System Migration and Comparison Guide: Arria10 SoC SOM to Terasic Titan S10 SOM

This document outlines the system migration and comparison from a widely adopted **Arria10 SoC SOM** to the **Terasic Titan S10 SOM**. Both modules are powerful System-on-Modules (SOMs) featuring FPGA and HPS (Hard Processor System) capabilities. However, transitioning from the Altera Arria 10 to the Altera Stratix 10 platform involves significant architectural enhancements in hardware, processor cores, high-speed transceivers, and software boot flows.

1. Core Specifications Comparison (Hardware, FPGA, HPS)

Specification	Arria10 SoC SOM	Terasic Titan S10 SOM	Migration Assessment & Differences
FPGA Device	Altera Arria 10 SX (10AS066H2F34I1HG)	Altera Stratix 10 SX (1SX110HN3F43I2VG)	**Major Upgrade**: Logic Elements (LEs) increased from 660K to 1100K; significant increase in multipliers and on-chip memory.
HPS Processor	32-bit Dual-core ARM Cortex-A9 MPCore	64-bit Quad-core ARM Cortex-A53 MPCore	**Architecture Upgrade**: Migrating from 32-bit to 64-bit; core count increased from 2 to 4. Requires recompiling Linux OS and applications.
Dimensions	83.1 mm x 95 mm	86 mm x 98 mm	**Slight Change**: Titan S10 is slightly wider in order to support the FMC+ connector.
Expansion Connectors	2x FMC Connectors	2x FMC+ Connectors; FMC version is also available.	**Interface Upgrade**: Titan S10 utilizes higher-bandwidth FMC+ connectors, providing more I/O and Transceiver lanes.
Transceivers (XCVR)	24x XCVR (Up to 10 Gbps)	32x XCVR (Up to 12.5 Gbps, 16 pairs per FMC+ connector)	**Bandwidth Increase**: More channels available with per-channel speeds increased from 10 Gbps to 12.5 Gbps.

Specification	Arria10 SoC SOM	Terasic Titan S10 SOM	Migration Assessment & Differences
Main Memory (DDR4)	FPGA: 4GB 32-bit DDR4 HPS: 4GB 32-bit DDR4	FPGA: 4GB 64-bit DDR4 HPS: 4GB 64-bit DDR4	Capacity remains the same, but Titan S10 features two 4GB 64-bit banks. The HPS 4GB bank can be shared between HPS and FPGA (via HPS EMIF).
Storage	32GB eMMC,512Mb QSPI Flash	32GB eMMC, 512Mb QSPI Flash	Both have eMMC for OS storage and QSPI for FPGA configuration. High hardware compatibility.
Ethernet (GbE)	2x Gigabit Ethernet PHY (Both are KSZ9031RNXCA, **directly connected to HPS via RGMII**)	2x Gigabit Ethernet PHY (**Port A**: KSZ9031RNX, **directly connected to HPS via RGMII Port B**: KSZ9031MNXIC, **connected to FPGA Fabric via GMII**)	**Major Architecture Change**: Both Arria 10 SOM ports connect directly to the HPS for system use. Titan S10's second port (Port B) connects through the FPGA side; **HPS peripheral configuration would route this internal to the HPS**, or it can be used as a dedicated hardware NIC on the FPGA.
USB Interface	USB 3.0 (Cypress FX3: 8-bit GPIF II) USB 2.0 OTG	USB 3.0 (Cypress FX3: 32-bit GPIF II) USB 2.0 OTG	**Significant Bandwidth Upgrade but Design Modification Required**: Although both platforms use the Cypress FX3 controller, the Arria 10 SOM is limited by FPGA I/O constraints and can only support an 8-bit GPIF II data bus ; whereas the Titan S10 fully unlocks performance, supporting a 32-bit GPIF II data bus.
System Management	MAX 10 CPLD, PMBus support	MAX 10 CPLD (BMC), PMBus support	Both use a MAX 10 device for board management (voltage, temperature, fan control).

