UG0891
User Guide
Hello FPGA Libero Design

Microsemi®

a Microchip company

a MICROCHIP company

Microsemi Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA
Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996
Email: sales.support@microsemi.com
www.microsemi.com

Microsemi makes no warranty, representation, or guarantee regarding the information contained herein or the suitability of its products and services for any particular purpose, nor does Microsemi assume any liability whatsoever arising out of the application or use of any product or circuit. The products sold hereunder and any other products sold by Microsemi have been subject to limited testing and should not be used in conjunction with mission-critical equipment or applications. Any performance specifications are believed to be reliable but are not verified, and Buyer must conduct and complete all performance and other testing of the products, alone and together with, or installed in, any end-products. Buyer shall not rely on any data and performance specifications or parameters provided by Microsemi. It is the Buyer's responsibility to independently determine suitability of any products and to test and verify the same. The information provided by Microsemi hereunder is provided "as is, where is" and with all faults, and the entire risk associated with such information is entirely with the Buyer. Microsemi does not grant, explicitly or implicitly, to any party any patent rights, licenses, or any other IP rights, whether with regard to such information itself or anything described by such information. Information provided in this document is proprietary to Microsemi, and Microsemi reserves the right to make any changes to the information in this document or to any products and services at any time without notice.

About Microsemi

Microsemi, a wholly owned subsidiary of Microchip Technology Inc. (Nasdaq: MCHP), offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Learn more at www.microsemi.com.

Contents

Figures

Tables

1 Revision History

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the most current publication.

1.1 Revision 1.0

This is the first publication of the document.

2 Hello FPGA

Hello FPGA is a low cost, compact-sized, and a feature-rich FPGA kit based on the non-volatile, Flash-based, and low-power SmartFusion2 SoC FPGA (M2S010) from Microchip. SmartFusion2 SoC FPGAs offer more resources in low density with a complete Microcontroller Subsystem that includes a 166 MHz ARM Cortex M3 processor with an Embedded Trace Macrocell (ETM) and Instruction Cache, embedded flash, and extensive peripherals including CAN, TSE, and USB.

An ultra-low power Flash*Freeze feature of SmartFusion2 SoC FPGAs allows the retention of design while maintaining the I/O state for low-power applications. A single pin interrupt puts the device in the Flash*Freeze mode and brings up these FPGAs to operational modes (within ~13 microseconds).

These FPGAs are ideal for general purpose functions such as Gigabit Ethernet or dual PCI Express control planes, bridging functions, input/output (I/O) expansion and conversion, video/image processing, system management and secure connectivity. These FPGAs are used by customers in Communications, Industrial, Medical, Defense, and Aviation markets.

The Hello FPGA kit is designed for FPGA beginners and enthusiasts. The Hello FPGA kit is ideal for developing control logic and data acquisition, image processing, signal processing, and artificial intelligence applications.

The kit supports the measurement of the live FPGA core power consumption while running these designs. The kit also allows users to freeze the design. The Hello FPGA kit includes a Microchip PIC32 MCU that it is used to program the SmartFusion2 SoC FPGA, monitor power, and general-purpose functions.

The Hello FPGA Kit demonstrates the following features of the SmartFusion2 SoC FPGA:

- DSP functions: FIR Filter.
- Parallel Processing and Complex functions: Artificial Intelligence (real time hand-written digit recognition).
- Low-Power: Shows power consumption in operational and Flash*Freeze modes.
- Instant ON: Highlighting fast wake-up from the Flash*Freeze mode to fully functional mode.

This user guide describes the following designs, which can be run on the kit:

- Video Demo Design, page 4
- Digit Recognition (AI) Demo Design, page 13
- DSP (FIR) Filter Demo Design, page 18

