For Stage 1: Handoff, implementation.

Vivado

1. Create blank project
2. Create Block Design
3. Add MPSoC IP
    a. Run block automation – apply board presets
4. Tie both fpd_aclk to pl_clk0
5. Validate Design
6. Create HDL Wrapper
    a. Let Vivado manage
7. Generate Bitstream
8. Export Hardware
    a. Include Bitstream

Vitis Project Creation

1. Create Platform Component
    a. Hardware Design – From the .XSA exported in Vivado step 8
        i. OS - Standalone
        ii. Processor – psu_cortexa53_0
        iii. Generate Boot Artifacts
        iv. Generate PMU Firmware
        v. Target FSBL – psu_cortexa53_0
    b. Build Platform
2. Create Boot Image (note: existing .bif can be imported if this step is being repeated)
    a. From top menu bar
        i. Vitis > Create Boot Image > Zynq Ultrascale+
    b. Remove all entries
    c. Add .elfs – located in each item's 'build' folder
        i. Fsbl.elf
            1. Type - Bootloader
            2. Destination – a53_0
        ii. Pmufw.elf
            1. Type – datafile
            2. Destination – pmu
        iii. .bit
            1. Platform\export\platform\hw\sdt\*.bit
            2. Type – datafile

            3. Destination device - PL
    d. Reorder items in list
        i. FSBL > PMUFW > .bit
    e. Create BOOT.bin
3. Boot from SD card
    a. Set boot switches to off-on-off-on
    b. Place BOOT.bin on SD and insert into board
    c. Boot board with SW7. This will execute program.
    d. To verify, connect to board over UART, restart board with SW6 to view output

We have now demonstrated that we have a working baseline FSBL and PMUFW for the board.

PMUFW – Initial modifications

Key Generation

The ShEF modifications to the PMUFW include hard-coded keys. We can start with generation of all the keys required for this project.

1. Generate AES key for bitstream encryption
    a. Create a .nky file with these contents:

```
Device          xczu1cg;
Key 0           <PASTE_YOUR_64_CHAR_KEY_HERE>;
IV 0            <PASTE_YOUR_24_CHAR_IV_HERE>;
```

    b. Create and replace 64_char_key with output of:
    openssl rand -hex 32
    c. Create and replace 24_char_IV with
    openssl rand -hex 12
2. Generate RSA-4096 Pair
    a. openssl genrsa -out device_key_4096.pem 4096
    b. openssl rsa -in device_key_4096.pem -text -noout > key_dump.txt

PMUFW modifications

1. Copy files from u96 source PMUFW code to our PMUFW application
    a. xpfw_mod_sec.c
    b. xpfw_mod_sec.h
    c. sha3.c
    d. sha3.h
    e. sha512.c
    f. sha512.h

    g. verify.c
    h. sign.c
    i. add_scalar.c
    j. ed25519.h
    k. fe.c
    l. fe.h
    m. fixedint.h
    n. ge.c
    o. ge.h
    p. key_exchange.c
    q. keypair.c
    r. precomp_data.h
    s. sc.c
    t. sc.h
    u. seed.c
2. xpfw_user_startup.c – modify this PMUFW file
    a. #include "xpfw_mod_sec.h"
    b. sec_ipi_mod_init()
        i. place this call at end of file
        ii. for functionality check, can sandwich this line with print debug
3. xpfw_mod_sec.c – modify with keys from key_dump.txt
    a. For these, convert "<hex>:<hex>" to "0x<hex>,0x<hex>"
    b. Replace root_sk with privateExponent
    c. Replace root_mod with modulus
    d. Leave root_pk untouched
    e. If first hex value of modulus is 0x00, key may have been padded by openssl. check number of hex values in key. If total is 513, delete the first 0x00 value.
4. xpfw_mod_sec.c – Contains extensive changes. Gemini has generated a diff for this port:
    a. PMUFW xpfw_mod_sec.c major adjustment

Build full project, create new boot.bin and test.

FSBL

1. dev_key.h
    a. copy from u96 files to our FSBL files.

    b. this file contains hard-coded keys. Copy over root_sk and root_mod from pmufw\xpfw_mod_sec.c
2. xfsbl_main.c - modifications
    a. #include “xfsbl_authentication.h”
       #include “dev_key.h”
    b. + static XSecure_Sha3 csu_sha3;
        i. Add this line after “XFsblPs FsblInstance = ...” ~line 54+
    c. Add major ShEF code block.
        i. Place as final lines of “case XFSBL_STAGE3”
        ii. ~line 249 after:
            FsblStage = XFSBL_STAGE4;
            EarlyHandoff = FsblStatus;
        iii. LEAVE COMMENTED OUT FOR NOW
            1. Code hashes partitions that do not yet exist.
3. xfsbl_hw.h – add definitions
    a. place in “pmu_global” section
    b. + #define OCM_SEC_BUFFER_ADDRESS   (0xFFFFFE00U)
    c. + #define XFSBL_R50_HIGH_ATCM_START_ADDRESS   (0xFFE00000U)
4. xfsbl_error.h – add definitions
    a. place after XFSBL_ERROR_PMU_GLOBAL_REQ_ISO defn. ~line 167
    b. + #define XFSBL_PSU_INIT_COMPLETED   (0x1U)
    c. + #define XFSBL_SECURE_RST_MASK   (0x00000001U)
5. xfsbl_hooks.c – add PMU communication logic
    a. unclear if this is required. May cause issues, so review later
    b. place just before “if (XFSBL_SUCCESS != Status) { “ check. ~line 163

```
+   // Write 1U to PMU GLOBAL general storage register 5 to indicate
+   // PMU Firmware that psu init is completed
+   XFsbl_Out32(PMU_GLOBAL_GLOB_GEN_STORAGE5,
XFSBL_PSU_INIT_COMPLETED);
+
```

6. consider adding temporary print statement early in XFSBL_STAGE4 case to verify that FSBL still functions.
7. Rebuild project. Generate new boot.bin. test

security_kernel port

1. Create new empty application component
2. + create new Domain. Create new domain for the r5_0, standalone
3. Finish creation of application

