Migrating Spartan 6 Design to 7 Series & Beyond

WP02 – FPGA Design Considerations

Adiuvo Engineering

ADIUVO
ENGINEERING AND TRAINING, LTD.

Migration from Spartan-6 to 7 Series and Beyond

Abstract – This white paper will outline the challenges and mitigation strategies which can be implemented when migrating from a Xilinx Spartan-6 FPGA to a Xilinx 7 Series device or beyond.

Contents

List of Figures

Introduction

The semiconductor shortage is having significant impacts on the supply chain and this is especially true for older nodes such as the 45nm upon which the Spartan-6 device is fabricated. Although there are still challenges with 7 series and UltraScale/UltraScale+ deliveries, I am informed that more modern nodes exhibit an improved long-term supply situation.

In this white paper, we are going to understand the differences between the Spartan-6 and 7 Series architectures. We will also discuss how we can select the most appropriate migration device from the 7 Series range along with how to migrate the tool chain from ISE to Vivado. This white paper will also examine how best to migrate a range of designs from pure RTL-based designs to those which contain a significant element of IP and softcore microcontrollers such as MicroBlaze within the programmable logic fabric.

First introduced in 2009, the Spartan-6 family is based on a 45 nm process and provides developers within the standard LX version 3.8K and 147K logic cells, up to 576 IO, 180 DSP elements, and 268 18Kb Block RAMS. The transceiver enabled LXT versions provides the logic resources of the LX family and provide up to eight GTP transceivers and 1 PCI Express end point.

Both the LX and LXT range of devices provide hard integrated memory controllers which support DDR, DDR2, DDR3, and LPDDR with data rates up to 800 Mb/s.

Introduced in 2010, the 7 Series consist of five families of devices including the Virtex-7, Kintex-7, Artix-7, Zynq-7000, and Spartan-7. This range of families in the 7 Series provides developers sufficient capacity, capability, and performance to migrate a Spartan-6 device but also enables significant opportunity for future product enhancement.

Spartan-6 to 7 Series Architectural Differences

Logical Elements - The fundamental element of an FPGA is the logic cell. Both the Spartan-6 and 7 Series have a function generator which consists of a six input Look Up Table (LUT) with two associated flip flops. Several of these function generators and flip flop structures are combined to create a slice. Each slice contains eight function generators and 16 flip flops. Within 7 Series devices, there are two types of slices -- Slice_M and Slice_L. The LUT within the Slice_M can act as distributed memories or shift registers. This is not possible within Slice_L. Architecturally, two slices are combined to create a configurable logic block.

While the Slice_M and Slice_L are identical between the spartan-6 and 7 Series devices, Spartan-6 devices also have a Slice_X. Slice_X is the most basic logical structure of the three slice configurations. Functions that were implemented using a Slice_X can be easily accommodated within the Slice_L which are available within the 7 Series. Of course, retargeting of Slice_X to Slice_L is automatic in synthesis. As Slice_X provides only the most basic logic functionality, migration to Slice_L in 7 Series devices can result in performance improvement.

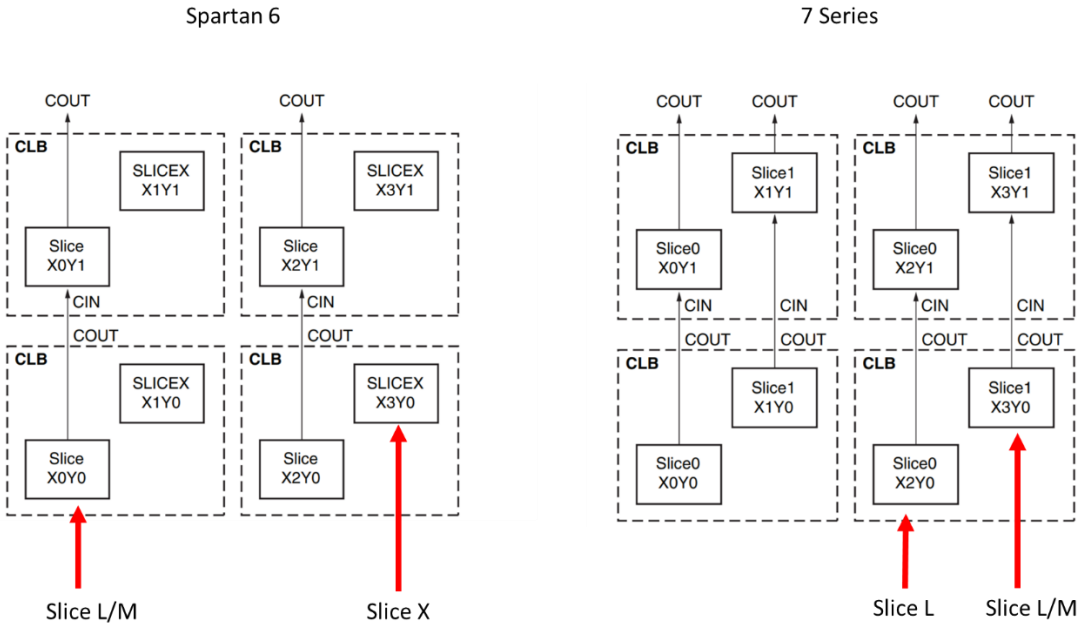

Figure 1 – Spartan-6 and 7 Series CLB Structure

Block RAM – One of the largest differences between Spartan-6 and 7 Series devices is in the Block Ram. Spartan-6 Block RAMs are arranged as 18Kb blocks which can be configured as two 9Kb memories. In comparison, 7 Series devices provide 36Kb blocks which can be configured as two 18Kb memories.

For most applications, the re-targeting should be automatic during synthesis. If, for example, there are many smaller memories like <9Kb, then a larger 7 Series device capable of supporting that memory granularity may be required.

However, 7 Series Block RAM offers the designer several significant improvements which can be very useful. This includes capabilities like providing build in FIFO, cascading Block RAMs, and built-in error correction codes.