2. Hardware & Interface Migration from Arria 10 SOM

2.1 Core Specifications Comparison (Power Supply)

Specification	Arria 10 SoC SOM	Terasic Titan S10 SOM	Migration Assessment & Differences
Main Power Input	Single **+12V** DC (+/- 5%) input. Minimum current: 3.8A. Maximum current: 8.7A (~104W, with 2 FMCs at full load).	Single **+12V** DC (+/- 5%) input. Minimum current: 4.16A. Maximum current: 8.33A (~100W).	**Highly Compatible**: Both require a single 12V power supply from the carrier board, with maximum power consumption around 100W. Existing carrier board power designs can be reused if they provide at least 12V/8.5A.
RTC / Security Backup Battery	Requires +1.8V or 1.55V battery power from the carrier board for the RTC.	Requires VBAT (e.g., 1.55V) battery power from the carrier board.	**Compatible**: Both require a battery holder on the carrier board to provide power for the RTC and secure key storage.

2.2 Carrier Board Connectors

- Arria 10 SOM uses standard VITA 57.1 two FMC connectors (HPC and custom LPC).
- Titan S10 upgrades to higher pin count and bandwidth **FMC+** connectors. The stack height between the SOM and carrier board changes from 7mm (Arria 10 SOM) to 8.5mm (Titan S10). The carrier board must be upgraded to support the FMC+ version.

2.3 High-Speed Signals & I/O (XCVR & LVDS)

- **Strict Grounding Requirements for Unused Transceiver Pins (Critical)**
  In the Stratix 10 FPGA design, unused high-speed signal pins have more stringent physical layout requirements compared to Arria 10 FPGA:
  - **Data Channels**:
    All unused transceiver receiver pins (GXB_RXp) must be directly connected to GND, while all unused transmitter pins (GXB_TXp) must be left floating.
  - **Dedicated Via Requirement for REFCLK_GXB**:
    For unused transceiver reference clock input pins (REFCLK_GXB), they must be directly connected to GND.
- **Pre-Layout LVDS Pin Assignment Validation (Mandatory)**

  When migrating your project to the Titan S10 SOM (based on the Stratix 10 FPGA), it is mandatory to validate the current LVDS pin assignments in the Quartus Prime software before finalizing the carrier board design.

  Before the hardware engineer defines the LVDS pin mapping and performs routing on the carrier board,

you must:

- Create a test project in Quartus Prime
- Instantiate the required LVDS IP
- Apply the intended pin assignments
- Perform a full compilation to verify design feasibility

This step ensures that the selected LVDS pins are valid and prevents potential carrier board redesign due to pin incompatibility.

2.4 Hardware Monitoring & Power Management

In the original Arria10 SOM architecture, PMBus is used as the primary power monitoring interface, allowing the carrier board to directly monitor various voltages and currents on the SOM.

On the Titan S10, the PMBus architecture has the following enhancements:

- **Reduced Monitoring Scope**
  The PMBus signals exposed through the FMCR connector on the Titan S10 are only connected to the **FPGA core power modules (Core Power)**.
  PMBus alone can no longer be used to monitor all power rails across the entire board.
  - If you would like to fully monitor the SOM status (including core power, board temperature, fan speed, etc.), you can connect the MCU to four FPGA I/O pins on the SOM (used as an SPI interface). Through the SPI interface between the FPGA and the MAX 10, the MCU can access internal register values in the MAX 10 to retrieve data such as core power, board temperature, and fan speed.

  The data path will be: MCU → FPGA → MAX 10
- **Hardware-Based Ownership Control**
  On the A10 platform, if the carrier board MCU needs to act as the PMBus master, the MAX 10 firmware on the SOM must be modified to disable its master role.

  On the Titan S10, this is replaced by a hardware control mechanism, **meaning no MAX 10 firmware modification is necessary.** You simply pull the **PMBUS_DONG_EN_n** pin on the FMCR connector low (GND) to disable the built-in PMBus master function in the MAX 10.

2.5 HPS Gigabit Ethernet (GbE) Interface Architecture Differences and Migration

When migrating from the A10 SOM to the Titan S10 (Stratix 10), changes in the network interface architecture are one of the most easily overlooked yet critical hardware differences. While users do not need to modify the RJ-45 connectors on their carrier board because the new PHY is fully compatible with the previous RJ-45 magnetics, it is important to note that the internal signal routing to the SoC are different.

