Keystone Bootloader
Keystone ROM Boot Loader

- Code to transfer application code from memory or host to high speed internal memory
- Boot loader code is burned in the DSP ROM (Non-modifiable)
- Base address for the Boot Code is 0x20B00000
- Boot Loader is broadly divided into two types
  - Memory boot where application is stored in a slow external memory
  - Host Boot where the boot is driven by a host device connected through fast transport.
- Seven different types of boot modes are supported
ROM Boot Modes

- Supported Boot Modes
  - I2C Boot
    - Master Boot (from I2C EEPROM)
    - Master-Broadcast Boot (Master Boot followed by broadcast to slave cores)
    - Passive Boot (external I2C host)
  - SPI Boot (from SPI flash)
  - SRIO Boot (from external host connected through SRIO)
  - Ethernet Boot (boot from external host connected through Ethernet)
  - PCIe Boot (boot from external host connected through PCIe)
  - HyperLink Boot (boot from external host connected through HyperLink)
  - EMIF16 NOR Boot (boot from NOR Flash)
    - Device Manual will detail supported types.

![Diagram of ROM Boot Modes](image-url)
Boot Mode Configuration Pins

- Boot mode and configurations are chosen using bootstrap pins on the device.
  - Pins are latched and stored in 13 bits of the DEVSTAT register during POR.
- The configuration format for these 13 bits are shown in the table:

<table>
<thead>
<tr>
<th>Boot Mode Pins</th>
<th>PLL Multi</th>
<th>I2C/SPI Ext Dev Cfg</th>
<th>Device Configuration</th>
<th>Boot Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- Boot Device [2:0] is dedicated for selecting the boot mode
- Device Configuration [9:3] is used to specify the boot mode specific configurations.
- PLL Multi [12:10] are used for PLL selection. In case of I2C/SPI boot mode, it is used for extended device configuration. (PLL is bypassed for these two boot modes)
Device Startup from Power on Reset (POR)

- Boot Startup procedure executed only once during:
  - Power On
  - Hard Reset
  - Soft Reset
- Bootstrap pins are only latched during Power On Reset (POR)
- Default boot parameter table is chosen based on the selected boot mode
- Boot strap pin configuration parameters are updated in the boot parameter table
- At completion, the ROM code branches to the main boot function, using the table to configure boot operation
  - The boot table can be modified and Boot can be re-executed after startup is complete by branching to the boot run function. (Typical Case: Secondary Boot Loader – I2C loads custom parameter table)
Device Startup from Hard/Soft reset

• For hard and soft resets the Boot code must determine the hibernation state.
  – Hibernation is the process of shutting down unused CorePacs and IP blocks to save power consumption of the overall system.

• Saving all relevant configurations and register values is the application’s responsibility based on the selected hibernation mode.
  – Hibernation1 – Values stored in MSMC SRAM.
  – Hibernation2 – Values stored in DDR3.

• The Application is also responsible for setting the appropriate hibernation mode in the PWRSTATECTL register.

• The Application will also set the branch address in the PWRSTATECTL register.
Hibernation explained

• Hibernation 1
  – The application needs to ensure that the chip control register is set correctly to avoid MSMC reset.

• Hibernation 2
  – MSMC is reinitialized to default values.

• For both modes, the Application is responsible for shutdown of all desired IP blocks

• A hard or soft reset can be configured to bring a hibernating device out of hibernation
  – After the reset, the boot loader code checks the PWRSTATECTL register to identify the hibernation mode and branch address.
  – Subsequent Actions
    • Peripherals and Corepacs are powered
    • The awakened device branches to the application code which utilizes the values stored in MSMC or DDR3 prior to hibernation
PLL Configuration

- The boot code sets the PLL multiplier based on the core frequency set in the EFUSE register.