4. Rebuild Platform component to accommodate new domain addition
5. Components: platform > settings > vitis-comp.json
    a. Locate cortexr5_0 settings, then BSP
    b. Enable xilsecure
6. Rebuild platform component to incorporate new library
7. In Components: security_kernel
    a. Right-click sources
    b. Import > files
    c. Navitage to u96 security_kernel source files and import everything
8. Verify security_kernel is using the correct linker script for memory placements
    a. Components: security_kernel > settings > UserConfig.cmake
    b. Navigate to Linker script file entry and verify it points to the newly-copied file.
9. security_kernel.h – modifications for 2023 Vitis
    a. make this change:

       - #define XSECURE_CSUDMA_DEVICEID   XPAR_XCSUDMA_0_DEVICE_ID

       + #define XSECURE_CSUDMA_DEVICEID XPAR_XCSUDMA_0_BASEADDR
    b.
10. ipi.c – modifications for 2023 Vitis
    a. make this change:

       - intc_config = XScuGic_LookupConfig(XPAR_SCUGIC_0_DEVICE_ID);

       + intc_config = XScuGic_LookupConfig(XPAR_XSCUGIC_0_BASEADDR);
    b. then this:

       - cfg_ptr = XIpiPsu_LookupConfig(XPAR_XIPIPSU_0_DEVICE_ID);

       + cfg_ptr = XIpiPsu_LookupConfig(XPAR_XIPIPSU_0_BASEADDR);
    c. this (just below #include "ipi.h"):

       + XScuGic gic_inst;

       + XIpiPsu ipi_inst;
    d.
11. ipi.h – modifications for 2023 Vitis
    a. make this change:
       ```
       - #define IPI_PMU_PM_INT_MASK_SEND    XPAR_XIPIPS_TARGET_PSU_PMU_0_CH0_MASK
       + #define IPI_PMU_PM_INT_MASK_SEND        XPAR_IPI1_0_IPI_BITMASK
       ```
    b. then this:

       - #define IPI_PMU_PM_INT_MASK_RECV   XPAR_XIPIPS_TARGET_PSU_PMU_0_CH1_MASK

       + #define IPI_PMU_PM_INT_MASK_RECV   XPAR_IPI1_1_IPI_BITMASK
    c. this:

       - XScuGic gic_inst;

       - XIpiPsu ipi_inst;

+ extern XScuGic gic_inst;

+ extern XIpiPsu ipi_inst;

d.

12. main.c – modifications for 2023 Vitis
    a. + #include “sleep.h”
    b. make this change:

       - status = rpu_gic_init(&gic_inst, XPAR_XIPIPSU_0_INT_ID,

       + status = rpu_gic_init(&gic_inst, XPAR_XIPIPSU_0_INTERRUPTS,

       (Xil_ExceptionHandler)rpu_ipi_handler, &ipi_inst);
13. If Vitis did not put all of those imported files into security_kernel\src, then CMake probably didn’t update correctly.
    a. In security_kernel\src\CMakeLists.txt:
    b. Make this change:

```
- aux_source_directory(${CMAKE_SOURCE_DIR} _sources)
+ set(_sources
+     ../fe.c
+     ../ge.c
+     ../ipi.c
+     ../keypair.c
+     ../key_exchange.c
+     ../main.c
+     ../platform.c
+     ../sc.c
+     ../security_kernel.c
+     ../seed.c
+     ../sha3.c
+     ../sign.c
+     ../verify.c
+ )
```

    c. Then this

```
- set_target_properties(${APP_NAME}.elf PROPERTIES LINK_DEPENDS
${CMAKE_SOURCE_DIR}/lscript.ld)

- target_link_libraries(${APP_NAME}.elf -Wl,-T -
Wl,\"${CMAKE_SOURCE_DIR}/lscript.ld\" -L\"${CMAKE_SOURCE_DIR}/\" -
L\"${CMAKE_LIBRARY_PATH}/\" -L\"${USER_LINK_DIRECTORIES}/\" -Wl,--start-group,-
l${_deps} -Wl,--end-group)

+ set_target_properties(${APP_NAME}.elf PROPERTIES LINK_DEPENDS
${CMAKE_CURRENT_SOURCE_DIR}/../lscript.ld)

+ target_link_libraries(${APP_NAME}.elf -Wl,-T -
Wl,\"${CMAKE_CURRENT_SOURCE_DIR}/../lscript.ld\" -
L\"${CMAKE_CURRENT_SOURCE_DIR}/../\" -L\"${CMAKE_LIBRARY_PATH}/\" -
L\"${USER_LINK_DIRECTORIES}/\" -Wl,--start-group,-l${_deps} -Wl,--end-group)
```

security_kernel size issue.

This will compile to something larger than the original lscript.ld allows our code to take up. Current file wants 208 kB, but linker script only giving it 87.5 kB.

R5 OCM is 256 kB total, but our other applications are placing things in this same memory immediately before and after what the security_kernel is allocated. We HAVE to get the file size down.

1. (-0s) - We can set the compiler to optimize for size. This brings the file size down to just 1216 bytes over size limits.
2. In the existing linker script, the stack is allocated 5.25 kB, which Gemini states may be larger than necessary. Trying to reduce.
    a. lscript.ld
    b. make this change:
       - _STACK_SIZE = DEFINED(_STACK_SIZE) ? _STACK_SIZE : 0x1500;
       + _STACK_SIZE = DEFINED(_STACK_SIZE) ? _STACK_SIZE : 0xD00;
    c.

Runtime port

1. Create new application project for runtime on the A53
2. Components: Platform > Settings > vitis-comp.json
    a. Add xilffs library to standalone_a53
    b. Rebuild platform
3. sd_driver.c – modifications for 2023 Vitis
    a. - res = f_read(&fil, (void*)load_addr, fil.fsize, &bytes_read);
       + res = f_read(&fil, (void*)load_addr, f_size(&fil), &bytes_read);
    b. - bitstream_size = fil.fsize;
       + bitstream_size = f_size(&fil);
4. compile