- **Network Architecture of A10 SOM: Standard Dual HPS Direct Connection**

  On the A10 SOM module, the network design follows a conventional SoC architecture:
  - **PHY Chips and Communication Protocol**: Both Ethernet ports use the Microchip KSZ9031RNXCA PHY chips and operate in RGMII (Reduced Gigabit Media Independent Interface) mode.
  - **Hardware Connection**: The RGMII signals from both PHYs are directly connected to the dedicated HPS I/O pins of the Arria 10 SoC and interface directly with the two internal MAC controllers (EMAC0 / EMAC1) inside the HPS.

- **Network Architecture of Terasic Titan S10 SOM (Stratix 10): Hybrid HPS + FPGA Design**

  Due to the limited number of HPS I/O pins on the Stratix 10 SoC, the Titan S10 adopts a hybrid design: one port connects directly to HPS I/O pins, while the other connects to FPGA I/O pins via the FPGA fabric.

  - **Ethernet Port A (ENETA)**:
    - **Design**: Equivalent to the Arria10 SOM architecture. Uses KSZ9031RN PHY (RGMII mode).
    - **Connection**: Directly connected to the HPS dedicated I/O pins and internal MAC controller of the Stratix 10. It is recognized as `eth0` in Linux.
  - **Ethernet Port B (ENETB) [Key Difference]**:
    - **Design**: Uses KSZ9031MNXIC PHY and operates in GMII (Gigabit Media Independent Interface) mode, which requires more pins. **Although the PHY chip for this port is changed to KSZ9031MNXIC, it is fully compatible with the RJ-45 magnetics used by the KSZ9031RN PHY. Therefore, users do not need to replace the RJ-45 connector or modify this part of their carrier board design.**
    - **Connection**: The PHY I/O signals are connected directly to the FPGA I/O pins via the FPGA fabric (core logic region), with no connections to the HPS dedicated I/O.