<table>
<thead>
<tr>
<th>Boot PLL Select [2:0]</th>
<th>Input Clock Freq (MHz)</th>
<th>core = 800 MHz Cluster</th>
<th>core = 1000 MHz Cluster</th>
<th>core = 1200 MHz Cluster</th>
<th>core = 1400 MHz Cluster</th>
<th>Core = 1250 MHz Cluster</th>
<th>Core = 1500 MHz Cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>50.00</td>
<td>Clkr: 0</td>
<td>Clkr: 0</td>
<td>Clkr: 0</td>
<td>Clkr: 0</td>
<td>Clkr: 55</td>
<td>Clkr: 0</td>
</tr>
<tr>
<td>1</td>
<td>66.67</td>
<td>Clkr: 0</td>
<td>Clkr: 0</td>
<td>Clkr: 0</td>
<td>Clkr: 0</td>
<td>Clkr: 1</td>
<td>Clkr: 0</td>
</tr>
<tr>
<td>2</td>
<td>80.00</td>
<td>Clkr: 0</td>
<td>Clkr: 0</td>
<td>Clkr: 0</td>
<td>Clkr: 0</td>
<td>Clkr: 0</td>
<td>Clkr: 4</td>
</tr>
<tr>
<td>3</td>
<td>100.00</td>
<td>Clkr: 0</td>
<td>Clkr: 0</td>
<td>Clkr: 0</td>
<td>Clkr: 0</td>
<td>Clkr: 0</td>
<td>Clkr: 1</td>
</tr>
<tr>
<td>4</td>
<td>156.25</td>
<td>Clkr: 0</td>
<td>Clkr: 0</td>
<td>Clkr: 0</td>
<td>Clkr: 0</td>
<td>Clkr: 4</td>
<td>Clkr: 4</td>
</tr>
<tr>
<td>5</td>
<td>250.00</td>
<td>Clkr: 0</td>
<td>Clkr: 0</td>
<td>Clkr: 0</td>
<td>Clkr: 0</td>
<td>Clkr: 0</td>
<td>Clkr: 9</td>
</tr>
<tr>
<td>6</td>
<td>312.50</td>
<td>Clkr: 0</td>
<td>Clkr: 0</td>
<td>Clkr: 0</td>
<td>Clkr: 0</td>
<td>Clkr: 0</td>
<td>Clkr: 7</td>
</tr>
<tr>
<td>7</td>
<td>47.22</td>
<td>Clkr: 0</td>
<td>Clkr: 0</td>
<td>Clkr: 0</td>
<td>Clkr: 0</td>
<td>Clkr: 2</td>
<td>Clkr: 12</td>
</tr>
</tbody>
</table>

Clkr: Clk Reference Frequency
Clkf: Clk Frequency
## Boot Device

### Boot Device Selection Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Boot Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Sleep / EMIF16(^1)</td>
</tr>
<tr>
<td>1</td>
<td>Serial Rapid I/O</td>
</tr>
<tr>
<td>2</td>
<td>Ethernet (SGMII) (PA driven from core clk)</td>
</tr>
<tr>
<td>3</td>
<td>Ethernet (SGMII) (PA driver from PA clk)</td>
</tr>
<tr>
<td>4</td>
<td>PCIe</td>
</tr>
<tr>
<td>5</td>
<td>I2C</td>
</tr>
<tr>
<td>6</td>
<td>SPI</td>
</tr>
<tr>
<td>7</td>
<td>HyperLink</td>
</tr>
</tbody>
</table>

1. See the device-specific data manual for information.

- For interfaces supporting more than one mode of operation, the configuration bits are used to establish the necessary settings.
Boot Configuration – EMIF16 Mode

- EMIF16 mode is used to boot from the NOR flash.
- The boot loader configures the EMIF16 and then sets the boot complete bit corresponding to corePac0 in the boot complete register and then branches to EMIF16 CS2 data memory at 0x70000000.
- No Memory is reserved by the boot loader.

<table>
<thead>
<tr>
<th>Sleep / EMIF16 Configuration Bit Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
</tr>
<tr>
<td>Reserved</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sleep / EMIF16 Configuration Bit Field Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit Field</td>
</tr>
<tr>
<td>Sub-Mode</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Wait Enable</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
Boot Configuration – Ethernet

- Ethernet(SGMII) boot configuration sets SERDES clock and device ID.

