not report a timing error because none occurred (the signal didn’t transit during a setup or hold time window). The logic may even work in the system! Nonetheless, a functional error has been left in the design. This problem could manifest when using a faster FPGA, or different environmental conditions. Furthermore, when the design is converted to a HardWire Array, the long net delay (that allowed the FPGA to work) will be substantially reduced and could cause a functional error in the logic.

Xilinx recommends the designer perform both unit-delay and maximum timing simulations. A unit delay timing simulation can be thought of as a “best case”, since the logic will always perform worse than unit delay. This sets the minimum pole for the simulation. A maximum simulation provides the maximum pole. If there are no functional differences between the two poles, then the design is likely to be free of any timing dependencies like those described above. See Figure 1. Since this only works for what has been simulated, it is further recommended that the system be tested within its environmental constraints (i.e., four corner testing).

Xilinx guarantees that logic which is functionally accurate in the FPGA will operate correctly in the HardWire Array. If timing dependencies, or any of the issues listed below exist, please notify Xilinx when the design is submitted for conversion.

In the FPGA, the net delays are built of two elements: Metal and PIPs (programmable interconnect points, which are built out of pass transistors). The metal has particular thermal characteristics while the PIPs have different thermal characteristics. Generally, at low temp, High V_{DD}, the circuit will run fast. At high temperature, low V_{DD}, the circuit will run slower. Designs should be very carefully simulated at both unit -delay (functional) and maximum. These two simulations should then be compared against each other so that any functional errors may be found and fixed. This is a particularly important step to guarantee the HardWire Array will operate in the system, once it is inserted.

In the HardWire Array, the net delay is purely a function of the metal wire since the pass transistors have been removed from the design. Since the device die area in the HardWire Array is reduced, net delays will be consequently smaller. For example, in a small net with only one PIP, both the FPGA net delay and the HardWire net delay may be 1.2 ns. However, if the FPGA has a 50 ns net, caused by many different PIPs, the HardWire device might only be 5 ns. Where feedthroughs are used in the FPGA, they are duplicated in the HardWire to mirror the timing of the FPGA.

Figure 1 shows an example relationship of timing in the FPGA compared to the HardWire. Note that the far right axis is the FPGA’s maximum delay. On the far left side is the minimum theoretical timing. These two poles are direct results of the unit-delay simulation for the left side and the real “speeds” file based simulation for the right side. The FPGA design will inherently run somewhere between these two poles. If the user has performed both simulations and compared the results, with no functional errors, then the design can be assured to work with in both the FPGA and the HardWire Array.

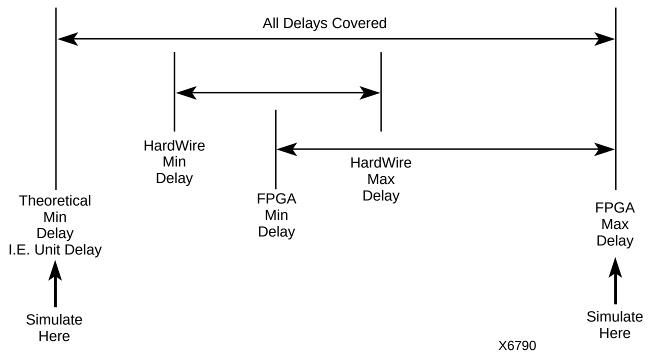


Figure 1.

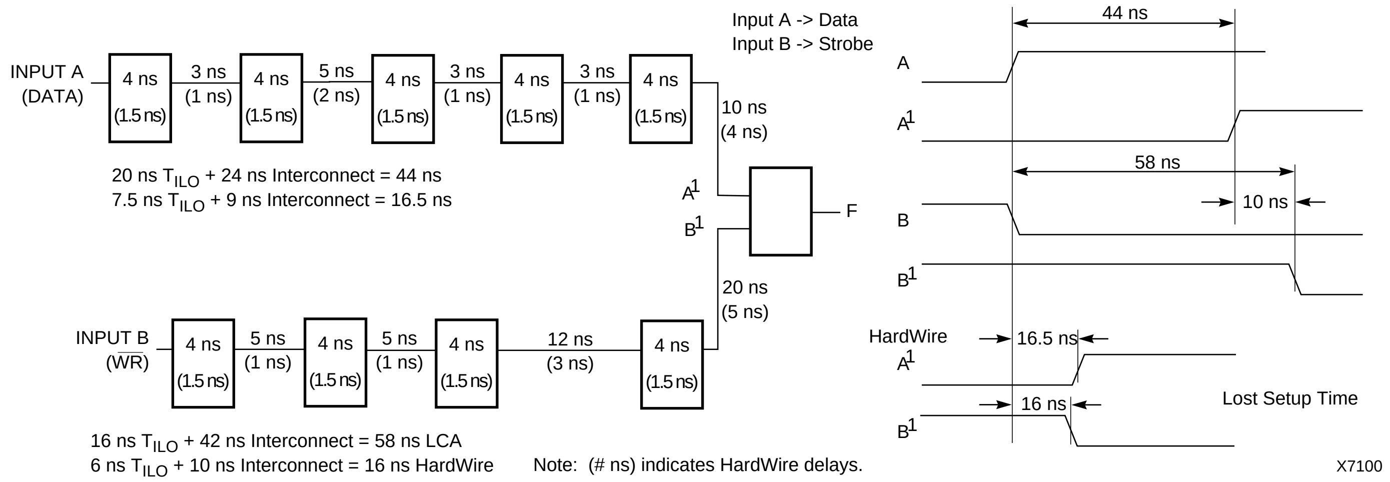

INPUT A (DATA)
4 ns (1.5 ns)
3 ns (1 ns)
5 ns (2 ns)
10 ns (4 ns)
20 ns T_{ILO} + 24 ns Interconnect = 44 ns
7.5 ns T_{ILO} + 9 ns Interconnect = 16.5 ns
A1
B1
F
20 ns (5 ns)
INPUT B (WR)
12 ns (3 ns)
16 ns T_{ILO} + 42 ns Interconnect = 58 ns LCA
6 ns T_{ILO} + 10 ns Interconnect = 16 ns HardWire
Note: (# ns) indicates HardWire delays.
Input A -> Data
Input B -> Strobe
44 ns
58 ns
10 ns
A
B
HardWire
16.5 ns
16 ns
Lost Setup Time
X7100

Figure 2. Race Condition Example

Since the HardWire uses only metal and the FPGA uses both metal and PIPs, net delays will be smaller for the HardWire array. As a result, the HardWire devices will run faster than FPGAs. Therefore, the HardWire poles will always sit between the actual worst case and the theoretical functional best case operation of the FPGA.

If the customer has performed both the simulations referenced above with the FPGA, then no further simulation is required in order to guarantee the HardWire will meet internal performance specifications. External specifications depend upon meeting setup and hold relationships. These are covered in the “System Level Setup / Hold Time Requirements” section.

Gated Clocks and Reset Directs

Glitching function generators driving CLOCK and RESET DIRECT pins can inadvertently trigger flip-flops to an undesirable state. Avoid using “gated” clocks and reset directs.