2.5.1 Step-by-Step Guide: How to Route Port B (ENETB) Back to HPS for Linux Use on Titan S10

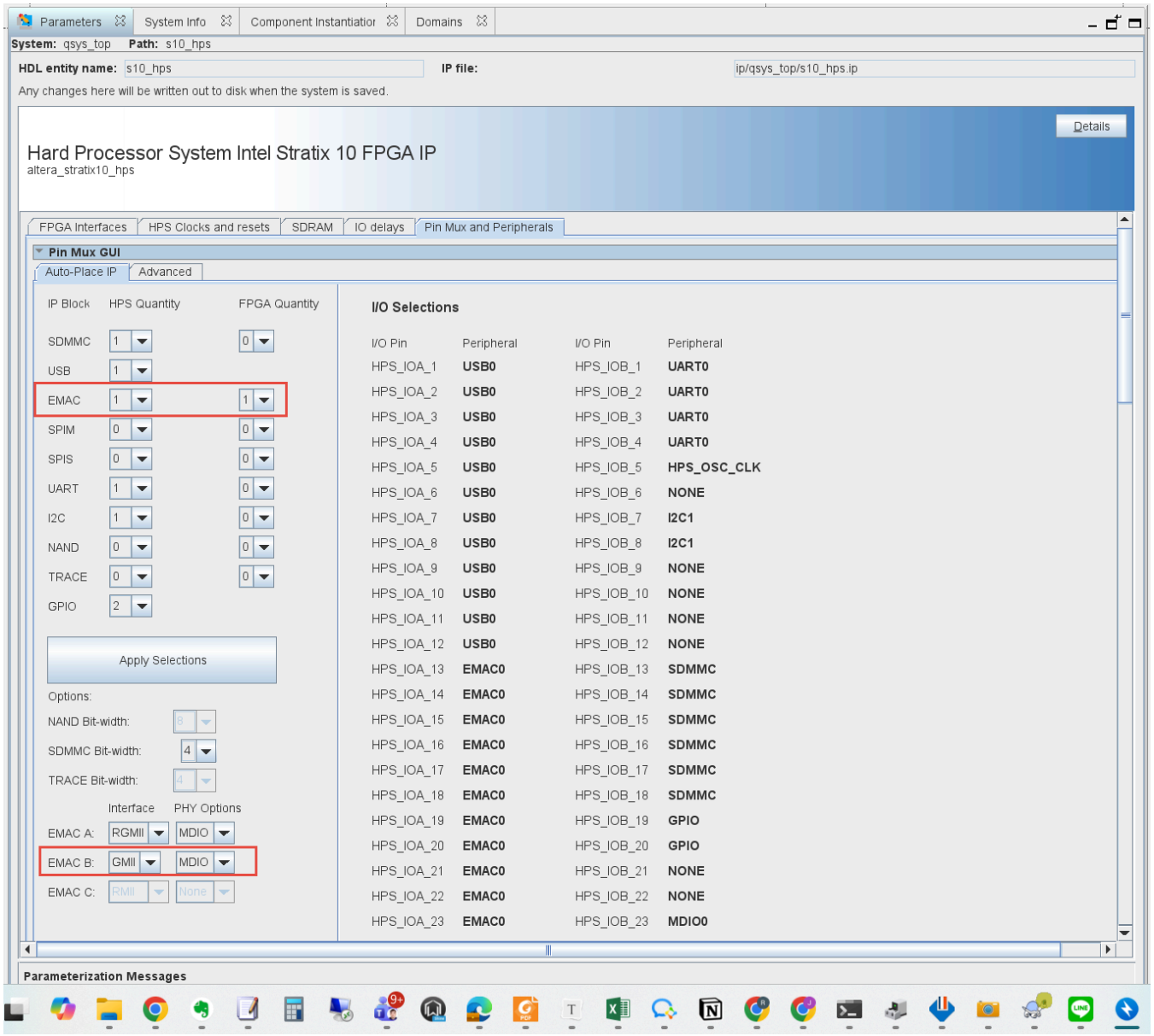
Although Port B on the Titan S10 is physically connected to the FPGA, you do not need to implement or purchase a costly soft MAC IP in the FPGA. Instead, you can use Platform Designer (Qsys) in Quartus to route the second hardware MAC controller inside the HPS (e.g., EMAC1 or EMAC2) through the FPGA to interface directly with the PHY.

(Note: Users can refer to the GHRD project in the Terasic Resource Package for reference designs.)

Below are the detailed implementation steps:

Step 1: Configure HPS EMAC Interface in Platform Designer (Qsys)

1. Open Platform Designer in Quartus Prime Pro and enter the Stratix 10 HPS IP configuration.
2. Locate the **Pin Mux and Peripherals** settings.
3. Configure the EMAC as shown in the figure below: one interface under **HPS Quantity**, and the other under **FPGA Quantity**.
4. Set **EMAC A** interface to `RGMII`, and **EMAC B** to `GMII`.
5. After configuration, the HPS IP will expose full GMII interface signals (ports), such as: `emac1_phy_txd_o[7:0]`, `emac1_phy_txen_o`, `emac1_phy_rxdv_i`, `emac1_phy_rxd_i[7:0]`, `emac1_clk_rx_i`, `emac1_phy_txclk_o`, `emac1_gmii_mdc_o`, `emac1_gmii_mdo_o`, etc.



Step 2: Perform Physical Connections in Top-Level RTL (Verilog/VHDL)

After generating the Qsys system, you must connect the HPS-exposed EMAC GMII signals to the FPGA external pins (i.e., pins connected to the KSZ9031MNXIC PHY) in the top-level RTL of your FPGA project.

1. Connect HPS output `emac_phy_txd_o[7:0]` to external `ENETB_TX_DATA[7..0]`.
2. Connect incoming `ENETB_RX_DATA[7..0]` to HPS input `emac_phy_rxd_i[7:0]`.
3. Connect corresponding clock signals (`ENETB_GTX_CLK`, `ENETB_RX_CLK`) and control signals (`TX_EN`, `RX_DV`).
4. **MDIO Management Interface**: Ensure HPS `emac_gmii_mdc_o` and `emac_gmii_mdio` are connected to `ENETB_MDC` and `ENETB_MDIO`. This allows the Linux PHY driver to access PHY registers via MDIO to retrieve link status and speed information.