<table>
<thead>
<tr>
<th>Ethernet (SGMII) Device Configuration Bit fields description</th>
<th>Bit field</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERDES Clock Mult</td>
<td>9</td>
<td>0</td>
<td>x8 for input clock of 156.25 MHz</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>x5 for input clock of 250 MHz</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>x4 for input clock of 312.5 MHz</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>3</td>
<td>Reserved</td>
</tr>
<tr>
<td>Ext connection</td>
<td>8</td>
<td>0</td>
<td>Mac to Mac connection, master with auto negotiation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>Mac to Mac connection, slave, and Mac to Phy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>Mac to Mac, forced link</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>Mac to fiber connection</td>
</tr>
<tr>
<td>Device ID</td>
<td>7</td>
<td>0-7</td>
<td>This value is used in the device ID field of the Ethernet ready frame. Bits 1:0 are use for the SR ID.</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The output frequency of the PLL must be 1.25 GBs.
Boot Configuration – Serial RapidIO

- SRIO boot configuration sets the Clock, Lane configuration, and mode

<table>
<thead>
<tr>
<th>Rapid I/O Device Configuration Bit Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td>Lane Setup</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SRIO Configuration Bit Field Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit Field</td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td>SR ID</td>
</tr>
<tr>
<td>Ref Clock</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Data Rate</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Lane Setup</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
Boot Configuration

I2C Master Mode

- In master mode the I2C Device Configuration uses 7 bits of device configuration instead of 5 bits used in passive mode.
- In this mode device will make the initial read of the I2C EEPROM while the PLL is in bypass.
- The initial boot parameter table will contain the desired clock multiplier which will be setup prior to any subsequent reads.

<table>
<thead>
<tr>
<th>Bit Field</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode</td>
<td>0</td>
<td>Master Mode</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Passive Mode</td>
</tr>
<tr>
<td>Address</td>
<td>0</td>
<td>Boot From I2C EEPROM at I2C bus address 0x50</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Boot From I2C EEPROM at I2C bus address 0x51</td>
</tr>
<tr>
<td>Speed</td>
<td>0</td>
<td>I2C data rate set to approximately 20 kHz</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>I2C fast mode. Data rate set to approximately 400 kHz (will not exceed)</td>
</tr>
<tr>
<td>Parameter Index</td>
<td>0-31</td>
<td>Identifies the index of the configuration table initially read from the I2C EEPROM</td>
</tr>
</tbody>
</table>
Boot Configuration – I2C Passive Mode

• In passive mode the I2C Device Configuration uses 5 bits of device configuration instead of 7 used in master mode.

• In passive mode the device does not drive the clock, but simply acks data received on the specified address.

<table>
<thead>
<tr>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rsvd (Must be 1)</td>
<td>Mode (1)</td>
<td>Receive I2C Address</td>
<td></td>
<td></td>
<td></td>
<td>Rsvd</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit Field</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode</td>
<td>0</td>
<td>Master Mode</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Passive Mode</td>
</tr>
<tr>
<td>Address</td>
<td>0-7</td>
<td>The I2C Bus address the device will listen to for data</td>
</tr>
</tbody>
</table>
Boot Configuration – SPI Mode

Similar to I2C, the bootloader reads either a boot parameter table or boot config table that is at the address specified by the first boot parameter table and executes it directly.

### SPI Device Configuration Field Descriptions

<table>
<thead>
<tr>
<th>Bit Field</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode (clk Pol/Phase)</td>
<td>0</td>
<td>Data is output on the rising edge of SPICLK. Input data is latched on the falling edge.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Data is output one half-cycle before the first rising edge of SPICLK and on subsequent falling edges. Input data is latched on the rising edge of SPICLK.</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Data is output on the falling edge of SPICLK. Input data is latched on the rising edge.</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Data is output one half-cycle before the first falling edge of SPICLK and on subsequent rising edges. Input data is latched on the falling edge of SPICLK.</td>
</tr>
<tr>
<td>4,5 pin</td>
<td>0</td>
<td>4 pin mode used</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>5 pin mode used</td>
</tr>
<tr>
<td>Addr Width</td>
<td>0</td>
<td>16 bit address values are used</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>24 bit address values are used</td>
</tr>
<tr>
<td>Chip Select</td>
<td>0-3</td>
<td>The chip select field value</td>
</tr>
<tr>
<td>Parameter Table Index</td>
<td>0-3</td>
<td>Specifies which parameter table is loaded</td>
</tr>
<tr>
<td>SR Index</td>
<td>0-3</td>
<td>Smart Reflex Index</td>
</tr>
</tbody>
</table>
**Boot Configuration – PCI Express**