Xilinx recommends the use of flip-flops with CLOCK ENABLES instead. This is good practice for circuits driven from derived clocks (e.g. divided by N of a fundamental clock frequency). In these situations, the CLOCK pin can be the fundamental, and the clock ENABLE pin should be tied to the derived clock.

If gated clocks or gated reset directs are unavoidable, Xilinx HardWire Engineering should be notified of these nets prior to conversion so that they may be analyzed. For any circuits that use gated clocks or reset directs, the logic should be designed so that inputs are always stable and that the signal changes are at least one CLB (T_{ILO}) delay apart from one another.

Multiplexers Implemented in Function Generators

Two input multiplexers can be implemented in a single F or G function generator. However, there is a possibility of generating a glitch if the selected signal and input changes within a CLB’s T_{ILO} delay. This is generally not a problem with data and address multiplexers if the output is given enough time to settle. If the multiplexer is feeding a clock or reset direct input for a CLB flip-flop, it is possible to toggle the register at an undesired time.

To avoid this problem, the edge(s) of select signals for a clock and/or reset direct multiplexer should be stable before and after the edge(s) of the inputs. The edges should be at least one CLB T_{ILO} delay apart.

Race Conditions

All race conditions in the circuit need to go through a careful analysis. Depending upon the routing resources responsible for the net delays the correct signal may always “win the race.” Then again, it might not! See Figure 2.

Delay Generators

Using the routing resources as delay lines in the FPGA is highly undesirable. This practice can create timing problems in the FPGA as well as the HardWire Array, so all delay generators should be removed from the design. This redesign effort should be completed and tested prior to submitting an FPGA for HardWire conversion.

Combinatorial Loops

Combinational feedback loops may cause incorrect circuit operation due to differences in routing delays. This design practice causes problems in the FPGA, and Xilinx strongly discourages the use of these circuits. If faster FPGAs are used in the application, significant functional problems could be experienced

In addition, the problem can be even worse in the HardWire Array. This is particularly true when the combinational loop has a lot of delay caused by net delays. The result of such circuits in a HardWire may not be the same as that of the FPGA.

One-Shots

One shots are often implemented by feeding the Q pin back to the RD pin where the pulse width is determined by the RD to Q delay + routing. In some cases there is a chain of logic between the Q to the RD. (See Figure 3.) The FPGA's cumulative delay for this path will always be larger than the HardWire Array and will therefore have a wider pulse. These circuits should be used with great care, and should not be used to clock other flip-flops since the width is not guaranteed to produce a viable clock pulse. If such a circuit is used in the FPGA design, it should be identified to Xilinx at the time of design submittal so that it can be properly analyzed.

Choppers or Differential Circuits

Chopper or differential circuits where the pulse width is determined by the difference in delays between two reconvergent paths should not be used. Designs which contain such circuits should be redesigned so as not to be dependent upon delay. Figure 4.

Ring Oscillators

Ring Oscillators, where the oscillation frequency is determined by the propagation delay time through a chain of inverters in the ring, should never be used in FPGA-based design.

"Tweaked" FPGA Design

Any design technique or structure that is normally unpredictable, but is "tweaked" to work in the FPGA may not work in the HardWire Array. "Tweaking" includes deliberate lengthening of routing to meet hold time, addition of extra delay gates to lengthen a path, etc.

TBUF Multiplexers

The FPGA supports the ability to build large data path multiplexers by utilizing three-state TBUFs. The HardWire Array converts these TBUFs into real multiplexers of the appropriate size and width. The TBUF longline Pull-up in the FPGA guarantees a high state on the longline if no TBUF is on. The HardWire Array maintains this functionality by guaranteeing the multiplexer output is high when no specific input is selected.

Simultaneous Switching Outputs

Current FPGAs have substantially more I/O pins than previous generations of FPGAs. Today's systems continue to increase in bit width as well. The result of this is a potential problem in the FPGA design which is known as Simultaneous Switching Outputs (SSOs).

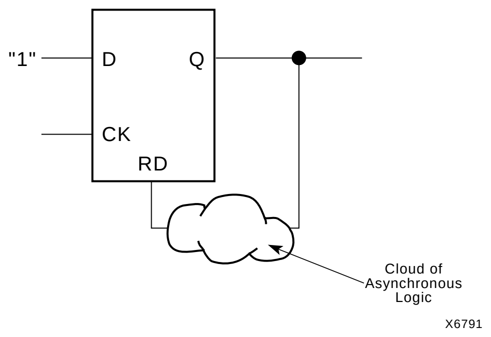


Figure 3. Example of a One Shot Circuit to Avoid

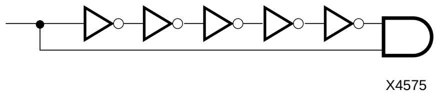


Figure 4. Differential Circuits

Simultaneous Switching Outputs can generally be defined to be the total number of outputs which switch from one state to another state within a certain amount of time. In the case of the Xilinx FPGAs, a working definition for SSOs is the number of I/O which switch within 8 ns of each other.

Figure 5 shows examples of both synchronous outputs relative to each other. The system clock provides the synchronous outputs A on the rising edge of the clock while the synchronous outputs B are generated off the falling edge. In addition, some timing relationship exists between the asynchronous outputs C and the synchronous output A. This timing relationship allows the C outputs to change within 8 ns of the A outputs. This makes ALL of the A outputs and C outputs SSOs. Conversely, Synchronous Output B and Asynchronous Output D do not switch within 8 ns of A or C, nor with each other. Therefore outputs B and D are not SSOs relative to A or B.

Another not-so-obvious aspect of SSOs in an FPGA has to do with synchronous outputs from CLB FFs. In this case, although all the internal flip-flops switch at the same time, the external signals may not due to routing delays from the CLB to the IOB. Since the HardWire will tend to equalize these delays, a SSO issue may present itself in the HardWire that does not *SEEM* to present itself in the orginal design. In reality, a faster FPGA may also experience this problem. Therefore, careful analysis should be done to avoid or minimize the effect.

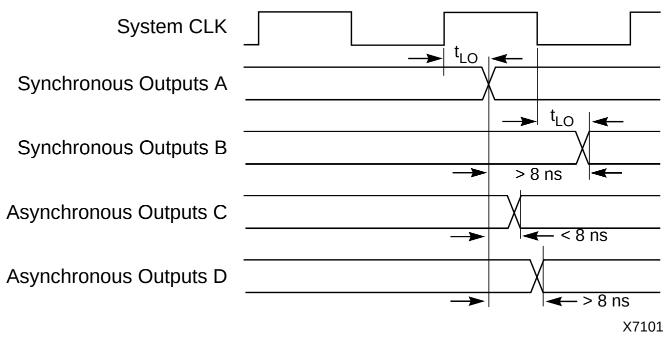
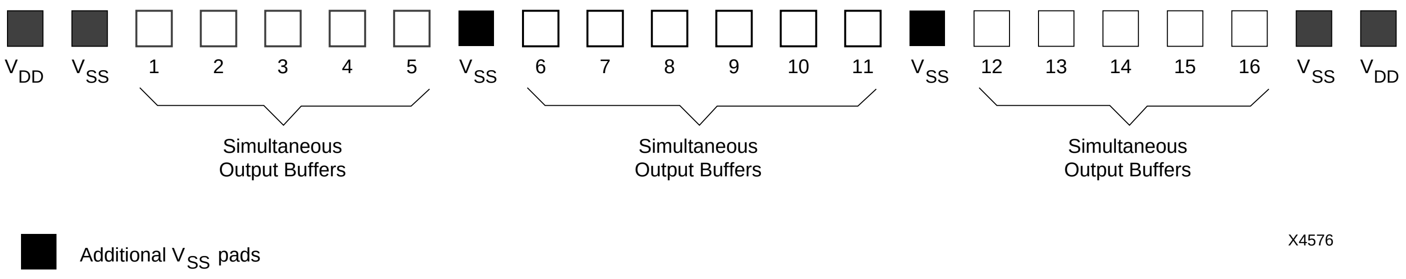