Step 3: Software and Linux Device Tree (DTB) Modifications

Since the hardware routing has changed, your Linux Device Tree must reflect these updates.

1. Ensure the second Ethernet interface is defined in the device tree (e.g., `ethernet@ff802000`).
2. Set the `phy-mode` property of this interface to `"gmii"`.
3. Verify that the MDIO bus address correctly matches the physical address of the Port B PHY.
4. Refer to the documentation below to configure PHY TX/RX clock skew settings: https://github.com/johnnyfan1979/public_doc/blob/main/documentation/Titan_S10_SOM/Titan_S10_revB_Ethernet_PortB_GMII_skew_settings.md
5. Recompile the Linux kernel with the new device tree. Linux kernel will need to be recompiled for the 64-bit processor architecture.

---

3. Quartus Project Migration Arria 10 SOM

3.1 Practical Advice for Developers

When developing a Titan S10 (Stratix 10) project, it is highly recommended to use the GHRD (Golden Hardware Reference Design) provided by Terasic as the foundation for extension. Do not create the project from scratch. This approach helps avoid issues such as HPS IO Hash mismatches that can lead to errors like “FPGA not ready” and system hangs. It also ensures proper DDR4 (HPS_EMIF) operation and a stable hardware handoff.

Please directly copy the GHRD project from the Titan S10 resource package under:
`\Demonstration\SoC_FPGA\GHRD`

In Platform Designer (Qsys), keep the existing HPS and EMIF default settings, and only add your custom FPGA logic on top of this architecture.

3.2 Key Considerations When Creating a Project from Scratch

3.2.1 Adding VID and Power Management Settings

If you are creating a **new custom Quartus project from scratch**, please note: You must manually add the **VID and Power Management configurations** to your project's .qsf (Quartus Settings File) to ensure the FPGA's power management functions operate correctly.

💡 Tip: You can also find this complete configuration block in the .qsf file of the Golden Top Project we provided. Feel free to copy and paste it directly into your own project!

To add it manually, please open your project's .qsf file and insert the following assignment parameters:

```
#============================================================
# VID Assignments
#============================================================
set_global_assignment -name USE_PWRMGT_SCL SDM_IO0
set_global_assignment -name USE_PWRMGT_SDA SDM_IO12
set_global_assignment -name VID_OPERATION_MODE "PMBUS MASTER"
set_global_assignment -name PWRMGT_BUS_SPEED_MODE "400 KHZ"
set_global_assignment -name PWRMGT_SLAVE_DEVICE_TYPE OTHER
set_global_assignment -name PWRMGT_SLAVE_DEVICE0_ADDRESS 1B
```

```
set_global_assignment -name PWRMGT_SLAVE_DEVICE1_ADDRESS 00
set_global_assignment -name PWRMGT_SLAVE_DEVICE2_ADDRESS 00
set_global_assignment -name PWRMGT_SLAVE_DEVICE3_ADDRESS 00
set_global_assignment -name PWRMGT_SLAVE_DEVICE4_ADDRESS 00
set_global_assignment -name PWRMGT_SLAVE_DEVICE5_ADDRESS 00
set_global_assignment -name PWRMGT_SLAVE_DEVICE6_ADDRESS 00
set_global_assignment -name PWRMGT_SLAVE_DEVICE7_ADDRESS 00
set_global_assignment -name PWRMGT_PAGE_COMMAND_ENABLE OFF
set_global_assignment -name PWRMGT_VOLTAGE_OUTPUT_FORMAT "DIRECT FORMAT"
set_global_assignment -name PWRMGT_DIRECT_FORMAT_COEFFICIENT_M 200
set_global_assignment -name PWRMGT_DIRECT_FORMAT_COEFFICIENT_B "-49"
set_global_assignment -name PWRMGT_DIRECT_FORMAT_COEFFICIENT_R 0
set_global_assignment -name PWRMGT_TRANSLATED_VOLTAGE_VALUE_UNIT VOLTS
set_global_assignment -name MIN_CORE_JUNCTION_TEMP "-40"
set_global_assignment -name MAX_CORE_JUNCTION_TEMP 100
set_global_assignment -name HPS_INITIALIZATION "HPS FIRST"
set_global_assignment -name STRATIX_JTAG_USER_CODE 4
```

3.2.2 Check HPS EMIF (DDR4) Settings

Please be sure to refer to the GHRD (Golden Hardware Reference Design) project provided by Terasic and ensure that the following settings are strictly consistent with the GHRD:

1. The DDR4 **Pin Assignments** are perfectly matched.
2. The `emif_hps` settings within Platform Designer (including parameters such as Memory Size, Timing, and Reference Clock) are identical.