- In PCIe mode, the host configures memory and loads all the sections directly to the memory.

<table>
<thead>
<tr>
<th>Bit Field</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SR ID</td>
<td>0-3</td>
<td>Smart Reflex ID</td>
</tr>
<tr>
<td>Bar Config</td>
<td>0-0xf</td>
<td>See Next Slide</td>
</tr>
</tbody>
</table>

**PCI Device Configuration Bit Fields**

<table>
<thead>
<tr>
<th>Bit Field</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rsvd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAR Config</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SR ID</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Boot Configuration – PCI Express

<table>
<thead>
<tr>
<th>BAR cfg</th>
<th>BAR0</th>
<th>BAR1</th>
<th>BAR2</th>
<th>BAR3</th>
<th>BAR4</th>
<th>BAR5</th>
<th>64 bit Address Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0b0000</td>
<td>PCIe MMRs</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>Clone of BAR4</td>
</tr>
<tr>
<td>0b0001</td>
<td></td>
<td>16</td>
<td>16</td>
<td>32</td>
<td>64</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0b0010</td>
<td></td>
<td>16</td>
<td>32</td>
<td>32</td>
<td>64</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0b0011</td>
<td></td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>64</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0b0100</td>
<td></td>
<td>16</td>
<td>16</td>
<td>64</td>
<td>64</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0b0101</td>
<td></td>
<td>16</td>
<td>32</td>
<td>64</td>
<td>64</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0b0110</td>
<td></td>
<td>32</td>
<td>32</td>
<td>64</td>
<td>64</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0b0111</td>
<td></td>
<td>32</td>
<td>32</td>
<td>64</td>
<td>128</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0b1000</td>
<td></td>
<td>64</td>
<td>64</td>
<td>128</td>
<td>256</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0b1001</td>
<td></td>
<td>4</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0b1010</td>
<td></td>
<td>4</td>
<td>128</td>
<td>128</td>
<td>256</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0b1011</td>
<td></td>
<td>4</td>
<td>128</td>
<td>256</td>
<td>256</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0b1100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>256</td>
<td></td>
</tr>
<tr>
<td>0b1101</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>512</td>
<td></td>
</tr>
<tr>
<td>0b1110</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1024</td>
<td></td>
</tr>
<tr>
<td>0b1111</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2048</td>
<td></td>
</tr>
</tbody>
</table>
**Boot Configuration – HyperLink**

- HyperLink boot mode boots the DSP through the ultra short range HyperLink.
- The host loads the boot image directly through the link and then generates the interrupt to wake the DSP.

### MCM Boot Device Configuration

<table>
<thead>
<tr>
<th>Bit Field</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SR Index</td>
<td>0-3</td>
<td>Smart Reflex Index</td>
</tr>
<tr>
<td>Ref Clock</td>
<td>0</td>
<td>156.25 MHz</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>250 MHz</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>312.5 MHz</td>
</tr>
<tr>
<td>Data Rate</td>
<td>0</td>
<td>1.25 GBs</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>3.125 GBs</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>6.25 GBs</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>12.5 GBs</td>
</tr>
</tbody>
</table>
Booting multiple cores

- During the boot process the boot loader code is loaded into the L2 of corePac0 from the ROM.
- The high 0xD23F bytes of this L2 is reserved for the boot code. User should not overwrite this area.
- All the other CorePacs are executing IDLE.
- The user should load the image into the L2 of CorePacs they want to boot up. (Use MAD Utilities)
- Before setting the boot complete register, the user should also set the start address of the code in the respective BOOT MAGIC ADDRESS of the CorePac’s L2.
- Finally the user image should also write the IPC interrupt register to bring the required CorePacs out of IDLE.
Secondary Bootload Option
Second Stage Boot Load Process

Q: What if more boot parameters are needed than can be specified in the boot pins?