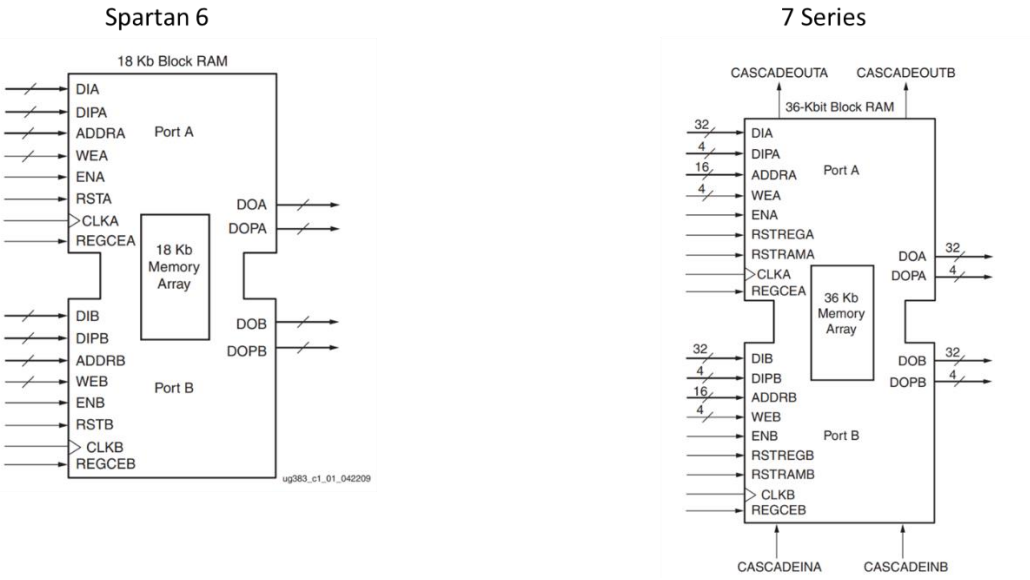

Figure 2 – Spartan-6 and 7 Series BRAM Blocks

DSP – Being able to leverage the parallelism of programmable logic to implement filters, FFTs, and arithmetic algorithms is a key benefit of FPGA implementation. Both Spartan-6 and Spartan-7 provide dedicated DSP elements which enable the developer to implement multiply accumulate functions. In the Spartan-6, the developer is provided with a DSP48A1 which provides 18x18 signed multiplication while 7 Series devices play DSP48E1 which implements a 25x18 signed multiply. Architecturally, the DSP48E1 provided in 7 Series devices also enables the implementation of an Algorithmic Logic Unit (ALU) and enables support of Single Instruction Multiple Data (SIMD) mode which allows increased throughput. The DSP48E1 is also capable of implementing pattern detection and 17-bit shifter structures as required by the application.

Mapping to the DSP48E1 from the DSP48A1 should be mostly automatic by the Vivado synthesis engine. However, to implement advanced modes such as SIMD, language templates are provided in the Vivado editor to allow ease of implementation to achieve the best performance.

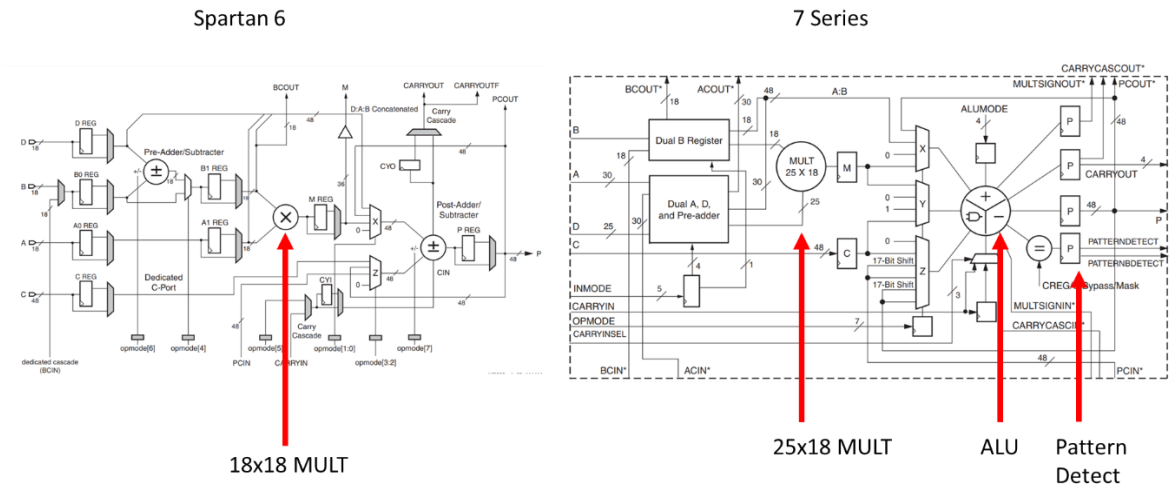

Figure 3 – Spartan-6 DSP48A1 and 7 Series DSP48E1 Structures

Clock – The clocking architecture of 7 Series devices is significantly simpler than that previously provided in the in Spartan-6 devices. Clocking in Spartan-6 has different buffer types which determined connections and connectivity, for example, BUFG, BUFH, and BUFIO2 or BUFPLL. Resource wise, the Spartan-6 provides the developer with Digital Clock Managers (DCM) and Phase Locked Loops (PLL) clocking resources.

7 Series devices provide the user with a simpler clocking architecture which, along with flexibility, provides a significant improvement in performance. Within a 7 Series Clock Management Tile (CMT), MMCM and PLLs are provided and associated with each IO Bank.

Spartan-6 designs which use either a PLL or DCM_SP will migrate to a MMCM in a 7 Series device. We must, of course, make sure we set the necessary parameters such as clock in period correctly. While most other buffers (e.g., BUFH and BUFG) will migrate automatically during synthesis, buffers that are specific to Spartan-6 like BUFIO2 will need to be migrated in the design if directly instantiated in the design.

Memory Interfaces – Being able to interface with high-performance external memories is critical for many designs. Both Spartan-6 and 7 Series devices provide the user with the ability to achieve this. However, the Spartan-6 implementation uses an integrated memory block whereas 7 Series devices use a Soft IP core to implement the memory controller where only the memory

PHY is hardened. This provides 7 Series devices with a more flexible approach to IO allocation and design which can be critical when working with complex PCB designs. 7 Series Memory Interface Controller can support DDR3, DDR3L, DDR2, and LPDDR2 providing maximum flexibility in selecting the memory.

Transceivers and PCIe – Spartan-6 LXT devices provide the developer with multi-gigabit transceivers in the GTP at a maximum speed of 3.2 Gb/s. 7 Series devices which support transceivers can support higher data rates or up to 6/25 Gb/s. 7 Series GTPs provide a quad implementation per tile compared to the dual-tile GTP in Spartan-6 architecture. The GTP clock has also evolved in 7 Series devices, enabling TX and RX to be independently clocked from any PLL. This is different than in the Spartan-6 GTP dual tile where the TX and RX must use the same clock. 7 Series GTPs also provide new capabilities such as Continuous Time Linear Equalization (CTLE) with auto adaptation and post equalization eye scan.

When it comes to implementing PCIe, 7 Series devices support both PCIe Gen 1 and Gen 2 in the Artix-7 range. This allows the user to benefit from greater performance as bandwidth is significantly increased if desired.

Additional New Features – As would be expected, the 7 Series range also introduced new features which provide benefits to the developer. The first of these is the XADC which is a 1 MSPS ADC which enables the developer to observe the internal supply voltages and die temperature. This can be very useful when implementing self-test and anti-tamper features. The XADC is also able to quantize 16 external differential signals, removing the need for additional low-speed ADCs used in board monitoring. Along with the ability to provide bitstream security, this is also enhanced in the XADC with the provision of AES256 CBC encryption and SHA-256 authentication.