PMUFW xpfw_mod_sec.c major adjustment

the u96 version of xpfw_mod_sec.c is incompatible with modern vitis. Here is the diff to make it compatible:

```
--- a/u96/pmufw/xpfw_mod_sec.c
+++ b/vitis_2/platform/zynqmp_pmufw/xpfw_mod_sec.c
@@ -23,34 +23,28 @@
 #include "xsecure_rsa.h"
 #include "xsecure_sha.h"
 #include "xilfpga_pcap.h"
-#include "xfpga_config.h"
+#include "xilfpga.h"

 #include "xpfw_ipi_manager.h"
 #include "xpfw_mod_sec.h"

+#define XPAR_XCSUDMA_0_DEVICE_ID 0
+
 /* Exponent of private key */
 u8 root_sk[RSA_SIZE] = {
-    0x8f,0xef,0x22,0x8e,0x42,0x6c,0x0b,0x0e,0x67,0x19,0xc7,0xd5,0x7a,0x98,
-    0xe8,0x70,0x09,0x9f,0x04,0xda,0xd6,0xd8,0xcc,0xea,0xa5,0x34,0xbf,0xb0,0x36,
-    0xf0,0x75,0x8f,0xb0,0x12,0x59,0x8e,0x23,0x89,0x3f,0xbf,0xf1,0xbd,0xf1,0x39,
-    0x63,0xa1,0xd4,0xc7,0xea,0xe7,0x6b,0x45,0x3a,0xad,0x1f,0xba,0xc8,0xf5,0xdd,
-    0xcc,0x3c,0x87,0xf7,0x1a,0x24,0xdc,0xb3,0x49,0x58,0x20,0x17,0x85,0xa6,0xf5,
-    0x8e,0x50,0x78,0x20,0x0f,0x15,0x26,0xcd,0xba,0x65,0x05,0x1c,0x28,0x64,0xad,
-    0x3b,0x32,0x40,0x9e,0x26,0x4e,0xb1,0x18,0xd5,0xc7,0x87,0x97,0x3a,0xdd,0xfa,
-    0x02,0x3e,0xbc,0xf3,0x24,0x66,0x9c,0xdd,0xfd,0xb9,0xf7,0xad,0x69,0xab,0x8f,
-    0x08,0x58,0xd1,0x55,0x3e,0x41,0x7f,0x5f,0x8d,0x06,0xf9,0x41,0x6d,0x7b,0x02,
-    0xe3,0xcf,0xdb,0x6b,0xde,0x70,0xee,0xac,0x85,0xd4,0x9c,0xaa,0xe1,0xfc,0x3f,
-    0xc3,0xf1,0x09,0xda,0x89,0x50,0x00,0xe9,0x89,0x5c,0x8e,0x04,0xb8,0x04,0x0f,
-    0x7a,0x96,0x1f,0x63,0x90,0x08,0x7c,0x48,0xee,0x4d,0x85,0x8c,0x69,0xc8,0x9d,
-    0x1b,0x9a,0x10,0xe3,0xd5,0x5b,0xb1,0xed,0xbe,0xd5,0x92,0x32,0x4e,0xa1,0xce,
-    0xcb,0x81,0x04,0xb1,0xf7,0x7f,0xdc,0x89,0x19,0x20,0x13,0x82,0x13,0xf0,0x29,
-    0xb9,0x19,0x6b,0xbd,0xad,0xa9,0x83,0xb3,0x0a,0xa4,0xa8,0xfa,0x0f,0x16,0x08,
-    0x4f,0xb3,0xf9,0x5d,0xbf,0x7c,0xd3,0x0c,0x30,0x9b,0x99,0x97,0x30,0x9c,0xad,
-    0x72,0x1b,0x0d,0x3f,0xfe,0x99,0x62,0xb8,0xc1,0xe1,0x5f,0xde,0x4d,0x0b,0x89,
-    0xcc,0xea,0x42,0xc5,0xcf,0x60,0x43,0xef,0x57,0x92,0x0e,0xf8,0x25,0x17,0xba,
-    0x17,0xd4,0xde,0x7f,0x58,0xe9,0xb9,0x54,0x69,0x28,0xff,0x6f,0x03,0xfd,0x31,
-    0xe5,0x8a,0xe7,0x57,0xa7,0xf6,0x58,0x3f,0x90,0xa1,0xa8,0x29,0x90,0xa1,0x0b,
-    0x99,0x8c,0xb8,0x1b,0x30,0x50,0xf4,0x2f,0x75,0xff,0xb0,0xb8,0x03,0xeb,0x92,
-    0x4b,0xa4,0x10,0xc2,0x09,0x80,0xe3,0x0e,0xe5,0x2e,0x45,0x20,0x64,0x35,0xc4,
-    0x0f,0x2d,0x1d,0xfa,0x28,0xb2,0x7c,0x54,0x0d,0x5c,0x56,0x8f,0xae,0x25,0xd9,
-    0xed,0xe2,0x11,0x60,0x34,0x42,0x94,0x8f,0xa5,0x49,0x12,0x1f,0xf6,0x33,0x95,
-    0x58,0xc2,0x37,0xa9,0x93,0xc1,0x92,0x3d,0xa5,0xf6,0x87,0xc9,0xa9,0xc9,0x50,
-    0x8d,0x86,0x69,0xf3,0x14,0xfe,0x72,0xb6,0x9f,0xf9,0x88,0xd8,0xc8,0xc3,0xfa,
-    0xa8,0x9b,0x19,0x8f,0x26,0xb3,0xc4,0x2a,0x66,0x29,0xd9,0x06,0x0e,0x2b,0xca,
-    0xff,0x09,0x18,0xd5,0x51,0xef,0x94,0x1a,0xb1,0x75,0xca,0xbb,0x7e,0xc9,0x32,
-    0x15,0xda,0x1a,0xf8,0x02,0x85,0x09,0x6b,0x18,0xce,0x2d,0xaf,0xf4,0xe0,0xe1,
-    0x68,0x3a,0x71,0x12,0xc9,0x11,0x17,0x70,0x76,0xab,0x18,0x29,0x7d,0x51,0x81,
-    0xdc,0xe4,0x4c,0x04,0xd4,0x3c,0x32,0xea,0xe7,0x90,0xcd,0x28,0x08,0xf4,0x2f,
-    0xe7,0xe0,0xdd,0x69,0x0e,0xb8,0xae,0x2b,0x0c,0x42,0x87,0x8f,0x42,0x9f,0x2f,
-    0x9a,0x61,0x9a,0x75,0x89,0xd8,0xbe,0xf3,0x02,0x0d,0x08,0x17,0xf4,0x99,0x62,
-    0x64,0x27,0x81,0x78,0xe2,0xfa,0x81,0xb6,0x9f,0x26,0xb6,0x5f,0xb9,0x91,0xf8,
-    0xbc,0x61,0xf1
+    0x3e,0x85,0xd6,0x2d,0x32,0x6c,0xe4,0xb8,0xc0,0x77,0xbe,0x58,0x8a,0xf7,0xfc,
+    0xbd,0x48,0x47,0x13,0x8c,0xcf,0x62,0x22,0xcb,0xa0,0x97,0xa7,0xa4,0x0c,0xf7,
+    0x26,0x43,0xf8,0x1b,0x04,0x8a,0xf8,0xdf,0x8e,0x1e,0xe3,0x20,0x5d,0x08,0xd8,
+    0x46,0x9f,0x62,0x53,0x3b,0x11,0x4b,0x74,0x36,0x28,0x87,0x6b,0x56,0x55,0x59,
+    0xbd,0xfb,0x02,0x8c,0xfb,0x70,0xbc,0xd9,0x67,0x30,0xd2,0x5a,0x3a,0x5e,0x44,
+    0x8b,0xe3,0x0a,0x8f,0xb3,0x25,0xa1,0x8e,0xdf,0xe2,0x87,0x1c,0x5a,0xd3,0x58,