A: Other parameter values can be updated through I2C boot mode

• In this case, the I2C boot will start with a I2C boot parameter table which will in turn load a custom updated parameter table for a specific boot mode.

• Once the default parameter table is updated, the boot code executes using the updated boot parameter structure, using the same process as the primary boot mode.
Second Stage Boot Load Specifics

- The EEPROM image loaded will have two boot parameter tables
- The First one will be an I2C boot parameter table, setting the core clock and also the address of the next block.
- The next block will have the desired boot mode specific boot parameter table with the user desired values.
- After loading this image into the EEPROM, the boot mode in the boot strap is set for I2C master boot.
- After POR, the I2C boot code is executed as a first stage boot load, which will update the default boot parameter table and re-enter the boot code, executing the boot code utilizing the user specific parameters.
- Advantages
  - Additional Boot Modes (TFTP/NAND/NOR)
  - DDR initialization
Intermediate Bootloader and Keystone EVM Specifics
Intermediate Boot Loader
Intermediate Boot Loader Overview

• Reference Code (Board Specific)

• Two stage boot loaders for booting an application from the NOR/NAND flash or Ethernet over I2C EEPROM

• The 1st stage boot loader loads the IBL config table, initialize PLL(in EVM) and loads and runs the 2nd stage boot loader

• The 2nd stage boot loader loads the application image
IBL Operation

- IBL resides in I2C EEPROM in address 0x51.
- In EVM, the FPGA always boots the DSP in I2C master boot at address 0x51.
- IBL executes the PLL fix and then re-enters the boot process in the boot mode set by the bootstrap pins.
- In addition, IBL also supports NAND/NOR and Ethernet boot originally not supported by ROM Boot Loader.
IBL Boot Mode Selection

• NOR Boot
  – DIP switch settings (pin1, pin2, pin3, pin4) in EVM
    • SW3(off, off, on, off)
    • SW4(on, on, on, on)
    • SW5(on, on, on, off)
    • SW6(on, on, on, on)

• NAND Boot
  – DIP switch settings (pin1, pin2, pin3, pin4) in EVM
    • SW3(off, off, on, off)
    • SW4(on, off, on, on)
    • SW5(on, on, on, off)
    • SW6(on, on, on, on)

• TFTP(Ethernet) Boot
  – DIP switch settings (pin1, pin2, pin3, pin4) in EVM
    • SW3(off, off, on, off)
    • SW4(on, on, off, on)
    • SW5(on, on, on, off)
    • SW6(on, on, on, on)
IBL Operations
NOR Boot

- Multistage Boot Process
- Application is loaded to the DSP from the NOR flash.
- On Booting the ROM transfers the control to IBL and IBL loads the application from NOR
NAND Boot

- Multistage Boot Process
- Application is loaded to the DSP from the NAND flash.
- On Booting the ROM transfers the control to IBL and IBL loads the application from NAND
TFTP(Ethernet) Boot

- Multistage Boot Process
- Application is loaded to the DSP from the TFTP server.
- On Booting the ROM transfers the control to IBL and IBL loads the application from the remote TFTP server.
Keystone EVM Specifics
KeyStone Evaluation Modules

• TI offers three types of EVMs for KeyStone Devices.
  – TMDXEV56678L & TMDXEV56670L
    • Low Cost EVM
    • Cost Efficient development tools
  – TMDXEV56678LE & TMDXEV56670LE
    • Low Cost EVM
    • Cost Efficient development tools
    • Comes with XDS560v2 onboard emulation
  – TMDXEV56678LXE & TMDXEV56670LXE
    • Low Cost EVM
    • Cost Efficient development tools
    • Comes with XDS560v2 onboard emulation
    • Crypto enabled
KeyStone Evaluation Modules