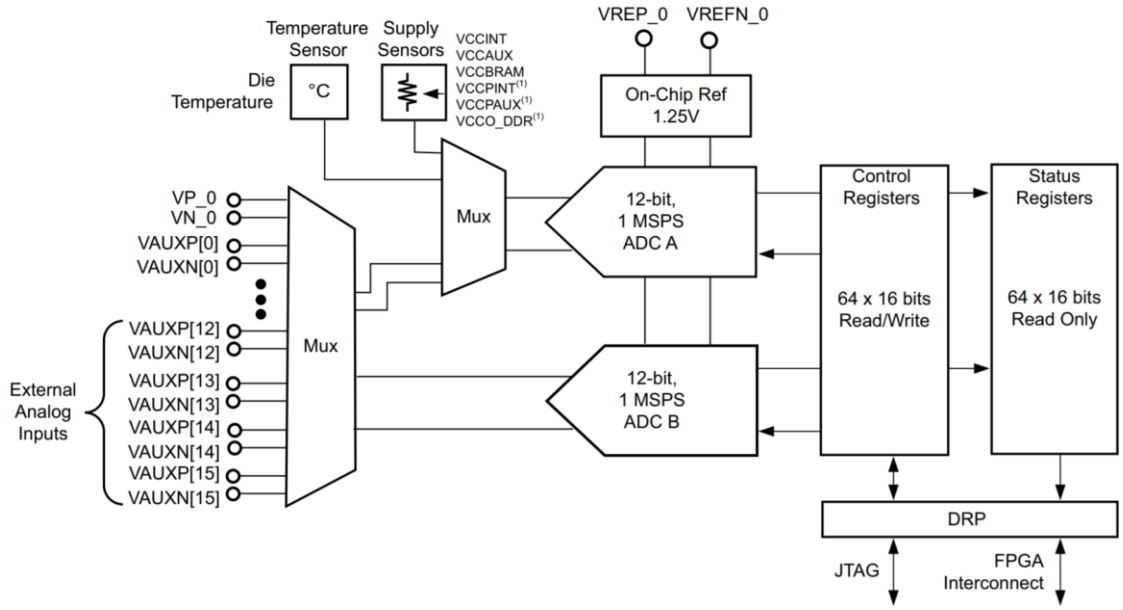

Figure 4 – XADC Structure 7 Series

Selecting the Most Appropriate Device

Each Spartan-6 design considered for migration is unique and the developer must think through each case individually. At a high level, it is possible to consider device selection from the flow chart below. This flow chart is based upon three key decisions points:

1. MicroBlaze – If a MicroBlaze is used in the design, we need to determine if we want to continue using the MicroBlaze or migrate to a higher performing A9 or A53 in the Zynq SoC or Zynq MPSoC families. Migration from MicroBlaze to Arm processor cores is something we will look at in detail in another blog. However, the Xilinx Vitis framework and BSP generation does a lot of the heavy lifting for us.

2. Transceivers Used – We need to determine if high speed multi-gigabit transceivers are being used as part of the application being migrated.

3. Size of the Spartan-6 Device – We can fit the device in a Spartan-7 device if the Spartan-6 device does not use transceivers and is a smaller device than the XCS6LX100. However, we need to consider a device from the Artix family if the Spartan-6 device for migration is larger.

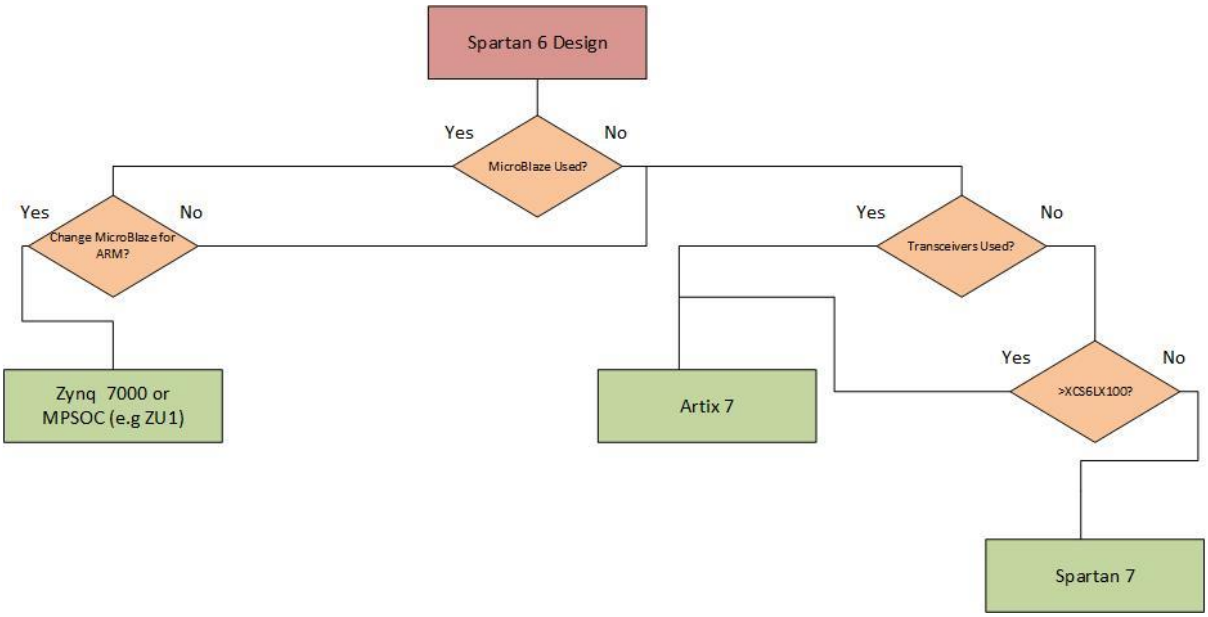

Figure 5 – Device Selection Spartan-6 to 7 Series flow chart

These decision points are high-level decision points intended to guide potential device selection. Once the recommended family has been selected, the engineer performing the migration needs to carefully look through and consider additional salient points of the design and consider the following to identify the actual target migration device:

1. Number of Block RAMS, DSP, clock management tiles, required.

2. Maximum number of IO pins required along with the number of differential pins required.

3. IO standards required – 7 Series IO is provided in two classes: High Range (HR) and High Performance. HR banks support IO standards of 3v3 and 2v5, while HP banks support IO standards up to 1v8 and are intended to support higher-performance interfaces. We also need to identify the specialist IO structures used in the Spartan-6

mitigation. Crucially ODELAY is only available within HP IO banks which means the engineer must consider selecting a device in the Kintex-7 range.

4. Company supply chain preference – It might be sensible to select a slightly larger or different device if the design will migrate to align with company supply chain preferences and the purchase of common components which can be used across several projects within the company.

While migration from Spartan-6 to a 7 Series device comes with overheads, it also comes with opportunity. Depending upon the device selected for migration, a larger device or higher performance device could be selected. For example, a Kintex-7 part can be used in place of a Artix-7 or a MPSoC in place of a Zynq-7000 SoC. These selections provide the resources to support future product roadmap enhancements that may have previously been limited due to device utilization constraints. This is also the case when using a Zynq SoC or Zynq MPSoC in place of a MicroBlaze processor. This provides easy support for a range of now commonly used interfaces such as USB and Gigabit Ethernet as well as advanced solutions such as SATA or DisplayPort.