Figure 5. Examples of SSO Timing



VDD
VSS
1
2
3
4
5
VSS
6
7
8
9
10
11
VSS
12
13
14
15
16
VSS
VDD
Simultaneous Output Buffers
Simultaneous Output Buffers
Simultaneous Output Buffers
Additional VSS pads
X4576

Figure 6. Simultaneous Switching Outputs

SSOs can cause a wide variety of problems in customer designs. The most obvious problem is internal ground bounce. This ground bounce (or for that matter V_{DD} bounce) can sometimes be severe enough to cause a clock transition on a stable internal FF. Further, the slew rate of the outputs can cause problems to the system because of undershoot or overshoot. In this case, the input of another system can sense an incorrect logic level. In addition, if the output has enough ringing, excess energy can cause damage to output and input structures in the ICs of the system. SSOs can therefore affect logic functionality, system integrity, and long term reliability.

There are some general guidelines which should be followed in order to avoid these effects. First, the single most important cause of SSO negative effects is relatively high inductance paths for the power and ground paths of the FPGA. It is very important to have sufficent ground or power pins. In the FPGA design, there are certain guidelines which should be followed in order to minimize SSOs.

The basic guidelines for SSOs are to have no more than 8 I/Os per V_{DD}/V_{SS} pair. This implies distributing the outputs across the die, possibly across multiple edges. In addition, it is imperative to make sure that global inputs, particulary clocks are isolated with their own pairs of power or grounds. Another rule of thumb to follow is to allow no more than 100 mA of IOH/ IOL current per V_{DD}/V_{SS} pair.

The customer should ensure that the package pin for the power or ground has the smallest possible inductance to a PCB power plane. Xilinx strongly recommends the use of separate power and ground plane layers on PCBs when contemplating designs which might have SSOs. Second, adequate high and low frequency bypass capacitors should be placed as close as physically possible to the power and ground pins.

A second aspect that affects SSOs is the slew rate of the output. Outputs which have very fast slew rates will tend to cause more system level problems for SSOs. Outputs which have slower slew rates will have less SSO effects. In the FPGA two different slew rates are available for the I/O. Whenever possible, the slow I/O slew rate should be used.

When a printed circuit board is being designed, consider leaving room for series termination resistors to reduce the effective rise and fall times. Series resistors offer the opportunity to tune the PCB impedance to the driver impedance. The artifact of adding series termination to critically dampen an output will be to reduce all of the effects of SSOs. An added benefit is that it will be easier to meet EMI /EMC guidelines for the whole system.

Clock inputs for the FPGA should be kept as far away as possible from SSOs. If possible, they should have their own ground pin that is not shared with other outputs. Since one of the effects of SSOs is internal clocking of statically set flip-flops, this step can go a long ways towards system reliability and robustness.

Another tactic is moving the outputs along different edges of the package. This will help to distribute the instantaneous current requirements of the Simultaneous Switch Outputs. If distribution is possible, this is the very best way to reduce the effects caused by SSOs.

Xilinx XC4400/XC5400 HardWire Family offers a unique ability to help resolve the SSO problem. In the XC4400/ XC5400 product families any pin can be made into a V_{DD} or V_{SS}. This means that careful upfront planning of an FPGA pinout can solve the SSO problem. Consider Figure 6. Here we see a total of 16 SSOs that have had additional ***user*** I/O pins allocated to V_{DD} or V_{SS}. The user PCB has these pins tied to V_{DD}/V_{SS}, therefore the FPGA I/O pin should be configured as an input I/O PUP or PDOWN. In the HardWire the pins can be tied to V_{DD} or V_{SS}. Two important facts should be noted here: A) the FPGA user I/O pin used for V_{DD}/V_{SS} must not be a configuration pin, and B) the customer must tell Xilinx that they have used additional pins for V_{DD}/V_{SS} since that information can not be ascertained otherwise.

Both the FPGA and the HardWire Array are vulnerable to the effects of Simultaneous Switching Outputs. The HardWire Array has similar I/O characteristics as the FPGA and will therefore perform in a similar manner. Xilinx HardWire Engineering evaluates all incoming designs for SSO effects and will report these to the customer in the HardWire Design Review Report. The customer will have an opportunity to modify and resubmit the design for conversion. However, this is often a time consuming process. Therefore, Xilinx strongly recommends employing good design practices listed above, and resolving any problems in the system design stage PRIOR to HardWire conversion.

System Level Setup / Hold Time Requirements

The FPGA has specified system level Setup and Hold time requirements relative to very specific conditions. The conditions typically assume the nearest IOB/CLB flip-flop to the IOB will be used for the measurement. The times specified in the FPGA data sheet are dependent upon the time through the IOB, a net, and the setup time of the IOB/CLB FLIP-FLOP for the input case and the opposite for the output case.

The HardWire Array utilizes the same structure as the FPGA, but will have a smaller net delay than the FPGA. The reduced delay is due to the fact that the signal in the FPGA must go through at least two programmable pass transistors while the HardWire Array provides only a wire delay.

The impact on the design for the customer is an effective decrease in required setup and available hold time requirements on the inputs. In addition, the available system hold time can decrease substantiality. The input flop-flop conditions are effectively unimportant because the system will guarantee that the internal requirements are met. (See Figure 7) However, the output flip-flop conditions must be carefully reviewed.