TMDXEVM6670L
TMX320C6670 Evaluation Module

- DC 12V
- DDR3 1333 512 MB
- 60-pin TI DSP JTAG
- Boot Mode / Configuration Setting
- On-Board RS-232 Serial Port
- USB mini-B
- AMC Type B+
- Hyperlink Connector
- C6670 (4 core DSP)
- NAND Flash 64 MB
- FPGA
- 80-pin header
- Giga Ethernet

Texas Instruments
CI Training
KeyStone Evaluation Modules

Highlights

• 4 switches for Boot Mode/Configuration settings.
• 512MB DDR3.
• RS232 Serial Port.
• HyperLink Connector.
• AMC Compliant
• Giga bit Ethernet port
• 64MB NAND flash
• FPGA for control.
Silicon 1.0 Advisory 8 Workaround

• In Silicon version 1.0 not all PLLs lock after POR
  – See Advisory 8

• Boot through any IP blocks that need proper clock might fail.

• By default the Intermediate Boot Loader (IBL) is loaded in the EEPROM at the address space 0x51.

• FPGA in the EVM decodes the boot strap pins and ALWAYS boots in I2C master boot to boot from 0x51.

• The IBL initializes the PLL correctly and reads the FPGA register to get the original boot configuration and re-enters the boot process after updating the DEVSTAT.

• The FPGA firmware will not have this work around for the EVMs with KeyStone 2.0 Silicon.
EVM Software Kit

- EVM Software Kit includes
  - MultiCore Software Development Kit (MCSDK)
  - High Performance DSP Utility Application (HPDSPUA) Demo
  - Image Processing Demo
  - Power on Self Test (POST)
Power On Self Test (POST)

• Used for executing series of EVM factory tests on reset.
• The PASS/FAIL results are indicated by the set of LEDs.
• The test results are also displayed in serial console through UART.
• POST resides on the EEPROM of the EVM.
• List of functional test done during POST are:
  – External Memory Read/Write
  – NAND Read Test
  – NAND Write Test
  – EEPROM Read Test
  – UART Write Test
  – Ethernet Loopback Test
  – LED Test
Power On Self Test Cont (2)

- POST can be used to get the following information:
  - FPGA Version
  - Board Serial Number
  - EFUSE MAC ID
  - Crypto Availability
  - PLL status Register

- During boot, the FPGA boot in I2C master mode to address 0x51.
- In Factory default, the boot process then jumps to I2C address 0x50 and executes the POST that resides there.
- On success, the 4 FPGA debug LEDs will turn off.
Power On Self Test Cont (3)

```
SOC Information

FPGA Version: 0006
Board Serial Number: 0DCE9530
FUSE MAC ID is: 90 D7 EB 9E 5F 83
SA is disabled on this board.
PLL reset type status register: 0x0000001
Platform init return code: 0x0000000

Additional Information:
(0x02350014) : 00000000
(0x02350024) : 00000000
(0x02350038) : 00121000
(0x02350040) : 00000000
(0x02350044) : 00000000
(0x02350048) : 00000000
(0x0235004c) : 00000000
(0x02350050) : 00000000
(0x02350054) : 00000000
(0x02350058) : 00000000
(0x0235005c) : 00000000
(0x02350060) : 00000000
(0x02350064) : 00000000
(0x02350068) : 00000000
(0x0235006c) : 00000000
(0x02350070) : 00000000
(0x02350074) : 00000000
(0x02350078) : 00000000
(0x0235007c) : 00000000
(0x02350080) : 00000000
(0x02350084) : 00000000
(0x02350088) : 00000000
(0x0235008c) : 00000000
(0x02350090) : 00000000
(0x02350094) : 00000000
(0x02350098) : 00000000
(0x0235009c) : 00000000
(0x023500a0) : 00000000
(0x023500a4) : 00000000
(0x023500a8) : 00000000
(0x023500ac) : 00000000
(0x023500b0) : 00000000
(0x023500b4) : 00000000
(0x023500b8) : 00000000
(0x023500bc) : 00000000
(0x023500c0) : 00000000
(0x023500c4) : 00000000
(0x023500c8) : 00000000
(0x023500cc) : 00000000
(0x023500d0) : 00000000
(0x023500d4) : 00000000
(0x023500d8) : 00000000
(0x023500dc) : 00000000
(0x023500e0) : 00000000
(0x023500e4) : 00000000
(0x023500e8) : 00000000
(0x023500ec) : 00000000
(0x023500f0) : 00000000
(0x023500f4) : 00000000
(0x023500f8) : 00000000
(0x023500fc) : 00000000

Power On Self Test
POST running in progress ...
POST I2C EEPROM read test started!
POST I2C EEPROM read test passed!
POST SPI NOR read test started!
POST SPI NOR read test passed!
POST GPIO NAND read test started!
POST GPIO NAND read test passed!
POST EMAC loopback test started!
POST EMAC loopback test passed!
POST external memory test started!
POST external memory test passed!
POST done successfully!
```
Additional Details
Boot Runtime – Ethernet