Tool Chain Changes

The major change between Spartan-6 implementations and 7 Series implementations is the tool chain used for development and implementation. Spartan-6 devices use the ISE, EDK, PlanAhead, and SDK tool chain while 7 Series use the Vivado and Vitis tool chains. Vivado is a quantum leap in capability compared to the older ISE tool chain. Vivado enables developers to work with pure RTL designs and leverage a large inbuilt IP Library using IP integrator. IP Integrator is ideal for creating embedded system designs which contain processors within either the processing system of a heterogeneous System on Chip or softcore implemented in logic such as MicroBlaze.

Vivado provides developers with end-to-end capability and includes synthesis, place and route, bit generation, and hardware programming and debugging. Embedded software development takes place within Vitis, which is an eclipse-based software development environment.

Vivado IP is standardized around the Arm eXtensible Interface (AXI) and provides three classes of interface:

- AXI MM – Full Memory Mapped interconnect capable of supporting burst access to increase throughput. This is ideal for Direct Memory Transfer.
- AXI Lite – Reduced single-beat memory mapped interface intended for configuration and control of IP within the programmable logic.
- AXI Stream – Unidirectional point to point high-speed data channel with no address component.

Migration between ISE and Vivado for pure RTL design can be as simple as creating a new project, selecting the targeted device, and importing the RTL. Of course, if there are any IP blocks specific to the spartan-6 instantiated and not inferred, the RTL must be updated to remove them. However, the RTL design should generally be implemented with minor modifications.

The main difference between the ISE and Vivado is in how constraints are used. ISE uses the UCF format while Vivado uses XDC which are based on the SDC format. Within our design, we can use constraints to define timing, IO standards, and control the implementation and placement of design elements.

Conversion from UCF to XDC is straightforward and well documented within UG911. An example of timing constraint and IO constraint migration is presented in the RTL Migration Example chapter.

To help get started creating XDC constraints for timing, IO, and placement constraints, Vivado provides several wizards which are enabled under the synthesis and implementation views. These are helpful in walking you through constraint creation.

One of the most significant challenges developers face is achieving timing closure. Vivado provides the developer with significant design analysis and reporting for greater design insight.

Inference vs Instantiation – One of the challenges of migration from the Spartan-6 to the 7 series is the migration of instantiated modules, for example, BRAM or DSP elements. Initially the developer may consider implementing a similar instantiation in the migrated design of the 7 series primitive.

However, to attain the most portable and flexible design, it is better to update the RTL to infer the desired primitive. Just as we do not instantiate CLBs and define their logic function and connections, nor should we in RTL design when it comes to working with BRAMs, DSPs, and

other functions. Of course, this could bring about iterations to ensure the RTL description is correctly interpreted by the synthesis tool and the correct primitive used.

If we want to infer specific primitives in synthesis within Vivado, we can leverage the language templates available within the Vivado Editor. Language templates also enable the developer to insert the coding structure which will be mapped by the synthesis tool to the desired primitive. Templates are provided in both VHDL and Verilog regardless of the developers preferred language.

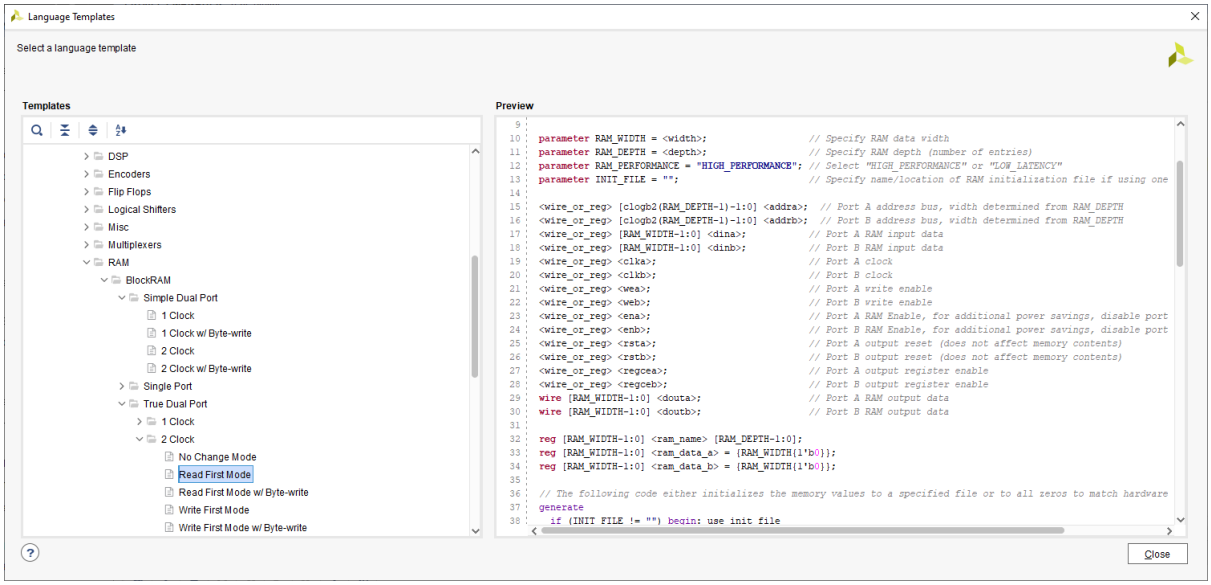

Figure 6 – Vivado Language Template

Whether migrating a design or starting a new one from scratch, it is best to use inference over instantiation whenever possible to ensure the design is portable and does not encounter future migration issues.

Design Analysis Report (DAR) - Enables the user to understand the design challenges (e.g., congestion) and make changes to the design or constraints. The design analysis report will provide information on the following:

- Timing – Provides information on the timing and physical characteristics of timing paths.
- Complexity – Provides information on routing complexity and LUT distribution.
- Congestion – Provides information on routing congestion.

Quality of Result Assessment (QoRA) – Provides information on the design which can be used by the developer to understand its complexity and if it can be implemented and achieve timing closure. Along with a detailed report, the QoRA will present a design score which indicates the probably of the design implementing with no issues.

Table 1 – Vivado QoRA Scores

SCORE	MEANING	CORRECTIVE ACTION
1	Design will not implement	Redesign RTL / HLS modules
2	Design will implement timing problems	Review constraints & RTL HLS
3	Design Runs have a small chance of success	Use QoR suggestions, review clocking, ML strategies
4	Design should meet timing if directives used	Use QoR suggestions, ML strategies
5	Design will implement without timing issues	Run Implementation

In addition to the result assessment, the tool will also make Quality of Result Suggestions (QoRS) which can, in some instances, enable suggestions to be automatically applied once the flow is rerun.

Clock Interaction and Clock Domain Crossing Reports – Vivado provides the developer with a greater understanding of clock interaction and clock domain crossing issues. The CDC report will provide developers with a greater understanding of the CDC paths within the design. This report is available following the completion of synthesis.