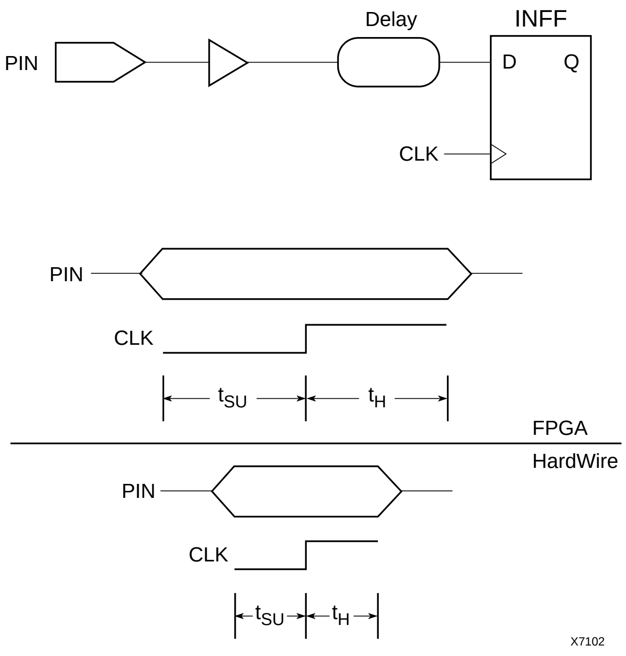


Figure 7: Demonstration of INPUT Measurements

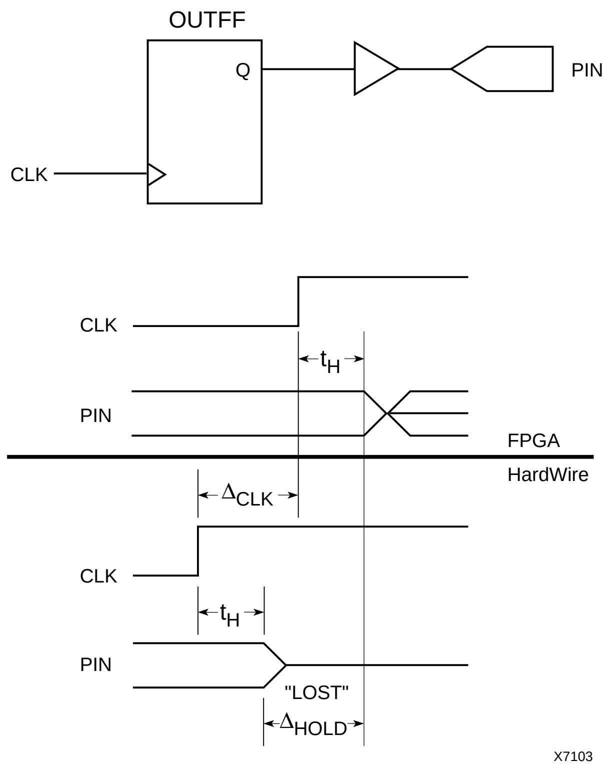


Figure 8. Output Measurements (schematic + timing diagrams)

The HardWire Array will have faster net delays compared to the FPGA. See Figure 8. In this case, the faster net delay will result in less available setup time from a clock to output perspective. Since the effective time for HardWire is likely to be different from the FPGA, Xilinx will publish the system hold time numbers in the Design Review Report. The customer will then have the option to approve those timing numbers. If additional hold time is required, Xilinx can add delay in the flip-flop Q to OBUF path to compensate.

I/O Slew Rate

The FPGA utilizes complimentary drivers providing 8 or 12 mA of source current and 8 or 12 mA of sink current. In addition, the FPGA allows for slew rate control of “fast” and “slow” outputs.

HardWire Configuration

A very important aspect for the customer to consider in HardWire is how to deal with the configuration of the device. In the FPGA world, data is stored on the system, generally in some form of PROM. This data is then downloaded to the FPGA via one of the many configuration modes. This allows the system to “wake-up” in an orderly fashion.

This orderly system wake-up is an often overlooked factor. FPGA systems typically have five stages that they must pass through prior to system operation. These are detailed in Figure 8. The basic stages are V_{DD} arriving at a valid level, Power-On-Reset, Initialization, Configuration, and Startup. At the end of the Startup phase, the user’s logic will be available. The important aspect of this is that the FPGA provides status information for the system to identify which phase it is currently in. Many systems use this status information to control different parts of the greater system. For example, the LDC pin might be used to hold a system’s CPU in reset until the FPGA is configured.

The HardWire, on the other hand, is a mask programmed device. Therefore, in theory, one should just be able to power it on, and start running the system. Many other vendors use this approach to FPGA conversion. The reality is that the HardWire must also maintain these same phases if the system is to be properly functional – even though it does not require any data to be loaded into it. Xilinx has provided a variety of mechanisms in HardWire to deal with the loading of data into the FPGA. These mechanisms are collectively known as Configuration Emulation.

HardWire Configuration Emulation

Xilinx FPGAs and the HardWire Arrays support seven different configurations modes described in Table 1. In addition, the HardWire has two additional modes - INSTANT_ON and NO_PROM. Each configuration mode is supported by “emulation” of the configuration process (patent-pending).

The HardWire Array behaves exactly like an FPGA device from power on until user operation, following the standard set of sequences which are documented in the FPGA Data Sheet. Refer to Figure 9 and the FPGA data sheets for any additional details.

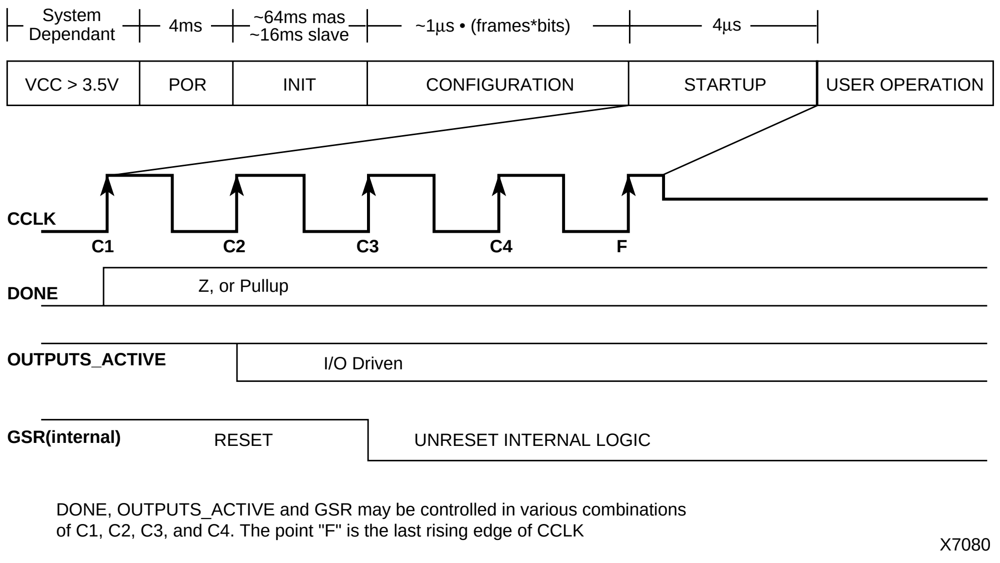


Figure 9. Details the HardWire Array behavior from power-on to user operation (XC4400 example).

Table 1. Configuration Modes

Mode	M2	M1	M0	CCLK	Data
Master Serial	0	0	0	output	Bit-Serial
Slave Serial	1	1	1	input	Bit-Serial
Master Parallel Up	1	0	0	output	Byte-Wide, ADDR 00000↑
Master Parallel Down	1	1	0	output	Byte-Wide, ADDR 3FFFF↓
Synchronous Peripheral*	0	1	1	input	Byte-Wide
Asynchronous Peripheral	1	0	1	output	Byte-Wide
Express	0	1	0	input	Byte-Wide
Reserved	0	0	1	-	-

* Synchronous Peripheral can be considered byte-wide Slave Parallel

First the device waits until the V_{DD} reaches approximately 3.5 Volts. Once power has been reliably detected, the HardWire Array will wait approximately 4 or 16 ms (POR phase) and begin the Initialization phase. During this phase an internal counter guarantees that the INIT pin is held low. Depending upon the configuration mode selected, the HardWire will wait for approximately 64 ms (master mode) or 16 ms (slave mode) longer.