- The SERDES, SGMII and switch are not configured if the options field of the Ethernet boot parameter table indicate initialization is bypassed. This will be the default case when the boot is initiated by hard or soft reset, reset isolation has been enabled, and the devices are powered up and enabled.
- SERDES: The boot ROM programs the SGMII_SERDES_CFGPLL register, the SGMII_SERDES_CFGRX registers (both lanes) and SGMII_SERDES_CFGTX register (both lanes).
- SGMII: The SGMII is enabled in full duplex mode, gigabit rate. Broadcast packet reception is enabled based on the boot parameter table.
- QMSS: The QMSS is configured to manage descriptors using a single memory region. Each descriptor is sized to 48 bytes with extended packet information block present.
- PASS: A custom firmware load is used for PASS. This load simply directs all received packets to the CPDMA, using flow configuration 0.
- Interrupt System: Polling is used to detect packet arrival in queue 896, so interrupts are not configured.
- After initialization, the device will broadcast a ready frame containing its device ID and MAC address.
- The Ethernet Host is responsible to receive the ready frame and follow up with the boot packets to the DSP using the device’s MAC address or ID. If Broadcast Rx is configured, then the host can broadcast a common image to all DSPs.
- The image is converted into a boot table and sent in packets from host to the DSP where the boot code reconstructs the image.
Boot Runtime – Serial RapidIO

• SRIO boot behaves as supported on previous devices for DirectIO mode. For Nyquist-Shannon, boot using Messaging Mode is supported as well, provided the host sends Ethernet IP messages.

• The boot ROM will not configure the SERDES or SRIIO if the boot options in the SRIIO boot parameter table show that configuration bypass is enabled. This will be the case in hard or soft reset when reset isolation is enabled and the SRIIO and SERDES have already been enabled.

• SERDES and SRIIO register configurations are based on templates. Basic values are taken from the template and modified to match the provided input frequency, number of lanes, and output frequency.

• SERDES: The SERDES is configured before the SRIIO. The boot ROM programs the SRIIO_SERDES_CFGPLL register, the SRIIO_SERDES_CFGRX registers, and the SRIIO_SERDES_CFGTX registers. The values programmed into these registers are based on the configurable parameters in the SRIIO boot parameter tables.

• In addition, for message booting over SRIIO, the RIO_RXU_MAP00, RIO_RXU_MAP00_H and RIO_RXU_MAP00_QID registers are programmed. They are programmed to route all messages (promiscuous) to queue 896 using flow ID 0.

• QMSS: The QMSS is configured to manage descriptors using a single memory region. Each descriptor is sized to 48 bytes with extended packet information block present (EInfo). The receive configuration is identical to that of Ethernet and a single function is used for both configurations.

• For DirectIO mode, the DSP will poll the boot magic address. Once this address is populated, the DSP branches to the address specified in the boot magic address. For Messaging Mode, the operation is equivalent to Ethernet Boot.
Boot Runtime – SPI

• The SPI is configured as directed by the boot parameter table. The SPI is operated through direct register reads and write.