The clock interaction report enables an easy viewing of the clocks in the design and their interaction as well as outlines the relationships between the clocks. This enables a system-level view of how clocks interact and guide the creation of necessary timing relationship constraints.

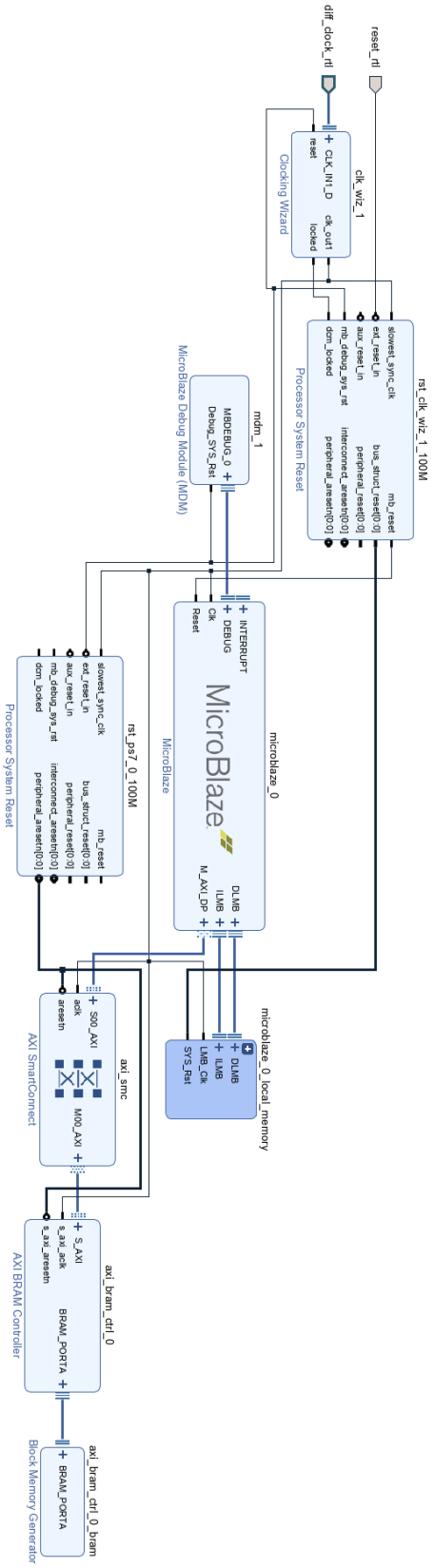

Figure 7 – Vivado IP Integrator MicroBlaze Block Diagram

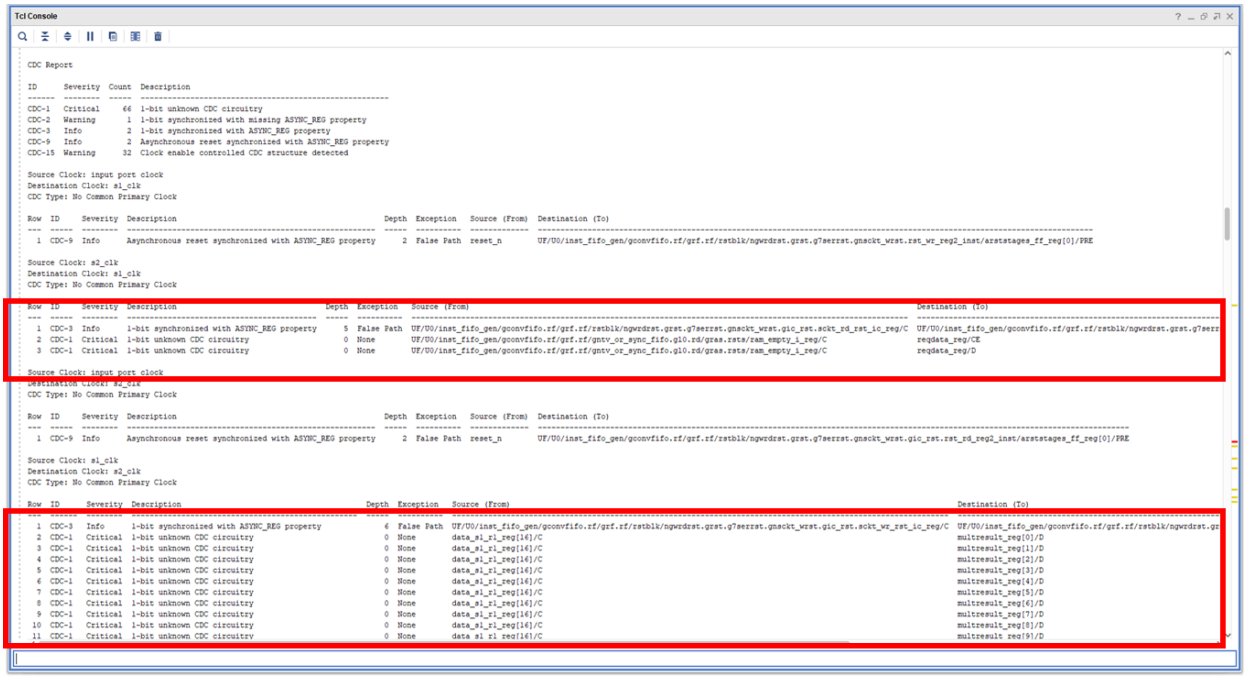

Figure 8- Vivado Clock Domain Crossing Report

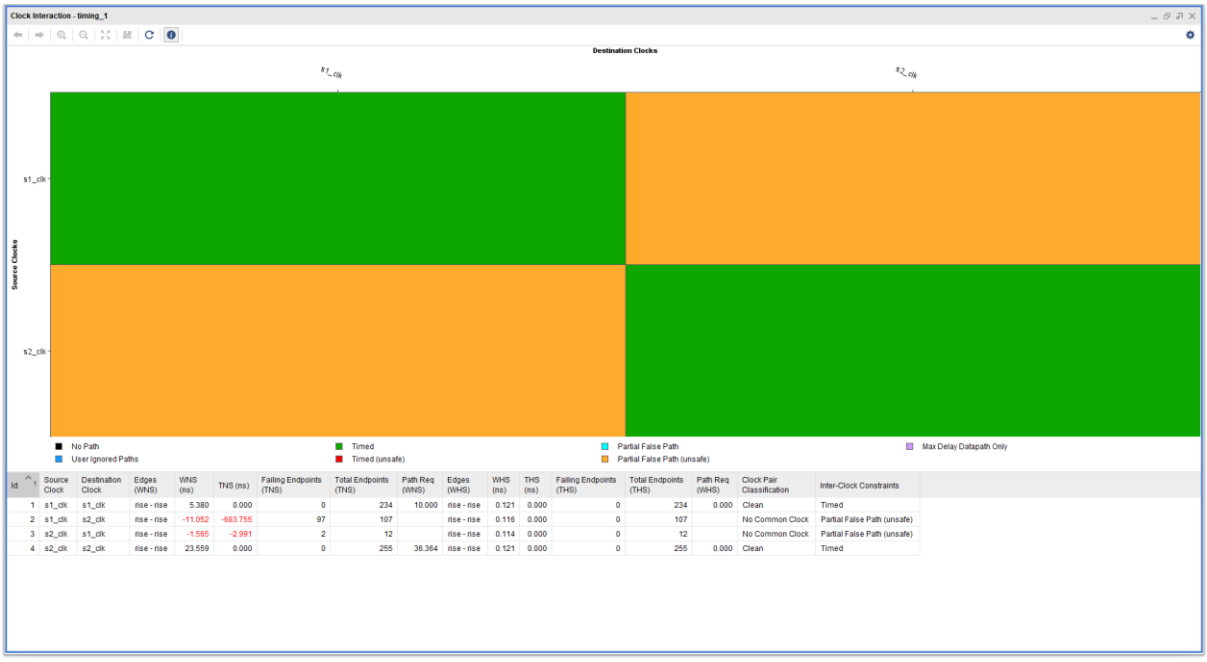

Figure 9- Vivado Clock Interaction Report

To understand the migration complexity from ISE to Vivado, the following flow chart can be used to scope migration steps and complexity.

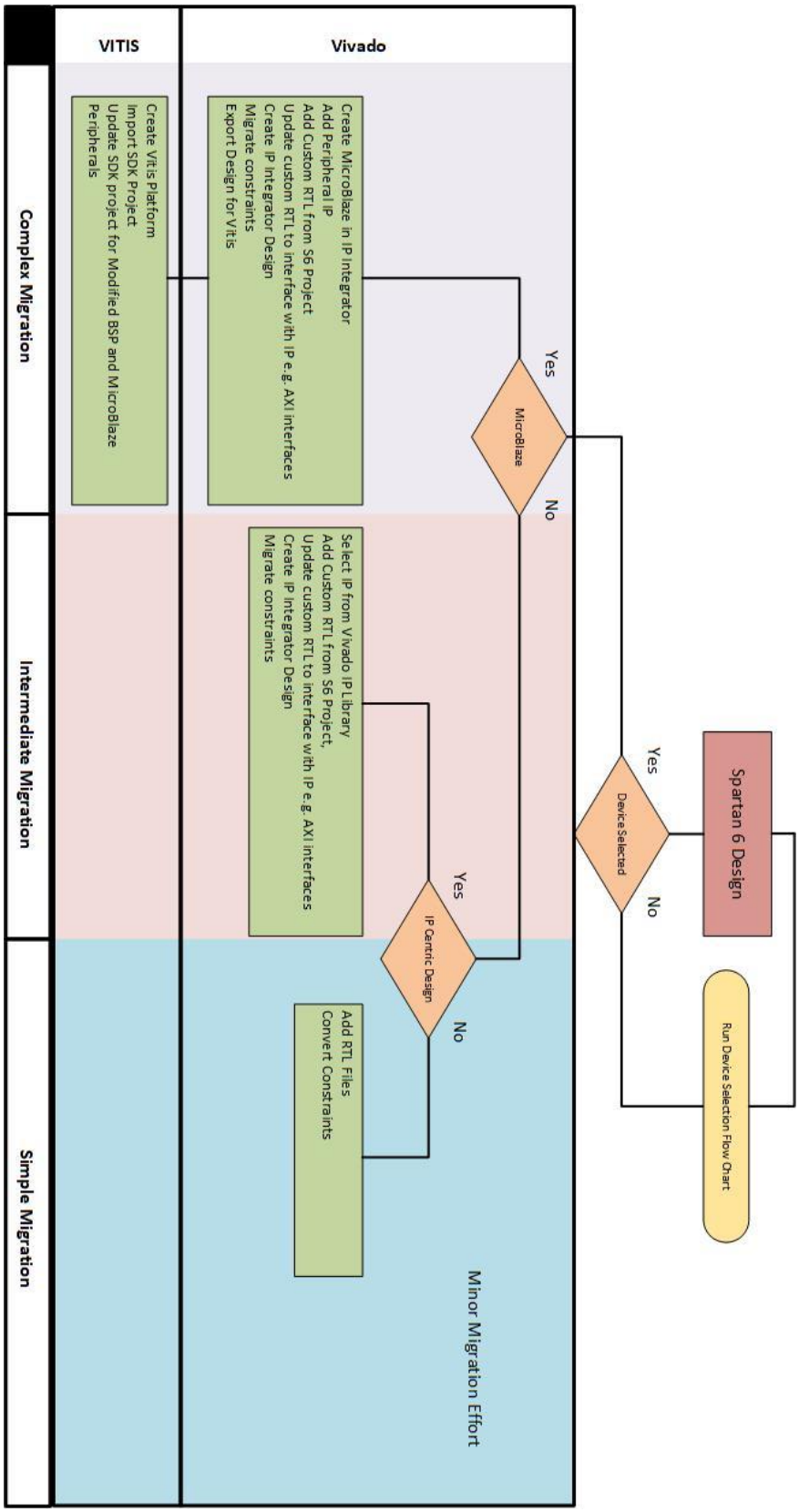

Figure 10 – Design Migration Flow Chart ISE to Vivado and VITIS

RTL Migration Example

One of the primary interfaces used in space electronics is SpaceWire. This is a differential high-speed communications protocol like firewire so both the flight FPGA and test equipment needs to implement SpaceWire. Several SpaceWire test equipment units use the Spartan-6 to implement the SpaceWire Codec. One example which is available open source (BSD License) is the 4 Links 400 Mbps Codec. This provides high-speed SpaceWire communication targeted to a Spartan-6 LX45T.

As this is an RTL project, a new Vivado project can be created that targets the desired device. In this case, I created a project targeting a Spartan-7 xc7s50. However, the UCF which defines both the pin out and the timing requirements will need migration.