Once the INIT pin is released internally by the HardWire, it will sample the INIT pin to see if another system component desires to delay the configuration phase by holding INIT low. If no other system component holds INIT low, INIT will transition from low to high where it will remain until the start-up phase has been completed. By holding INIT Low, boundary scan operation, if chosen by the customer may be performed at this time ("Before Configuration" BSCAN). When INIT transitions high, the configuration emulation logic then waits two internal clock cycles and samples the states of the mode pins (M2, M1, M0). Based upon this encoding, one of the seven configuration modes will be chosen for emulation. The encoding of the mode pins is identical to that specified in the FPGA data sheet.

At the end of the two clock synchronization period, the actual configuration phase will begin. During this configuration phase, the HardWire Array will load data from its serial or parallel path as is indicated by the configuration mode. The HardWire Array will transmit the header information to all downstream devices. The HardWire Array will swallow the correct amount of data for itself, and then passes any remaining data down stream to any other FPGA devices that exist in the chain. Once all of the FPGAs in the chain have received their proper amount of data, the HardWire Array will proceed into the Start-up phase.

The Startup phase is used to ensure that multiple FPGA/HardWire devices will wake-up in a predefined order. At this time, the flip-flops are in UN RESET, the DONE pin is allowed to float high, and the user outputs become active as specified by the customer design. The order is specified by options set in the MakeBits program, and is reported in the design_name.mbo file. The HardWire Array supports all of the specialized Start-up modes that exist in the FPGA in order to guarantee correct system operation.

Configuration Emulation: INSTANT_ON and NO_PROM Options

In addition to the standard seven configuration modes, the HardWire Array supports two other configuration modes intended to further reduce system cost. Each mode reduces the total system cost by allowing FPGA configuration storage element (SPROM, EPROM, SRAM, etc.) to be removed from the system. Each mode has unique features which must be analyzed before choosing to use the option.

INSTANT_ON

INSTANT_ON is recommended for single FPGA to HardWire design conversions where the system timing is known and no other programmable logic needs to be loaded from the FPGA. In order for this mode to be implemented in the HardWire array, it must be specified at the time of initial design submittal.

In the INSTANT_ON mode, the HardWire Array will skip the configuration phase shown in Figure 9. The device will proceed from V_{CC} reaching a valid level through the Power-On-Reset phase and then through the initialization phase. At this point, rather than entering the configuration phase, the FPGA proceeds directly to the Startup phase. Refer to Figure 10.

INSTANT_ON may be used if multiple devices are to be converted. However, all devices in the daisy chain must be HardWire Arrays configured in the INSTANT_ON mode of operation. It is not possible to have a mixed Configuration Emulation chain since the INSTANT_ON mode does not pass any data down the chain, nor does it acquire any data from the programmable storage element.

In addition, if INSTANT_ON mode is chosen, no other configuration modes will be available. The INSTANT_ON logic takes the place of the Configuration Emulation logic for this mode, and reduces the die area. When the INSTANT_ON option is chosen by the user, the mode pins M2, M1, and M0 must all be statically set on the user's PC board to the value specified in the Design Submittal Form.

The INSTANT_ON mode reduces the system's power-on-to-HardWire-operational time. This is due to the fact that no data is loaded into any of the HardWire devices on the board that use the INSTANT_ON mode. Note that Figure 10 shows the "Configuration" time of approximately 1 microsecond / bit is missing. Care should be exercised in verifying that the system is not affected by changing the overall time from power-on to the HardWire Array being ready for device operation.

NO_PROM

The INSTANT_ON option affects designs in two ways. First, INSTANT_ON changes the system timing because there is no time delay for the configuration of the HardWire Array. Second, the INSTANT_ON option does not permit the user to have other non-HardWire devices in the configuration chain. Xilinx has added an additional configuration emulation mode called NO_PROM to address both these effects.

Like the INSTANT_ON option, the NO_PROM option must be chosen by the user prior to conversion to a HardWire array. This mode works in concert with the other seven configuration modes, and is a data monitoring device that watches the first unit of incoming data to the configuration chain. If that data is an expected first piece of data, the Configuration Emulation logic will default to the NO_PROM mode.

The major advantage of the NO_PROM mode is the ability to remove the programmable element that is used to configure the FPGA at a later time. Removing the programmable element further reduces the overall system cost by saving part count and the cost of the device(s).

This feature is especially useful for designers who are interested in structuring a multiple HardWire conversion program. In this situation, the customer may not desire to convert all the FPGA designs to HardWire at once. Since any programmed FPGA on the board need to be configured, the configuration storage element is still required. However, once all the designs are in production as HardWire the configuration storage element can be removed. This can result in a substantial overall cost reduction.

The basic operation of the NO_PROM mode is nearly identical to the INSTANT_ON mode. Refer to Figure 11.

The primary difference between INSTANT_ON and NO_PROM is that when the NO-PROM mode reaches the configuration phase, it reads the first data bit / byte from the bitstream, and then determines if it is to enter the NO_PROM condition. The data stream of a FPGA expects the very first byte to be a 0xFF. If the NO_PROM option is chosen, Xilinx adds internal pull-down resistors to the D[7:0] configuration pins. (Or to the DIN pin only if a serial mode is used). Since these pins are often dedicated in the design, or are at least not driven during FPGA configuration, the HardWire Array can sense the data as a 0x00 instead of the required 0xFF. When this condition is detected, the HardWire device will automatically jump to the Startup phase, bypassing the remainder of the Configuration phase. In order to accommodate other HardWire devices in the chain, each device will also present its DOUT as a LOW. This guarantees that ALL HardWire devices in the chain will sense the LOW at the same time and will bypass the rest of the Configuration Phase. If an FPGA is in the device chain, a configuration error will result and the INIT pin will be driven low by the FPGA. Therefore the configuration storage element should not be removed until all devices in the chain are converted to HardWire Arrays.

The designer needs to review any effects of having internal pull-down resistors (approximately 50K ohms) on the D[7:0] or DIN configuration pins. **These pulldowns would take precedence over any specified user pull-ups**. Only the master device is required to have the pull-down resistor(s).

The NO_PROM mode is always present when Configuration Emulation is enabled. However, the pull-down resistors are not added to the configuration data pins unless the NO_PROM option is chosen in the Design Submittal Form. To use the No_PROM mode without the use of the internal pull-down resistor(s), the designer should ensure that the first byte or bit written to the lead device is a 0x00. (or to the customer value).