• The boot ROM initializes the peripheral and begins reading blocks of data starting at the address specified in the boot parameter table (see Table 34). The ROM reads the data in blocks. Each block consists of two 16 bit word headers.

• The data in the blocks must contain boot table data. The data might also start with new boot parameter table. The new boot parameter block would specify a higher clock speed and a new boot address for the actual boot data.
Boot Configuration – PCIe Express

- The bootloader configures the base address registers, the number of windows, and their size.
- The PCIe power-up is configured through the external pin.
- If the PCIe boot is the primary boot, the BAR size configuration is driven by the BAR config fields as below.
- Once the BAR configurations are done, the host can access to the memory map of the DSP and image is loaded into the memory.
- At the end the boot magic address is also set to the entry point of the image.
- The DSP is brought out of IDLE by either a MSI interrupt or a legacy interrupt. (See PCIe user guide for further details.)
## Device Startup Summary

<table>
<thead>
<tr>
<th>ROM Boot Code Action</th>
<th>Reset Type</th>
<th>Hibernation</th>
<th>Core</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Power ON</td>
<td>NA</td>
<td>0</td>
<td>System Initialization, Full Boot</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Not 0</td>
<td>Set boot complete, idle, wake then branch</td>
</tr>
<tr>
<td></td>
<td>Hard, Soft</td>
<td>H-1</td>
<td>0</td>
<td>Restore MSMC, DDR, L2 MMRs, clear PWRSTATECTL, Branch to application defined location</td>
</tr>
<tr>
<td></td>
<td></td>
<td>None, H-2, Standby</td>
<td>0</td>
<td>Clear PWRSTATECTL, Branch to application defined location</td>
</tr>
<tr>
<td></td>
<td></td>
<td>All</td>
<td>Not 0</td>
<td>Wait for PWRSTATECTL = 0, Branch to application defined location</td>
</tr>
<tr>
<td></td>
<td>Local</td>
<td>All</td>
<td></td>
<td>Branch to application defined location</td>
</tr>
<tr>
<td></td>
<td>ROM Re-entry</td>
<td>All</td>
<td></td>
<td>Trap</td>
</tr>
</tbody>
</table>
Boot Runtime
I2C Master Mode

• Master Mode: In master mode the boot ROM reads blocks from the I2C EEPROM. The first 4 bytes are the header with the size of the block and the checksum.
  – The first block will be a **Boot Parameter Table**. They will set the proper PLL configuration for the core clock settings. Each block will be 128 bytes in size and the boot parameter table will specify the next block to read. The next block can be another boot parameter table or a boot table or a boot config table.
  – In case of **Boot Table Mode**, blocks are read from the I2C, the 4 byte header is stripped, and the data is passed to the boot table processing function. The boot code will parse the boot table and determines the start address. The code then populates the BOOT_ADDRESS register. Once the boot table is completely parsed the boot code branches to the address in the BOOT ADDRESS register and brings the DSP corePAC out of idle and execute the code.
  – In **Config Table Mode**, the data read from the I2C contain configuration tables. Each element in the table consists of three 32 bit fields.
    • This mode is typically used to poke registers needed before boot can be run, or to execute functions from a previously loaded boot.
    • Each entry in the table falls into one of three types.
      – Standard Entry for read-modify-write of an address
      – Branch entry for a function call to the specified address
      – Table Terminate to end and re-run boot
Boot Runtime

I2C Master-Broadcast Mode

• Master-Broadcast: If enabled, the DSP will re-broadcast the boot image loaded from I2C to all passive devices. This is used in case we have multiple DSPs in a same system and one DSP acts as a master driving the other DSPs.
Boot Runtime

I2C Passive Mode

• Passive Mode: the boot ROM operates the I2C device in receiver mode.
  – The header format for passive mode is 19 xx xx yy yy zz zz. Where 19 is the I2C slave address, xxxx is the length, yyyy is Checksum and zzzz is the boot option.
  – The boot option can be again boot parameter table, boot table or a boot config table.