The main conversion here is from the UCF used in ISE to the XDC used in Vivado. Conversion is straightforward. The first element we need to convert is the timing constraints. In the UCF file, the timing constraints are defined by the TNM_NET and TIMESPEC constraints. For this project they are defined as follows:

 # Clock timing contraints
NET "iclock" TNM_NET = iclock;
TIMESPEC TS_iclock = PERIOD "iclock" 8 ns HIGH 50%;

In this definition, signal iclock is the output from an IBUFDS. Downstream, the design has DCM and global buffers but iclock is root clock.

In Vivado, we would use the XDC create clock command

create_clock -period 8.000 -name CLK_125MHz_p -waveform {0.000 4.000} [get_ports CLK_125MHz_p]

When I migrated the design to Vivado, I accounted for the slight difference in how Vivado analyzes clocks compared to ISE. Vivado assigns time zero to the point at which the clock is defined and ignores all delays upstream of the declaration point. As such, clocks should be defined at the primary input pins. If a clock is created in a design logic (e.g., counter, DCM etc.), that should be defined using the create generated clock constraint.

If you are concerned about which clock constraint to use, you don't need to write the constraint by hand in Vivado. You can use the Timing Wizard in Vivado (under the tools menu) once the synthesis has been completed. The timing wizard will walk you through the creation of a target constraint file (XDC) while the definition of constraints will save to the target constraints file.