Failure to follow the standard settings can easily lead to the following fatal errors during the boot process: Memory calibration failure caused by incorrect timing or clock settings (e.g., `DDR: Error as SDRAM calibration failed`).

---

4. Software, Linux & Boot Flow

This chapter provides an overview of the key software migration tasks when moving from the Arria 10 SOM to Titan S10 SOM, focusing on Linux, boot architecture, and deployment flow. It explains the transition from a 32-bit Cortex-A9 platform to a 64-bit Cortex-A53 system, the adoption of the new SDM-based dual-flash boot architecture, the introduction of Arm Trusted Firmware (ATF), and the required updates to U-Boot, Linux kernel, and device tree design. In addition, this chapter highlights practical development resources for building and programming the Linux system, including GSRD, eMMC update methods, and the complete `.jic` generation and flashing procedure for QSPI boot preparation.

4.1. Processor Architecture & Toolchain Migration (32-bit → 64-bit)

- **Key Difference**:
  Arria 10 uses a 32-bit dual-core ARM Cortex-A9 processor, while Stratix 10 (Titan S10) upgrades to a 64-bit quad-core ARM Cortex-A53 processor.
- **Required Actions**:
  The original 32-bit toolchain must be replaced with a 64-bit `aarch64` cross-compilation toolchain (e.g.,

`aarch64-linux-gnu-gcc`).
All components must be recompiled, including:

- Linux kernel
- U-Boot
- Device drivers
- C/C++ applications
- Bare-metal programs if operating in bare metal mode

4.2. Adoption of the New SDM-Based Boot Flow

- **Key Difference**:
  In Arria 10, the boot process is managed by the internal HPS Boot ROM.
  In Stratix 10, a dedicated **Secure Device Manager (SDM)** is introduced to control both FPGA configuration and HPS boot.
- **Required Actions**:
  - Titan S10 uses **HPS Boot First dual-flash mode** by default.
  - Boot images must be split into two storage devices:
    1. **SDM Flash (QSPI Flash)**:
       - SDM firmware
       - HPS EMIF configuration
       - HPS FSBL
    2. **HPS Flash (eMMC Flash)**:
       - FPGA I/O configuration
       - U-Boot SSBL
       - Linux kernel
       - Device Tree (DTB)
       - Root File System (RootFS)

4.3. Introduction of Arm Trusted Firmware (ATF) & Bootloader Redesign

- **Key Difference**:
  Stratix 10 significantly enhances the TrustZone security architecture by introducing **Arm Trusted Firmware (ATF)**.
  U-Boot and Linux must access secure system resources via **SMC (Secure Monitor Call)**.
  The legacy `bsp-editor` tool used in Arria 10 is deprecated.
- **Required Actions**:
  - Build ATF (`bl31.bin`) as the secure monitor firmware.
  - Generate U-Boot and Device Tree directly via build scripts.
  - Rebuild:
    - FSBL

        - SSBL (U-Boot)

4.4. Linux Kernel & Device Tree (DTB) Porting

- **Key Difference**:
  Stratix 10 introduces a **Cache Coherency Unit (CCU)** in the system interconnect.
- **Required Actions**:
    - Use a Stratix 10–specific Linux kernel branch.
    - Base the device tree on `socfpga_stratix10_socdk.dts` and modify accordingly.

  Important Note – Network Routing:
    - On Titan S10, the second Ethernet PHY (Port B) is connected through the FPGA fabric (GMII interface), not directly to the HPS.
    - The device tree must correctly define:
        - MAC IP mapping
        - Interface routing
    - Otherwise, Linux will not detect the second Ethernet interface properly.