+    0x58,0xc9,0xdb,0x12,0x6b,0x56,0xef,0x33,0xab,0x0b,0x21,0x5d,0xbd,0x5a,0xe4,
+    0xf0,0xdf,0xe6,0x26,0xa8,0xb6,0x4d,0x31,0x41,0x56,0x6f,0x1c,0x66,0x58,0xa1,
+    0x8d,0xcb,0x23,0xa4,0x16,0x1c,0xb9,0x59,0x59,0x02,0x54,0x28,0x5b,0x43,0x9c,
+    0xfb,0xf1,0x1f,0x73,0x46,0x45,0x0a,0xf7,0x6b,0x02,0x12,0xb1,0xaa,0x72,0x1b,
+    0x10,0x19,0xd8,0x78,0x23,0xf3,0x82,0xb2,0xce,0x81,0xb1,0x16,0x42,0x38,0x69,
+    0xf4,0xde,0x9b,0x83,0xd4,0xc8,0x7e,0xc0,0x7f,0xf0,0x38,0xd8,0xce,0xda,0xaf,
+    0x4d,0x0b,0x66,0x9e,0xd1,0x8c,0x08,0x39,0x70,0xf0,0x33,0x8c,0x20,0xa5,0xf8,
+    0xc4,0xbe,0x81,0x67,0x60,0x16,0xc2,0x69,0x10,0xfb,0x98,0xda,0x01,0xa0,0xab,
+    0xd2,0xd6,0xf5,0x9c,0xf4,0x74,0x59,0xa1,0x6f,0xca,0x5a,0x01,0x5f,0xf2,0x8d,
+    0xfd,0x6e,0xc8,0x87,0xc9,0x15,0xcd,0x6a,0x58,0xee,0x87,0x4c,0x6e,0x5e,0x22,
+    0x92,0x45,0x0b,0x73,0x2b,0xc0,0xb2,0x59,0x8b,0x6a,0x97,0xbc,0xf4,0x77,0xd0,
+    0xce,0x94,0x89,0xc9,0xba,0x6e,0x16,0xc0,0xca,0x15,0xe9,0xeb,0x7b,0x10,0xa4,
+    0xa3,0xbd,0x3d,0x6e,0x52,0x26,0xef,0xac,0x01,0x72,0xb5,0x3d,0x4e,0x65,0xd0,
+    0xc9,0xaf,0x8d,0x6e,0xde,0x63,0x63,0x66,0x0d,0x88,0x9a,0x9a,0x5e,0xc6,0xf4,
+    0x00,0x2c,0x13,0x7d,0x5a,0x19,0x95,0x6d,0x27,0xf1,0x60,0x28,0x3e,0x6c,0x84,
+    0x8c,0xc1,0xd5,0x74,0x7a,0x14,0x6c,0x78,0x72,0x1f,0x86,0x75,0x5e,0xdc,0x70,
+    0xa0,0x1a,0x07,0xd1,0xda,0x8a,0xa3,0xc4,0x75,0xe2,0x40,0xba,0x1e,0xf1,0x92,
+    0x9e,0x56,0x41,0xb2,0x23,0xb5,0xdc,0xab,0x2e,0x19,0xf8,0x38,0x88,0x2c,0x8e,
+    0xae,0x22,0x0a,0x4e,0x22,0xf2,0x30,0x84,0xb1,0x82,0x2f,0xc7,0xb7,0x01,0x6f,
+    0x54,0x08,0xb8,0x69,0x91,0xfd,0x4d,0xc3,0x0c,0xcd,0x69,0xd9,0x82,0x38,0x5e,
+    0x34,0x5d,0x84,0xcc,0xf3,0x1e,0x4d,0xa6,0x43,0x77,0xc5,0x2c,0x99,0x30,0xfe,
+    0x39,0x86,0xd6,0xf6,0x1c,0xe2,0x6d,0xcf,0xdc,0x5b,0xe4,0x2a,0xca,0xe8,0x07,
+    0xfd,0xd2,0x46,0xd6,0x19,0x31,0xd7,0xdc,0xc6,0x23,0x9c,0x02,0x7c,0xa8,0x50,
+    0x2a,0xdd,0xc9,0xcb,0xc8,0x64,0xb3,0xff,0x28,0x66,0xda,0xa5,0x78,0x67,0xb8,
+    0x47,0xdc,0x01,0xad,0xd1,0xaa,0x13,0x4b,0xe7,0xc9,0xc2,0xb4,0x1e,0x1b,0xbc,
+    0x94,0x56,0xe2,0x10,0x85,0x86,0x15,0xa6,0x1e,0xd1,0xe9,0x4c,0x3e,0x80,0xb0,
+    0xd0,0xb4,0xe6,0x45,0xed,0x94,0x2d,0x8f,0x2e,0x29,0x89,0xd8,0x6a,0x98,0x6d,
+    0x40,0x3c,0xee,0xc8,0x67,0x7c,0x77,0xdd,0x7e,0x16,0x32,0x32,0x3b,0xda,0x2c,
+    0x24,0x4d
 };

 /* Exponent of Public key */
@@ -58,32 +52,32 @@

 /* Modulus */
 u8 root_mod[RSA_SIZE] = {
-    0xc0,0x36,0x58,0xd5,0x05,0x9f,0xf6,0x9f,0x8c,0xb2,0x9a,0x9c,0x36,0x68,
-    0xb8,0x2a,0x28,0xc4,0x36,0x4d,0x4e,0x33,0xce,0xdb,0xcf,0x2b,0xdd,0x38,0xda,
-    0x01,0xdc,0x11,0xf9,0x4a,0xba,0x6a,0x70,0xeb,0xe1,0xc0,0x48,0xc4,0xa7,0x21,
-    0x43,0xa4,0x66,0xf5,0xc0,0xdb,0x74,0x6e,0xf3,0xd8,0xfe,0x6f,0x42,0x4b,0x1c,
-    0x13,0x40,0x0e,0xf5,0x6e,0xa1,0x13,0x8d,0x45,0x4f,0xfc,0x3e,0x0a,0xe2,0x48,
-    0x83,0xbc,0x2d,0xbd,0x79,0xf1,0xe2,0x42,0xbc,0x03,0x6b,0x5c,0x11,0x1d,0x38,
-    0x66,0x27,0xbf,0x7c,0x55,0x1d,0x7a,0xe0,0x01,0xd6,0x8c,0x15,0xcb,0xe7,0xa7,
-    0x87,0xf1,0x79,0x2d,0x23,0xab,0x20,0x18,0x2d,0x07,0x1a,0x04,0x23,0x6f,0x52,
-    0x55,0xcc,0x6d,0xd3,0x8d,0xdb,0xce,0x83,0x24,0x71,0xca,0xc0,0xca,0xa0,0xca,
-    0xce,0xb5,0x7c,0x26,0x1c,0x3c,0x3e,0xaa,0xbe,0xed,0x16,0x82,0xe7,0x5a,0x6b,
-    0x75,0x74,0xe5,0xff,0xce,0xcf,0xea,0x99,0x95,0x28,0x7f,0x34,0x0c,0xa6,0x0b,
-    0xb8,0x2b,0x09,0x00,0x7a,0x15,0xb9,0x05,0xcb,0x13,0xa7,0x94,0xea,0x0e,0x04,
-    0x11,0xe5,0xfb,0xb0,0xd3,0xa8,0x7c,0x68,0x7f,0xb6,0xf9,0xcd,0x62,0x67,0x1a,
-    0xb6,0xfd,0x84,0x9d,0xce,0xb9,0x99,0x83,0x60,0xe2,0x95,0x33,0x43,0x8d,0xba,
-    0x50,0xc4,0x29,0x6f,0x33,0x38,0x88,0x31,0x51,0x4f,0xc9,0xbe,0x26,0x04,0x80,
-    0xfa,0xb3,0x9d,0xdb,0x72,0xab,0x7c,0x98,0x01,0x0a,0xcc,0x8a,0x04,0x3c,0x2a,
-    0x8f,0x39,0x5b,0x2d,0x7c,0x78,0x71,0x6f,0xc2,0x5f,0xc8,0x3e,0xd4,0xc1,0x55,
-    0xa7,0xd9,0x96,0x2d,0x08,0xc0,0x0a,0x99,0x54,0x87,0x73,0x6c,0x9b,0x6b,0x65,