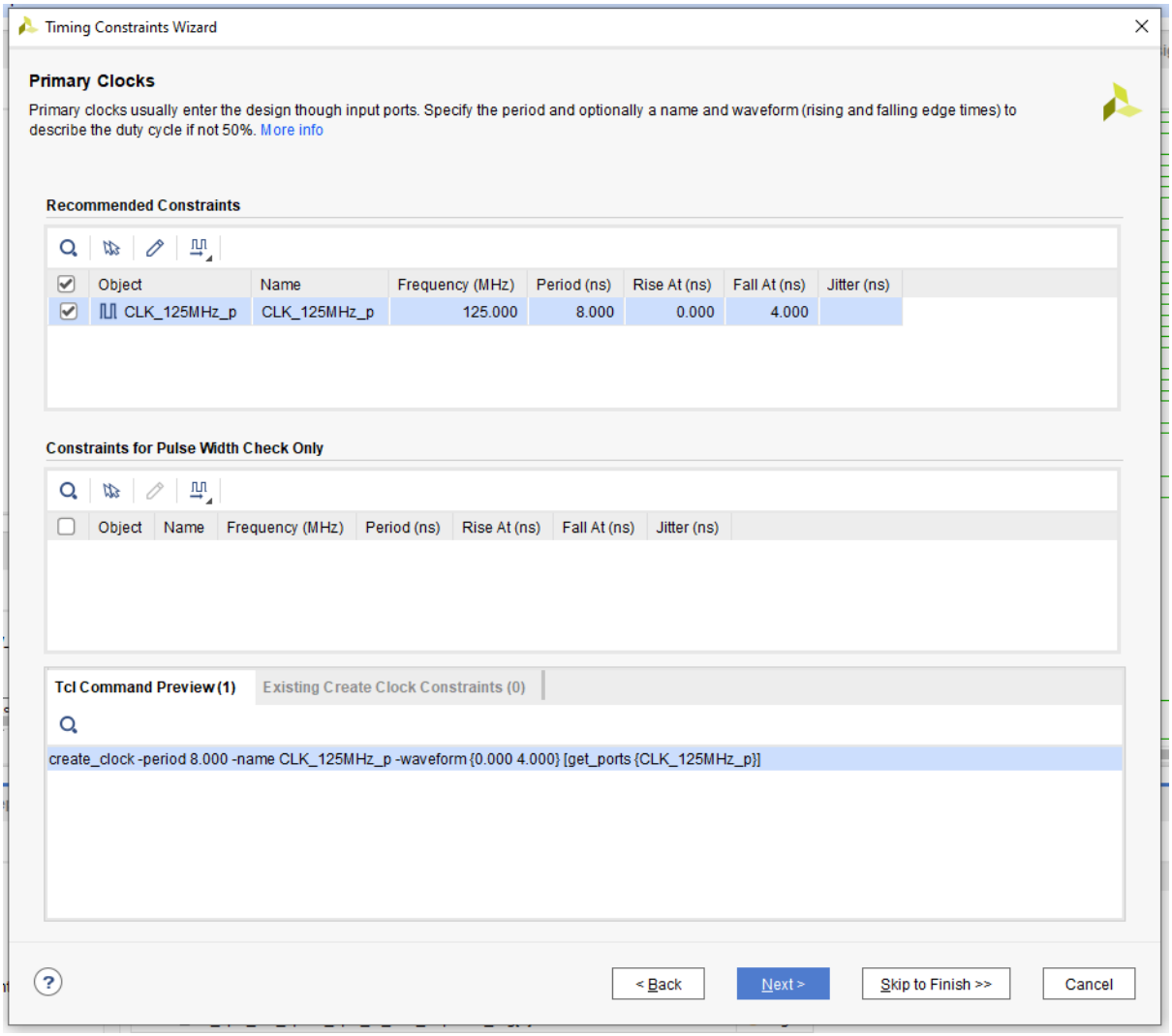

Figure 11 – Vivado Constraints Wizard

With the timing issues addressed, the final stage of the migration of this design is to port the IO constraints from the UCF to the XDC file. The current definition in the UCF is as follows:

NET Din2_p LOC = "M16" | IOSTANDARD = LVDS_33 | DIFF_TERM = "TRUE";
NET Din2_n LOC = "M18" | IOSTANDARD = LVDS_33 | DIFF_TERM = "TRUE";
NET Sin2_p LOC = "L17" | IOSTANDARD = LVDS_33 | DIFF_TERM = "TRUE";
NET Sin2_n LOC = "L18" | IOSTANDARD = LVDS_33 | DIFF_TERM = "TRUE";

Again, we need to convert these to a XDC format suitable for use with Vivado. If desired, we can write a XDC file by hand in the existing XDC file created for the project.

set_property IOSTANDARD LVDS_25 [get_ports Din1_p]
set_property IOSTANDARD LVDS_25 [get_ports Din1_n]
set_property IOSTANDARD LVDS_25 [get_ports Din2_p]
set_property IOSTANDARD LVDS_25 [get_ports Din2_n]
set_property PACKAGE_PIN A3 [get_ports Din1_p]
set_property PACKAGE_PIN A5 [get_ports Din2_p]

Alternatively, can use the I/O Ports tab in the synthesis view to define the pin allocation, IO standard, and any other IO required features if you are unsure of the exact format.. Like the timing information, this will be saved to the target XDC file which can be inspected to understand the XDC syntax.

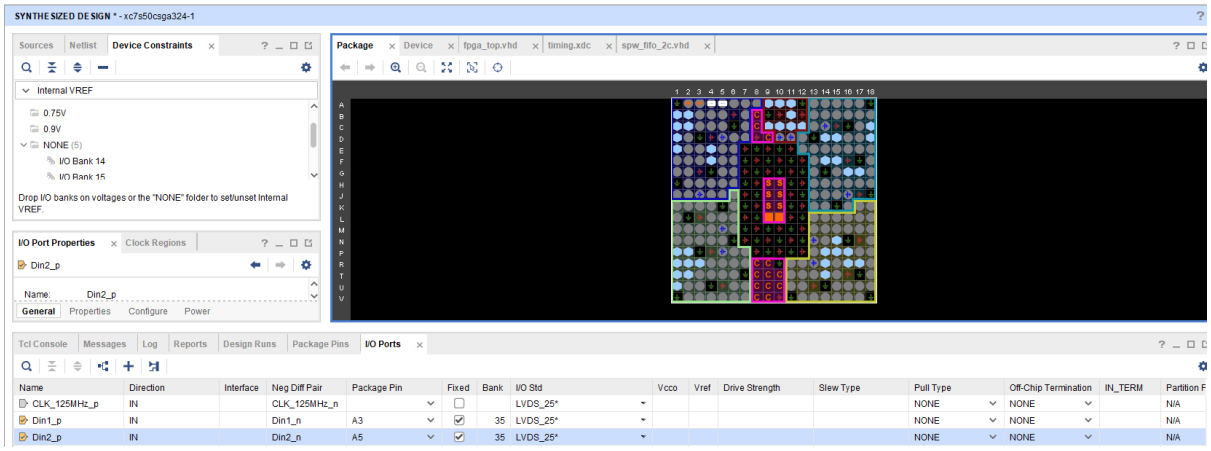

Figure 12 – Vivado IO Assignment Wizard

We can now build and implement the project and generate the bit stream that is ready for porting to the new target 7 series device. Of course, this is a relatively straightforward port of a straightforward RTL design.