> 💡 **Summary Recommendation**:
>
> The primary effort in software migration lies in:
>
> - Low-level bootloader setup (SDM firmware, ATF, U-Boot)
> - Transition to a 64-bit system
>
> For the Titan S10 platform, these foundational migration tasks have already been completed.
> Users can directly leverage the provided reference source code to accelerate development.

4.5 Recommended Development Resources: Linux System Design and eMMC Programming Guide

To help you quickly build a Linux system on the Titan S10 (Stratix 10) and successfully program the compiled system image into the on-board eMMC storage device, we strongly recommend referring to the following comprehensive guides during your development:

1. GSRD (Golden Hardware and Software Reference Design)

- **Description**: This document provides a complete tutorial on the Golden System Reference Design (GSRD) for the Titan S10, covering the standard process from building the underlying hardware architecture to integrating the software (Linux / U-Boot / Device Tree). It is highly recommended to use this as the starting point for all customized projects.
- **Reference Link**: Titan S10 SOM GSRD

2. eMMC Programming via TFTP Guide

- **Description**: During the development phase, updating via the network is the most efficient method. This document provides detailed instructions on how to set up a TFTP server, download the system image to the development board via Ethernet, and write it to the eMMC.
- **Reference Link**: Titan S10 Som Write EMMC with TFTP

3. eMMC Programming via USB Flash Disk

- **Description**: If you are in a development environment without an internal network, this guide will show you how to use the USB interface on the carrier board to plug in a USB flash drive and use Linux commands to clone the image from the drive directly into the eMMC.
- **Reference Link**: Titan S10 SoM Programming eMMC with USB Flash Disk

4.6 Generating and Flashing the Titan S10 SoM .jic

To help you quickly build a Linux system on the Titan S10 (Stratix 10) and successfully program the compiled system image into the on-board eMMC storage device, we strongly recommend referring to the following comprehensive guides during your development:

1. Prepare scripts and files

📍 **Purpose:** Collect all required dependencies for subsequent format conversion and programming steps. This includes automation scripts for merging and format conversion, configuration files, and HPS boot-related software components (such as U-Boot SPL).

- After compilation, copy `sof_with_hps.bat`, `sof_to_jic.bat`, and `flash_image_qspi_hps.pfg` from the GHRD project to the Quartus project.
- Copy the `software` directory from the GHRD to the Quartus project.
- Copy `program_qspi_flash` from `GHRD/output_files` to the `output_files` directory of the Quartus project.

2. Generate core.rbf and jic files

📍 **Purpose:** Bind the compiled FPGA hardware configuration (`.sof`) with the HPS bootloader (U-Boot SPL), and convert formats accordingly. The generated `.jic` is used for non-volatile storage and handles early-stage boot, while the `.core.rbf` is a pure FPGA bitstream used by U-Boot for dynamic FPGA configuration in later stages.

- Double-click to run the `sof_with_hps.bat` script to generate the `golden_top_hps.sof` file in the `output_files` directory, achieving the binding of the sof and U-Boot SPL.
- Double-click to run the `sof_to_jic.bat` script to generate the `golden_top_hps.hps.jic` and `golden_top_hps.core.rbf` files.

3. Flash the jic to QSPI flash

📍 **Purpose:** Write the firmware package (including the First Stage Bootloader, SPL) into the on-board non-volatile memory (QSPI Flash). This ensures that after power cycle, the SoM can boot the HPS from QSPI Flash.

- Connect the board to the PC using a download cable.
- Double-click to run `flash_program_hps.bat` under `output_files/program_qspi_flash` to flash the jic to the QSPI flash.

4. Regenerate kernel.itb (execute in a Linux environment)

📍 **Purpose:** Package the Linux Kernel, device tree (`.dtb`), and the generated FPGA bitstream (`.core.rbf`) into a FIT (Flattened Image Tree) image (`kernel.itb`). This allows U-Boot to load all software and hardware configurations from eMMC in a unified and consistent manner during Linux boot.