Another use of the NO_PROM mode is in identifying if a HardWire device or programmable part is present. For example, one possible method uses the Peripheral Asynchronous Configuration Emulation mode, and has both software and hardware that detects if the INIT pin is driven LOW when a 0x00 is written to the FPGA or HardWire device. The INIT pin will be driven LOW on an FPGA if it detects an error in the configuration data stream. If the INIT pin does go LOW, the device is an FPGA and requires being reset via PROG and then programmed. If INIT does not go LOW, the device will be identified as a HardWire Array, and will go straight to the start-up phase causing DONE to go HIGH.

The DONE pin could also be sampled after approximately 10 ms. If DONE is HIGH, the device is a HardWire Array and is programmed. If DONE is LOW, then the device is an FPGA and requires being reset (using PROG) and then must be programmed.

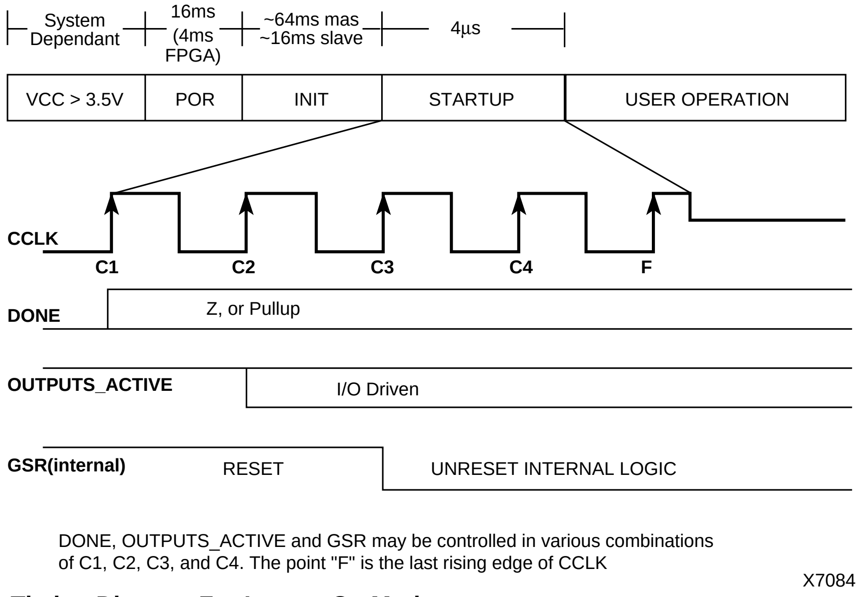


Figure 10. Configuration Timing Diagram For Instant-On Mode

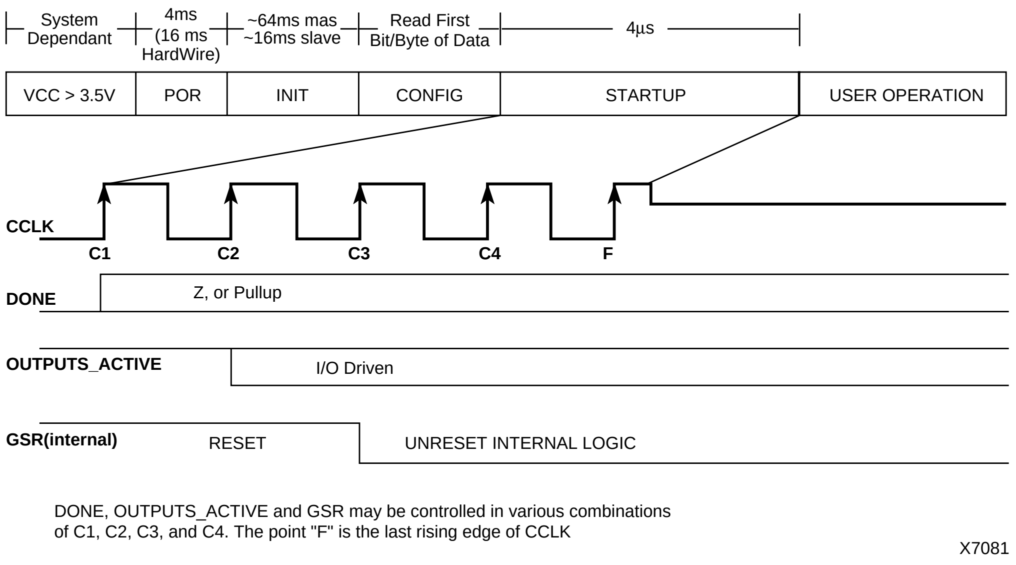


Figure 11. NO_PROM Mode Startup Sequence

HardWire Configuration Daisy Chaining

A HardWire Array can be a stand alone replacement of the corresponding FPGA, as shown in Figure 12. In a daisy chain, the HardWire Array is fully interchangeable with any programmable device in the chain as shown in example 1, 2, 3, and 4. For more information on FPGA configuration, refer to the Programmable Logic Data Book.

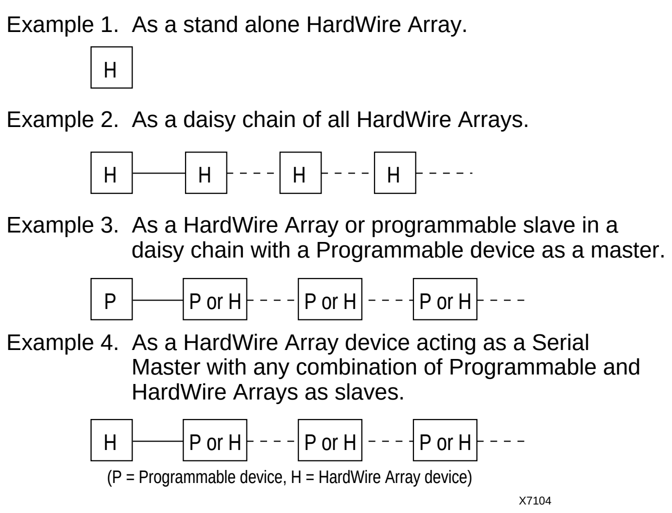

Example 1. As a stand alone HardWire Array.
H
Example 2. As a daisy chain of all HardWire Arrays.
H
H
H
H
Example 3. As a HardWire Array or programmable slave in a daisy chain with a Programmable device as a master.
P
P or H
P or H
P or H
Example 4. As a HardWire Array device acting as a Serial Master with any combination of Programmable and HardWire Arrays as slaves.
H
P or H
P or H
P or H
(P = Programmable device, H = HardWire Array device)
X7104

Figure 12.

The benefit of HardWire daisy chaining and NO_PROM configuration Emulation Mode is that different designs may be converted at different times. If one design is ready for production, but another in the daisy chain is not, the HardWire Array can emulate the full sequence to program the FPGAs in the chain. Once all FPGAs have been converted to HardWire, the PROM can be removed from the system for additional cost savings.

HardWire Makebits Options and STARTUP

The design engineer sets the order in which three major events take place in the FPGA during the startup phase:

- When the DONE pin goes High
- When the Outputs are no longer held 3-stated
- When the internal Global Set/Reset is released

The default sequence of events, as set by makebits is: DONE pin High, I/Os active, and release of internal Global Set/ Reset. At the end of the STARTUP phase, the HardWire Array will behave like a programmed FPGA. These sequences are illustrated in Figure 11.