-    0xc2,0xd9,0x8b,0x1c,0xc2,0x62,0x9b,0xad,0x49,0x81,0xaa,0x02,0xc2,0x4f,0x7b,
-    0xc1,0xe6,0x09,0x4c,0xe0,0x9d,0x05,0xf1,0x60,0x12,0x88,0xaa,0x6b,0xaa,0xd7,
-    0xb4,0x40,0xc7,0xa2,0x53,0x37,0xcb,0x22,0xcd,0x89,0x0d,0xc1,0x6b,0x24,0x69,
-    0x7f,0x8b,0xcd,0x07,0x52,0xa3,0xa4,0x68,0xe7,0xc5,0xdd,0x9c,0x84,0x1a,0xa4,
-    0xa3,0xda,0x3b,0x4d,0xd7,0xaf,0x5e,0xe3,0x69,0x6c,0x76,0x4b,0xf8,0xa7,0xd4,
-    0xa6,0x69,0xc9,0x98,0xf8,0x78,0x50,0xcd,0x5f,0xb1,0x86,0x62,0xb0,0x79,0x83,
-    0x89,0xa0,0x13,0xf4,0xe3,0x25,0x6b,0xe0,0xcf,0x65,0xa7,0xa4,0x31,0x29,0xe0,
-    0xc5,0x94,0x4a,0xaf,0x82,0x00,0x1c,0x45,0x99,0xd6,0x4c,0x8c,0xcf,0xf9,0x84,
-    0x21,0x56,0x63,0x17,0x8d,0x6c,0x2e,0xde,0x36,0x97,0x09,0x72,0x8e,0xd0,0xe2,
-    0xe3,0xbd,0x18,0x29,0x2d,0x3b,0xf9,0x43,0x0a,0x50,0xb5,0x4d,0xe9,0xb7,0x93,
-    0x71,0x76,0x8a,0xf7,0x52,0x3a,0xec,0x03,0xd0,0x5c,0x2f,0xce,0x7a,0x7e,0xb8,
-    0x31,0xc0,0x85,0x6b,0xaf,0x77,0xda,0x79,0x22,0x20,0xc2,0xd6,0x43,0x91,0x2e,
-    0xa9,0x31,0x3a,0xc8,0x76,0xde,0xd9,0x7d,0xc2,0xde,0x2a,0x8e,0xc5,0x0e,0x25,
-    0x96,0x2e,0x31,0xec,0xff,0x0d,0x3b,0x0f,0x8e,0x83,0x81,0xe7,0xbe,0x83,0xb6,
-    0xcd,0x9b,0x49,0xad,0xe8,0x6c,0xea,0x0a,0xc5,0x23,0xfd,0x62,0x67,0x32,0xe5,
-    0x1d,0xbe,0x6a,0x61,0x20,0xcf,0x08,0xdf,0x6e,0x1a,0xe5,0x17,0x57,0x6b,0xf0,
-    0x78,0x2b,0xd5
+    0xdd,0xe2,0xa6,0x16,0x7e,0x51,0x23,0xbf,0x23,0xc2,0x28,0x11,0x18,0xce,
+    0xb6,0xeb,0x98,0x33,0xbb,0x92,0xa3,0x25,0xd5,0x5a,0xff,0xcf,0x51,0x62,0x0e,
+    0xee,0xc9,0x30,0x66,0x3c,0x7a,0x3d,0x6e,0x48,0xb5,0x85,0xc6,0xf5,0x82,0x1d,
+    0xee,0xff,0x42,0xb4,0xb8,0x8e,0xca,0xad,0x8d,0x6f,0x33,0xb0,0x09,0xf3,0x2c,
+    0x3c,0x8a,0xdb,0x9e,0x3d,0xb7,0x4d,0x02,0xb4,0x84,0xd5,0x8f,0x73,0x55,0x2a,
+    0x24,0x19,0xaa,0xf2,0x0c,0xf8,0xa8,0x4d,0x23,0x69,0x40,0x0b,0x84,0xa6,0x66,
+    0xee,0x93,0x9c,0x85,0x32,0x02,0x0a,0xee,0x63,0x97,0x92,0x0c,0x3d,0x24,0xd7,
+    0x74,0xc2,0xdd,0x26,0x5d,0x8e,0x7a,0x99,0x65,0xa2,0x41,0xcc,0x46,0xc6,0x55,
+    0x08,0x16,0x77,0x8a,0xfd,0x3c,0xe8,0xd4,0x03,0xdb,0xd7,0x89,0x98,0x70,0x49,
+    0x4f,0x06,0x39,0xcc,0x68,0xb6,0xc4,0xba,0x18,0xf3,0xc7,0xaf,0xa6,0x21,0x84,
+    0x3a,0x6b,0x72,0x8d,0xe5,0xa3,0xa4,0x8b,0x36,0x58,0x16,0xf3,0x65,0xf3,0x16,
+    0xe3,0xb9,0x6b,0x6d,0x25,0x64,0x37,0x8e,0x02,0x0f,0xac,0xb9,0x7e,0xd3,0x10,
+    0x7c,0x18,0x0c,0x23,0x93,0x0a,0xe4,0xd7,0x0e,0x79,0x8a,0xc1,0xd4,0xb9,0x49,
+    0xd6,0x7a,0x49,0x15,0xe3,0x28,0xa9,0x28,0x6f,0x47,0xbf,0x7b,0x38,0x06,0x78,
+    0x27,0x32,0x51,0x6f,0x5b,0x57,0x64,0xd2,0xd8,0xbf,0x3e,0xb9,0xab,0xe1,0x9c,
+    0xbe,0x3f,0xea,0xca,0x2d,0xf6,0x83,0x25,0x95,0x5f,0xbb,0xb0,0x3f,0x83,0x7e,
+    0x89,0x62,0x36,0x01,0x31,0xf6,0xa7,0xa0,0xa8,0x18,0x1f,0x6d,0xb7,0x47,0xa3,
+    0x78,0xa1,0x86,0x74,0x04,0x0d,0x4b,0x19,0xe6,0x82,0xb5,0xf0,0x2c,0xf7,0xae,
+    0xdf,0x75,0x94,0x64,0x5f,0x2c,0x16,0x52,0x75,0x4b,0x77,0x11,0x82,0xad,0x84,
+    0x81,0x4b,0x52,0x32,0x0e,0xbc,0x7b,0x14,0x19,0x4b,0x87,0xd9,0x98,0xf9,0xb0,
+    0x0f,0x8b,0x68,0xe8,0x02,0x5a,0x10,0x5d,0x83,0x31,0x02,0x78,0xc5,0xa4,0xc6,
+    0x7a,0x0b,0x1d,0x94,0xc8,0x8d,0xa5,0x69,0x1a,0x86,0xe0,0x95,0x91,0xd5,0x43,
+    0x6d,0xf3,0x5f,0xe9,0x6a,0x6e,0x55,0x97,0xa9,0xa7,0x83,0x69,0x56,0xe3,0xad,
+    0xbb,0x43,0xbe,0x68,0x65,0x94,0x3d,0x7b,0xb4,0x50,0x78,0x78,0x29,0x35,0xbd,
+    0xb4,0xe3,0x8d,0xaa,0x2e,0xf2,0x21,0x3c,0x0e,0x0c,0xe6,0x59,0x67,0x43,0xa7,
+    0xde,0xf7,0x0b,0x9a,0xc2,0x05,0x18,0x3f,0x22,0xda,0x07,0x85,0x69,0x6b,0xd6,
+    0x30,0x27,0x52,0x2d,0x54,0x3e,0x4d,0xc1,0x77,0x8b,0xe9,0x47,0xd9,0x75,0x3c,
+    0x6f,0x6d,0x97,0x09,0x65,0x16,0xd0,0x24,0x83,0x35,0x3d,0x4a,0x99,0x81,0x57,
+    0x10,0xcf,0xa9,0x15,0x62,0x05,0x49,0x8e,0xf0,0x28,0xd7,0x51,0xf5,0xd3,0x11,
+    0x75,0x91,0x42,0xc4,0xed,0xa6,0x2f,0x08,0xcb,0x3e,0xf9,0x50,0x4d,0xfb,0xf1,
+    0xe0,0xc1,0xc0,0x3d,0x3f,0x9f,0x7b,0x59,0xee,0xe6,0xc2,0xa0,0x0b,0xff,0x8f,