- You need to prepare the Linux Image file, the `fit_kernel_stratix10_titan_s10_som.its` file, and the `socfpga_stratix10_titan_s10_som.dtb` file. Place these files in the same directory, along with the previously generated `golden_top_hps.core.rbf`.
- The contents of the `fit_kernel_stratix10_titan_s10_som.its` file are as follows:

```
/*
 * Copyright (C) 2019 Intel Corporation
 *
 */

/dts-v1/;

/ {
    description = "FIT image with kernel, DTB and FPGA core binary";
    #address-cells = <1>;

    images {
        kernel {
            description = "Linux Kernel";
            data = /incbin/("./Image.lzma");
            type = "kernel";
            arch = "arm64";
            os = "linux";
            compression = "lzma";
            load = <0x6000000>;
            entry = <0x6000000>;
            hash {
                algo = "crc32";
            };
        };


        fdt-4 {
            description = "socfpga_socdk_combined";
            data = /incbin/("./socfpga_stratix10_titan_s10_som.dtb");
            type = "flat_dt";
            arch = "arm64";
            compression = "none";
            hash {
                algo = "crc32";
            };
        };

        fpga-4 {
            description = "FPGA bitstream for GHRD";
```

```
                data = /incbin/("./golden_top_hps.core.rbf");
                type = "fpga";
                arch = "arm64";
                compression = "none";
                load = <0xA000000>;
                hash {
                    algo = "crc32";
                };
            };
        };

        configurations {
            default = "board-4";


            board-4 {
                description = "board_4";
                kernel = "kernel";
                fdt = "fdt-4";
                fpga = "fpga-4";
                signature {
                    algo = "crc32";
                    key-name-hint = "dev";
                    sign-images = "fdt-4", "kernel", "fpga-4";
                };
            };
        };
    };
```

- Generate kernel.itb

```
# Convert Image to Image.lzma
    xz --format=lzma Image

# Generate kernel.itb
    mkimage -f fit_kernel_stratix10_titan_s10_som.its kernel.itb
```

- Replace the `kernel.itb` in the eMMC image with the newly generated `kernel.itb`.

---

5. Appendix

5.1 Key Considerations for Migrating FIFO IP from Arria 10 to Stratix 10

1. Output Register and Read Mode Binding

- **Note:** For Stratix 10 DCFIFO, it strictly requires registered outputs in Normal mode and unregistered outputs in Show-ahead mode.
- **Migration Impact:** Arria 10 allows you to dynamically choose whether to register the `q` output via the `add_ram_output_register` parameter. If your Arria 10 project uses "Normal mode with unregistered

output" or "Show-ahead mode with registered output," this configuration is not supported in Stratix 10. You must modify your design and adjust the read timing accordingly.

2. Power-up Reset Requirement

- **Note:** In Stratix 10 devices, to ensure the FIFO IP operates correctly, you **must assert the `aclr` (asynchronous clear) or `sclr` (synchronous clear) signal during system power-up**. Please verify that your system's reset circuitry includes an automatic clear mechanism upon power-up.

3. True Dual-Port (TDP) Dual Clock Mode Limitations

- **Note:** If your underlying memory architecture uses the True Dual-Port (TDP) dual clock mode, Arria 10 supports `aclr`, `sclr`, and `byteena` (byte enable) signals. However, Stratix 10 uses an **Emulated TDP dual clock mode, which does not support the `aclr`, `sclr`, and `byteena` signals**.
- **Clock and Depth Restrictions:** In Stratix 10's emulated TDP mode, the clock frequency ratio of the fast clock (Clock B) divided by the slow clock (Clock A) is recommended to be greater than or equal to 7. Furthermore, the **DCFIFO depth must be strictly greater than this clock frequency ratio** for the IP to function properly.

4. Asynchronous Clear (`aclr`) Synchronization Design

- **Note:** For Stratix series DCFIFO, asserting the falling edge of the `aclr` signal while the `wrreq` (write request) is high can create a race condition with the rising edge of the write clock (`wrclk`). Stratix 10 supports the `write_aclr_synch` parameter, which automatically adds a circuit to internally synchronize the `aclr` signal with the write clock, preventing potential FIFO corruption.

Revision History

Date	Version	Changes
2026.04	V1.0	First Version