It should be noted that the HardWire Design Center requires the .MBO (Makebits option) file in order to correctly process a customer design. This file contains the sequence of events in the STARTUP phase and is therefore necessary.

HardWire Boundary Scan

HardWire Boundary Scan User Logic supports most FPGA BSCAN User Logic modes. (Since the re-programmable elements of the FPGA have been removed from the HardWire device, the Configuration and Readback modes are not supported.) The FPGA BSCAN macro also supports the ability to connect two additional data streams corresponding to the USER1 and USER2 Boundary Scan instruction decodes. The HardWire Array supports this feature fully and automatically.

If Boundary Scan is intended to be used after the configuration process, then the BSCAN User logic must be instantiated in the FPGA. For more information on the use of the BSCAN User Logic, refer to the Xilinx Libraries Guide for the appropriate FPGA. An example of the FPGA BSCAN User Logic block is shown in Figure 13.

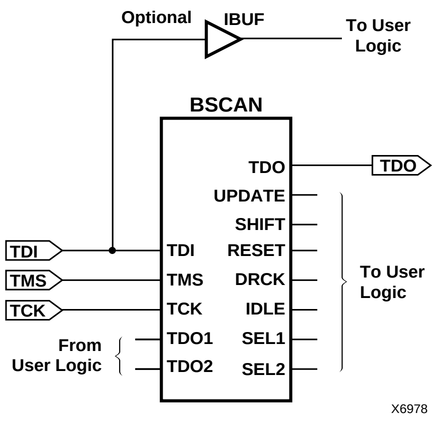


Figure 13. FPGA BSCAN User Logic Block

The HardWire Boundary Scan (BSCAN) behaves identically to the FPGA with a few exceptions. BSCAN has two major areas that are affected by the HardWire implementation: BSCAN commands and I/O logic control.

Two BSCAN commands (BSCAN Configuration and BSCAN ReadBack) have been replaced by two additional commands. Refer to Table 2 for supported command information.

These commands are useable for additional testing within the device. The first command, I-SCAN is used to allow boundary scan access to the I_LATCH portion of the IOBs (see figure 15). The second command is used to test all of the internal flip-flops in the user's design.

In the FPGA BSCAN, there are three D type register and three D type latches per IOB. Figure 14. details the logic.

Each register and latch are paired together, and then grouped into three register-latch pairs for each IOB in the device. The D registers are turned into a miniature scan chain of three elements each, while the Q of each register feeds the D of each corresponding latch. The registers are labeled as I_REG, O_REG, and T_REG to signify what element of the IOB they have control over. Each Latch is labeled I_LATCH, O_LATCH, and T_LATCH as well.

Table 2. BSCAN Supported Command Information

IR2	IR1	IR0	INSTRUCTION
0	0	0	EXTEST
0	0	1	SAMPLE/PRELOAD
0	1	0	USER1
0	1	1	USER2
1	0	0	FULL_SCAN
1	0	1	BYPASS
1	1	0	I_SCAN
1	1	1	BYPASS

X6787

The HardWire implementation is similar except that the I_LATCH is replaced by a scannable D register, see Figure 15 .

Since the FPGA I_LATCH is not testable in a "general purpose" manner, a scannable D register for the I_LATCH is substituted in the HardWire Array. Each Scan-In (SI) pin of each I_LATCH is connected to the previous IOBs I_LATCH Q output (SO). In this manner, a scan chain is built. This scan chain is accessible in Xilinx Manufacture Test Mode, or via an added BSCAN command previously discussed. See Figure 15. Since there is a D register instead of a Latch in the design, the effective time of change for the IOB.I signal is later in the TCK cycle. Since all BSCAN operations are synchronous, no functional difference exists.

The BSCAN User Logic is also supported by the HardWire Array as previously discussed. This permits designers to add up to two of their own scan chains to the design. All pins on the BSCAN user logic macro are automatically supported. Note that the BSCAN macro must have the TDI, TCK, TMS and TDO pins connected to the TDI, TCK, TMS and TDO symbols (respectively). If boundary scan operation after configuration is required, the BSCAN macro must be instantiated. Refer to the User Logic section above for more details.

Boundary Scan - Before the Configuration Phase

Both the FPGA and the HardWire Array support the ability to perform boundary scan operations before, during, and after the configuration process. They differ in the way BSCAN is handled before and during configuration.

Prior to configuration, the FPGA is an unknown state. Refer to Figure 16A. It is impossible to predict the state of the internal nodes. At some point in time, the FPGA will begin it's initialization cycle and begin clearing its memory. By holding the INIT pin low, the customer may enter BSCAN operations. However, the contents of the FPGA are unknown since the initialization cycle may not have been completed.

The HardWire Array is a mask-programmed device, and therefore has a pre-defined state of every node. Refer to Figure 16B. This means that it is possible to read data back from the HardWire which is different than the same test on a FPGA prior to it's configuration. The solution is to make sure that the test program ignores the bits in the BSCAN data stream which are sampled from internal states. (e.g. Sample DR).

Boundary Scan - During the Configuration Phase

Only three operations are supported by the FPGA during configuration. These are SAMPLE/PRELOAD, BYPASS and CONFIGURATION.