+    0xf7,0xe8,0xe8,0xaa,0x02,0x48,0xa3,0xa9,0x55,0xcc,0xef,0x2d,0xa5,0x6a,0x10,
+    0xf1,0x95,0x18,0xa8,0xd0,0x7a,0x0e,0x9c,0xf6,0xf4,0x39,0xa7,0xe9,0x18,0x41,
+    0xd0,0x11,0x3e,0x83,0x3f,0x92,0x70,0x2f,0x00,0xa9,0x2e,0xd0,0x04,0x5a,0x62,
+    0xf6,0x40,0xa1
 };

 /* Hash with PKCS padding */
@@ -219,7 +213,7 @@

     //Hash the attestation key with SHA3 KECCAK
     XCsuDma_Config *csu_config;
-    csu_config = XCsuDma_LookupConfig(XPAR_PSU_CSUDMA_DEVICE_ID);
+    csu_config = XCsuDma_LookupConfig(XPAR_XCSUDMA_0_DEVICE_ID);
     if (csu_config == NULL){
         XPfw_Printf(DEBUG_ERROR, "PMU: Failed to configure CSU\r\n");
         return;
@@ -328,68 +322,76 @@
  * the bitstream binary loaded into memory.
  */
 static void sec_load_bitstream(){
-    u32 status;
-    s32 fpga_status;
-    u8 bitstream_digest[48];
-    u32 msg_buf[8]; //Buffer to hold message to R5
-    u32 resp_buf[2] = {0};
-    u32 i;
-    u32 bytes_sent = 0;
-
-    if(bitstream_size == 0 || bitstream_addr == NULL){
-        XPfw_Printf(DEBUG_ERROR, "PMU: Bitstream address or size\r\n");
-        return;
-    }
-
-
-    //XPfw_Printf(DEBUG_DETAILED, "PMU: Loading bitstream from address 0x%08x\r\n",
-    //        bitstream_addr);
-    //XPfw_Printf(DEBUG_DETAILED, "PMU: bitstream size %d\r\n", bitstream_size);
-
-//    for(i = 0; i < bitstream_size; i++){
-//        XPfw_Printf(DEBUG_DETAILED, "%02x", bitstream_addr[i]);
-//    }
-//    XPfw_Printf(DEBUG_DETAILED, "\r\n");
-
-    //Hash the bitstream first with SHA3
-    XCsuDma_Config *csu_config;
-    csu_config = XCsuDma_LookupConfig(XPAR_PSU_CSUDMA_DEVICE_ID);
-    if (csu_config == NULL){
```

```
-        XPfw_Printf(DEBUG_ERROR, "PMU: Failed to configure CSU\r\n");
-        return;
-    }
-    status = XCsuDma_CfgInitialize(&csu_dma, csu_config, csu_config->BaseAddress);
-    if (status != XST_SUCCESS){
-        XPfw_Printf(DEBUG_ERROR, "PMU: Failed to initialize CSU\r\n");
-        return;
-    }
-    XSecure_Sha3Initialize(&secure_sha3, &csu_dma);
-    if (status != XST_SUCCESS){
-        XPfw_Printf(DEBUG_ERROR, "PMU: Failed to initialize SHA3\r\n");
-        return;
-    }
-    XSecure_Sha3Digest(&secure_sha3, bitstream_addr, bitstream_size, bitstream_digest);
-
-    //Load the bitstream through PCAP.
-    fpga_status = XFpga_PL_BitSream_Load(bitstream_addr, 0, 0);
-    if(fpga_status == XFPGA_SUCCESS){
-        XPfw_Printf(DEBUG_DETAILED, "PMU: PL Configuration successful\r\n");
-    }
-    else{
-        XPfw_Printf(DEBUG_DETAILED, "PMU: PL Configuration failed\r\n");
-    }
-
-    //Send an IPI to the sender (R5_0) containing the hash of the bitstream
-//    XPfw_Printf(DEBUG_DETAILED, "PMU: Bitstream hash is 0x");
-//    for(i = 0; i < 48; i++){
-//        XPfw_Printf(DEBUG_DETAILED, "%02x", bitstream_digest[i]);
-//    }
-//    XPfw_Printf(DEBUG_DETAILED, "\r\n");
-
-
-
+    u32 status;
+    s32 fpga_status;
+    u8 bitstream_digest[48];
+    u32 msg_buf[8];
+    u32 resp_buf[2] = {0};
+    u32 bytes_sent = 0;
+
+    // NEW: Structure for 2023.2 XilFPGA API
+    XFpga XFpgaInstance = {0};
+
+    if(bitstream_size == 0 || bitstream_addr == NULL){
+        XPfw_Printf(DEBUG_ERROR, "PMU: Bitstream address or size error\r\n");
+        return;
+    }
+
+    // 1. Hash the bitstream
+    XCsuDma_Config *csu_config;
+    csu_config = XCsuDma_LookupConfig(XPAR_XCSUDMA_0_DEVICE_ID);
+    if (csu_config == NULL){
+        XPfw_Printf(DEBUG_ERROR, "PMU: Failed to configure CSU\r\n");
+        return;
+    }
+    status = XCsuDma_CfgInitialize(&csu_dma, csu_config, csu_config->BaseAddress);
+    if (status != XST_SUCCESS){
+        XPfw_Printf(DEBUG_ERROR, "PMU: Failed to initialize CSU\r\n");
+        return;
+    }
+    XSecure_Sha3Initialize(&secure_sha3, &csu_dma);
+
+    // Cast to (u8*) to fix volatile warning