MicroBlaze Migration

Many Spartan-6 designs include a MicroBlaze softcore processor which performs sequential processing implementing UART communications, network stacks, and human machine interfacing.

Within the Spartan-6 MicroBlaze ecosystem, solutions are developed using Xilinx Platform Studio to create the processor and the Software Development Kit (SDK) to create the application software. While XPS does provide the ability to create MicroBlaze solutions using AXI interconnects, most Spartan-6 MicroBlaze designs are implemented using older interface standards which are the Processor Local Bus (PLB) and Local Memory Bus (LMB). These busses are used to connect the MicroBlaze to peripheral interfaces like UART and GPIO, along with memories such as on chip BRAM and external DDR.

Often

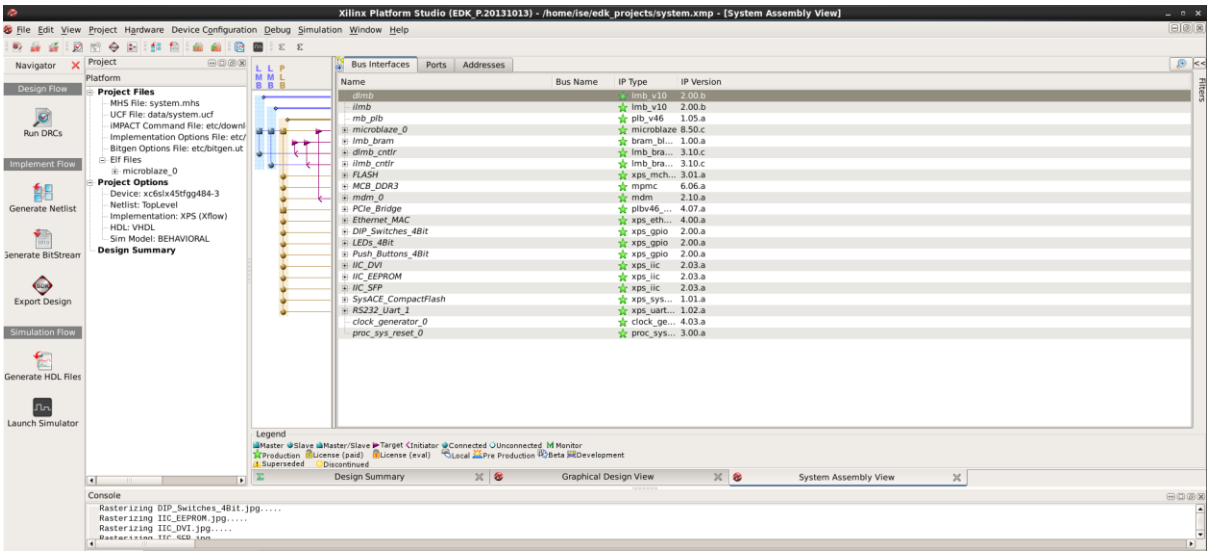

Figure 13 – XPS MicroBlaze Creation for Spartan-6 Implementation

Of course, many MicroBlaze solutions contain custom IP cores which perform application-specific functions. This custom peripheral needs to be updated to support AXI interfacing if not already supported. The updated IP can be reused in the updated Vivado project and connected to the MicroBlaze processor.

Migrating a Spartan-6 design that contains a MicroBlaze to a 7 Series device requires more porting than a pure RTL design but also offers potentially more capability and design reuse.

To get started porting a Spartan-6 MicroBlaze design to a 7 Series device, we must first recreate the MicroBlaze design in the Vivado IP Integrator. This enables a MicroBlaze processor to be implemented and connected to the necessary library IP in an integrated environment. The Vivado IP Integrator is not only board aware, enabling peripherals to be configured for specific board configurations such as DDR memories, UARTs, PCIe interfaces, but it's also able to automate connections between AXI interfaces to accelerate solution development.

When dealing with MicroBlaze applications, engineers must also address a change in software development tool from SDK to Vitis. Vivado will provide the Xilinx Shell Archive (XSA) which enables Vitis to create a platform including a board support package of APIs to enable SW developers to interface and work with the IP peripherals within the design.

The software application from SDK must be imported into Vitis and updated to support the new IP peripherals. Vitis makes upgrading easier by allowing an exported SDK project to be imported into Vitis.

ISE System Generator Designs

Some designs in Spartan-6 may be developed using ISE System Generator. Like MicroBlaze solutions, these designs are best migrated by recreating the design in Model Composer in Vivado. Like the MicroBlaze solution, there are several reasons for this including a change in available IP cores and migrating to AXI interfacing on IP blocks.

The easier migration path is therefore recreating where the ISE System Generator design is used as a reference design to support the creation in Model Composer.

Wrap Up

Developers needing to convert a design from Spartan-6 to 7 Series devices may at first be daunted but as outlined in this white paper, the steps taken to select a suitable migration device and migrate the design are straightforward and achievable.

References

The following references may be of assistance in migrating from Spartan 6 to 7 Series and beyond devices.

1. www.adiuvoengineering.com
2. UG911: ISE to Vivado Design Suite Migration Guide
3. UG429: 7 Series Migration Methodology Guide
4. UG1026: UltraScale Architecture Migration Methodology Guide
5. UG904: Vivado User Guide - Implementation
6. UG949: UltraFast™ Design Methodology Guide
7. XAPP1311: Hot Swapping with FPGAs
8. WP484: DDR2/DDR3 Low-Cost PCB Design Guidelines for Artix-7 and Spartan-7 FPGAs
9. XAPP1313: Spartan-7 FPGA Configuration with SPI Flash and Bank 14 at 1.35V
10. XAPP1286: 7 Series FPGAs Gen2 Integrated Block for PCIe to AXI4-Lite Bridge
11. WP473: Software Migration to 64-bit ARM Heterogeneous Platforms
12. WP470: Unleash the Unparalleled Power and Flexibility of Zynq UltraScale+ MPSoCs
13. WP482: Managing Power and Performance with the Zynq UltraScale+ MPSoC
14. 53109: Vivado - Are Spartan-6, Virtex-6 and older devices supported in the Vivado design tools?
15. 44225: 7 Series Power Sequencing - Hot-swap/-plug capability
16. 40603: MIG 7 Series FPGAs DDR3/DDR2 - Clocking Guidelines
17. 43989: 7 Series, UltraScale, UltraScale+ FPGAs and MPSoC devices - LVDS_33, LVDS_25, LVDS_18, LVDS inputs and outputs for High Range (HR) and High Performance (HP) I/O banks
18. 62332: 14.7 ISE - Artix-7 and Zynq device support clarification