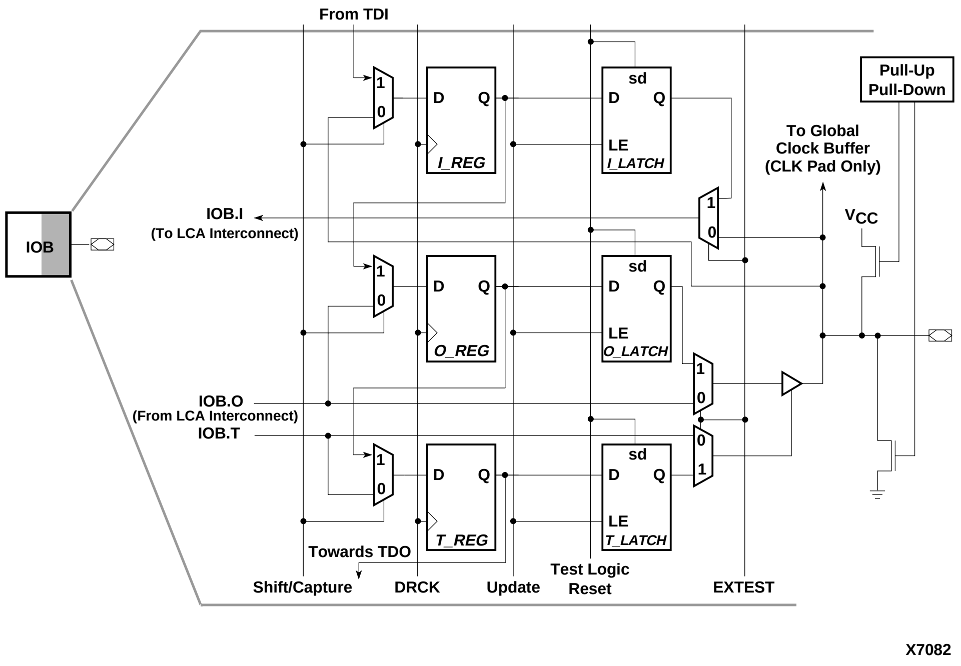


Figure 14. FPGA BSCAN

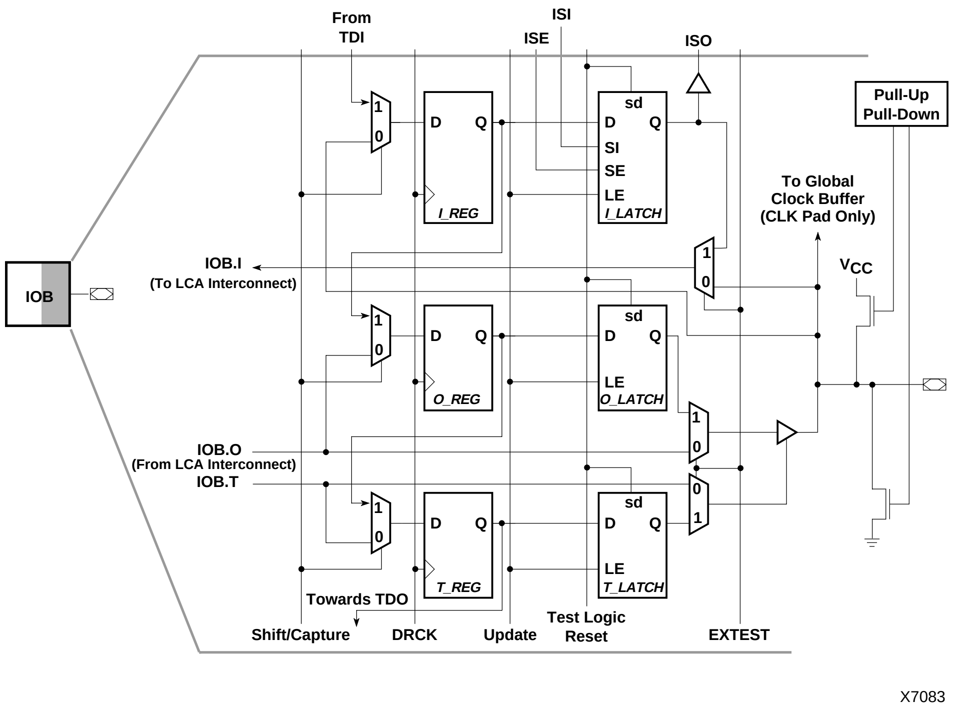


Figure 15. HardWire BSCAN Logic

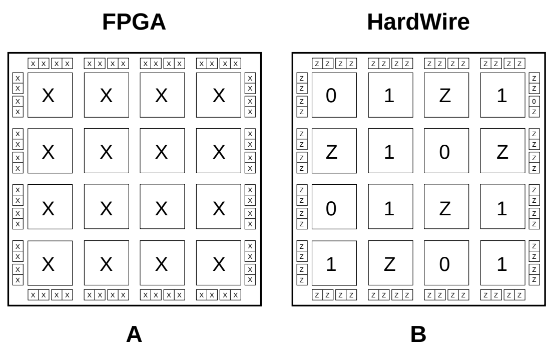
Note in Figure 17A that the BSCAN operations during configuration have all the I/Os held at Z. This condition will last during the entire configuration process. The HardWire Array does not support configuration emulation via BSCAN nor does it support the BSCAN CONFIGURATION instruction. However, the HardWire Array will have the I/Os held at Z until it reaches the end of the STARTUP phase.

The SAMPLE / PRELOAD instruction is not recommended since the exact state of the configuration will asynchronously change while sampling, resulting in incorrect data. The BYPASS instruction is permitted during the configuration phase but is not recommended. See Figure 17. The internal states may have been affected by BSCAN operations in the HardWire so the FPGA may not have the same state information because it has not yet been configured. See Figure 17A & 17B.

Boundary Scan - After the Configuration Phase

The HardWire Array and the FPGA behave identically once each has been configured, see Figures 18A and 18B. BSCAN operations are permitted after configuration only if the customer has instantiated the BSCAN user logic macro in their design and dedicated the TDI, TMS, TCK and TDO pins for BSCAN.

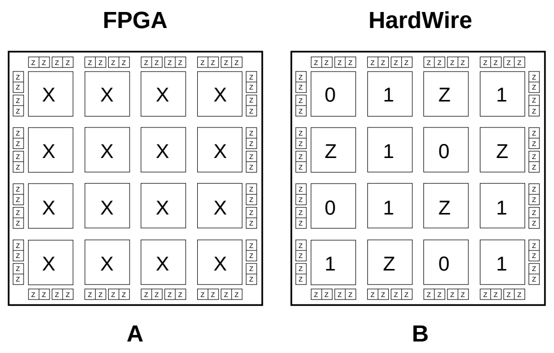
In order to ensure results of BSCAN tests in the FPGA are consistent with the HardWire Array, Xilinx recommends that only the nodes which are directly controllable via BSCAN be used for testing. This is due to the fact that any BSCAN testing done before or during configuration can affect the internal state of the HardWire Array, which is already “programmed” from power on.



INSTRUCTION	IR2	IR1	IR0
EXTEST	0	0	0
SAMPLE/PRELOAD	0	0	1
USER1	0	1	0
USER2	0	1	1
FULL_SCAN	1	1	0
RESERVED	1	0	1
I_SCAN	1	1	0
BYPASS	1	1	1

X7105

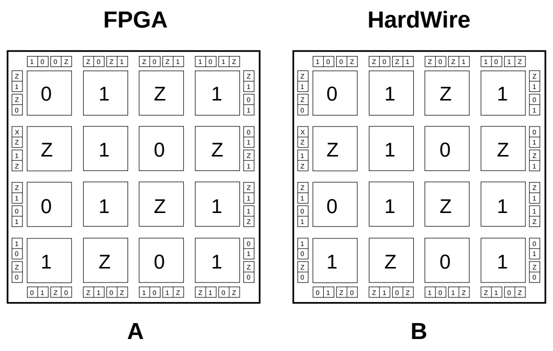
Figure 16. The Unknown State of the FPGA and HardWire Prior to Configuration



INSTRUCTION	IR2	IR1	IR0
SAMPLE/PRELOAD	0	0	0
NOT AVAILABLE	0	0	1
NOT AVAILABLE	0	1	0
NOT AVAILABLE	0	1	1
NOT AVAILABLE	1	1	0
RESERVED	1	0	1
NOT AVAILABLE	1	1	0
BYPASS	1	1	1

X7106

Figure 17. BSCAN Operations During Configuration



INSTRUCTION	IR2	IR1	IR0
EXTEST	0	0	0
SAMPLE/PRELOAD	0	0	1
USER1	0	1	0
USER2	0	1	1
FULL_SCAN	1	1	0
RESERVED	1	0	1
I_SCAN	1	1	0
BYPASS	1	1	1

X7107

Figure 18. BSCAN After Configuration