+    XSecure_Sha3Digest(&secure_sha3, (u8*)bitstream_addr, bitstream_size,
bitstream_digest);
+
+    // 2. Load the Bitstream (FIXED FOR 2023.2 API)
+    // Initialize the XilFPGA library
+    fpga_status = XFpga_Initialize(&XFpgaInstance);
+    if (fpga_status != XST_SUCCESS) {
+         XPfw_Printf(DEBUG_ERROR, "PMU: XFpga Init Failed\r\n");
+         return;
+    }
+
+    /*
+     * API: XFpga_BitStream_Load(InstancePtr, Addr, KeyAddr, Size, Flags)
+     * KeyAddr = 0 (We are not using an external user key here)
+     * Flags = 0 (Implies XFPGA_FULL_BITSTREAM)
+     */
+    fpga_status = XFpga_BitStream_Load(&XFpgaInstance,
+                                       (UINTPTR)bitstream_addr,
+                                       0,
+                                       bitstream_size,
+                                       0);
+
+    if(fpga_status == XFPGA_SUCCESS){
+        XPfw_Printf(DEBUG_DETAILED, "PMU: PL Configuration successful\r\n");
+    }
+    else{
+        XPfw_Printf(DEBUG_DETAILED, "PMU: PL Configuration failed %d\r\n",
fpga_status);
+    }
+
+    // 3. Send Response
     while(bytes_sent < SHA3_SIZE){
         msg_buf[1] = IPI_BITSTREAM_HASH_MASK;
+        // Cast to (u8*) to fix volatile warning
         memcpy(&msg_buf[2], &bitstream_digest[bytes_sent], 16);

         status = XPfw_IpiWriteMessage(sec_ipi_mod_ptr, IPI_PMU_0_IER_RPU_0_MASK,
-                msg_buf, 8);
+                 msg_buf, 8);
         if(status != XST_SUCCESS){
-            XPfw_Printf(DEBUG_ERROR, "PMU: IPI Write Message Failed \r\n");
-            return;
+            return;
         }
         status = XPfw_IpiTrigger(IPI_PMU_0_IER_RPU_0_MASK);
         if(status != XST_SUCCESS){
-            XPfw_Printf(DEBUG_ERROR, "PMU: IPI Trigger failed \r\n");
-            return;
+            return;
         }
         status = XPfw_IpiPollForAck(IPI_PMU_0_IER_RPU_0_MASK, (~0));
         if(status != XST_SUCCESS){
-            XPfw_Printf(DEBUG_ERROR, "PMU: IPI Poll for Ack Failed \r\n");
-            return;
+            return;
         }
         status = XPfw_IpiReadResponse(sec_ipi_mod_ptr, IPI_PMU_0_IER_RPU_0_MASK,
                 resp_buf, 2);
         if(status != XST_SUCCESS){
-            XPfw_Printf(DEBUG_ERROR, "PMU: IPI Read Response failed \r\n");
-            return;
+            return;
         }


         //Check that the expected reply matches
         if(resp_buf[1] != IPI_BITSTREAM_HASH_MASK){
-            XPfw_Printf(DEBUG_ERROR, "PMU: RPU failed to ack hash command\r\n");
-            return;
+            return;
         }
         bytes_sent += 16;
     }
@@ -421,11 +423,17 @@

     //XPfw_Printf(DEBUG_DETAILED, "PMU: Received command 0x%08x", cmd);
     if(cmd == IPI_BITSTREAM_HASH_MASK){
-        //Load bitstream addr and size into global variables
-        memcpy(&bitstream_addr, &payload[2], 4);
-        memcpy(&bitstream_size, &payload[3], 4);
+        // Fix: Read into temp vars to avoid volatile warnings in memcpy
+        u32 temp_addr, temp_size;
+        memcpy(&temp_addr, &payload[2], 4);
+        memcpy(&temp_size, &payload[3], 4);
+
+        bitstream_addr = (u8*)(UINTPTR)temp_addr;
+        bitstream_size = temp_size;

         //XPfw_Printf(DEBUG_DETAILED, "PMU: Received FPGA Program cmd \r\n");

         //Schedule the task to load the bitstream into FPGA
         status = XPfw_CoreScheduleTask(mod_ptr, 0U, sec_load_bitstream);
     }
@@ -438,7 +446,7 @@
             XPfw_Printf(DEBUG_ERROR, "PMU:Error: invalid index for cert hash\r\n");
             return;
         }
-        memcpy(&cert_hash[start_index], &payload[2], 16);
+        memcpy((void*)&cert_hash[start_index], &payload[2], 16);

         //Check if the full attestation PK has been received.
         if(start_index == (u16)32U){
```

xfsbl_main.c major ShEF segment

```
// Hash the security kernel partition
u8 kernel_hash[XFSBL_HASH_TYPE_SHA3] __attribute__ ((aligned (4))) = {0};

u32 kernel_length;
const XFsblPs_PartitionHeader * partition_header;
PTRSIZE kernel_load_addr;

//Partition in ATCM
partition_header = &FsblInstance.ImageHeader.PartitionHeader[4];
kernel_length = partition_header->TotalDataWordLength * 4U;
kernel_load_addr = ((PTRSIZE)(partition_header->DestinationLoadAddress)) +
XFSBL_R50_HIGH_ATCM_START_ADDRESS;

XFsbl_Printf(DEBUG_GENERAL, "Partition Address: %016x\r\n", kernel_load_addr);
XFsbl_Printf(DEBUG_GENERAL, "Partition Length: %08d\r\n", kernel_length);

XSecure_Sha3Initialize(&csu_sha3, &CsuDma);
XSecure_Sha3Update(&csu_sha3, (u8*)kernel_load_addr, kernel_length);

//Partition in OCM
partition_header = &FsblInstance.ImageHeader.PartitionHeader[5];
kernel_length = partition_header ->TotalDataWordLength * 4U;
kernel_load_addr = ((PTRSIZE)(partition_header->DestinationLoadAddress));

XFsbl_Printf(DEBUG_GENERAL, "Partition Address: %016x\r\n", kernel_load_addr);
XFsbl_Printf(DEBUG_GENERAL, "Partition Length: %08d\r\n", kernel_length);

XSecure_Sha3Update(&csu_sha3, (u8*)kernel_load_addr, kernel_length);

//Partition in DDR
partition_header = &FsblInstance.ImageHeader.PartitionHeader[6];
kernel_length = partition_header ->TotalDataWordLength * 4U;
kernel_load_addr = ((PTRSIZE)(partition_header->DestinationLoadAddress));

XFsbl_Printf(DEBUG_GENERAL, "Partition Address: %016x\r\n", kernel_load_addr);
XFsbl_Printf(DEBUG_GENERAL, "Partition Length: %08d\r\n", kernel_length);

XSecure_Sha3Update(&csu_sha3, (u8*)kernel_load_addr, kernel_length);

XSecure_Sha3Finish(&csu_sha3, kernel_hash);
```

```
//Write the kernel hash to OCM
XFsbl_Printf(DEBUG_GENERAL,"Kernel Hash: ");
int i;
for (i = 0; i < XFSBL_HASH_TYPE_SHA3; i++){
    XFsbl_Printf(DEBUG_GENERAL, "%02x", kernel_hash[i]);
    Xil_Out8(OCM_SEC_BUFFER_ADDRESS + i, kernel_hash[i]);
}

//Generate the keygen seed using the kernel hash and the device key
XSecure_Sha3Initialize(&csu_sha3, &CsuDma);
XSecure_Sha3Update(&csu_sha3, kernel_hash, XFSBL_HASH_TYPE_SHA3);
XSecure_Sha3Update(&csu_sha3, root_sk, 512);
XSecure_Sha3Update(&csu_sha3, root_mod, 512);

XSecure_Sha3Finish(&csu_sha3, kernel_hash);

XFsbl_Printf(DEBUG_GENERAL,"\r\nKeygen Seed: ");

//Truncate the SHA3-384 output to SHA3-256
for (i = 0; i < 32; i++){
    XFsbl_Printf(DEBUG_GENERAL, "%02x", kernel_hash[i]);
    Xil_Out8(OCM_SEC_BUFFER_ADDRESS + XFSBL_HASH_TYPE_SHA3 + i, kernel_hash[i]);
}
```