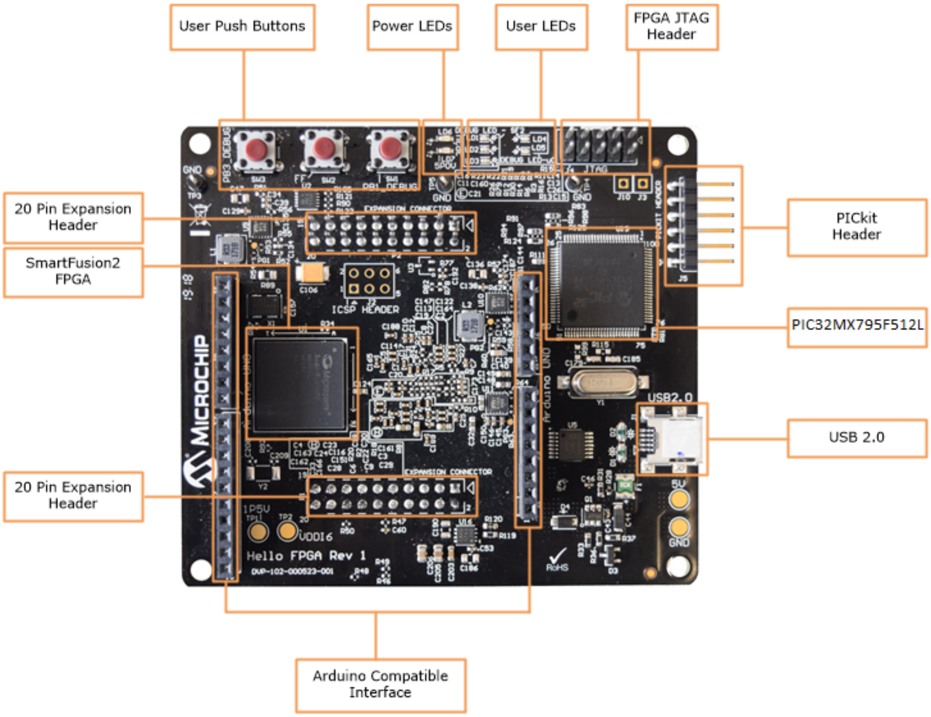
These designs are created using Microchip's Libero SoC Design Software. The software includes all of the necessary IPs required to build such application prototypes.

Note: These designs also include a user-friendly GUI application to test specific features functions. These demo designs include the Flash*Freeze feature for demonstration. Users can monitor the power consumption when the FPGA is in operational and Flash*Freeze modes to evaluate the low-power advantage.

The kit includes Arduino and Mikrobus connectors for flexibility for prototyping, and expansion kits. This can be very useful as the user's FPGA knowledge and the associated tools experience increases. The kit can work as a standalone unit or as an extension to existing MCHP kits.

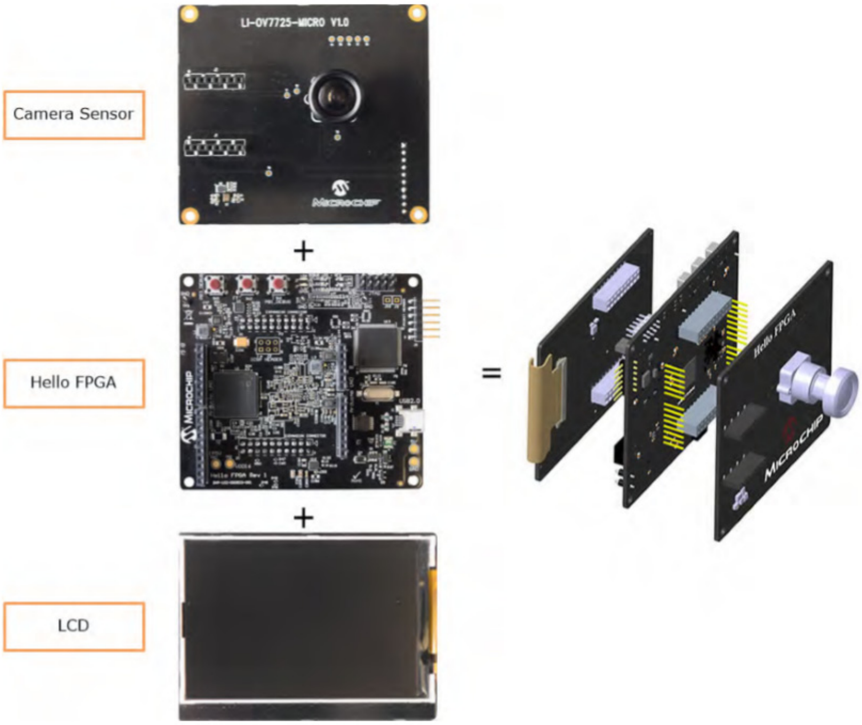
Figure 1, page 3 shows the Hello FPGA kit, which consists of the SmartFusion2 FPGA and MCU (PIC32MX795F512L).

Figure 1 • **Hello FPGA Kit**



The Hello FPGA kit includes the LCD board and the camera sensor. These boards can be connected to the Hello FPGA kit as shown in Figure 2, page 3.

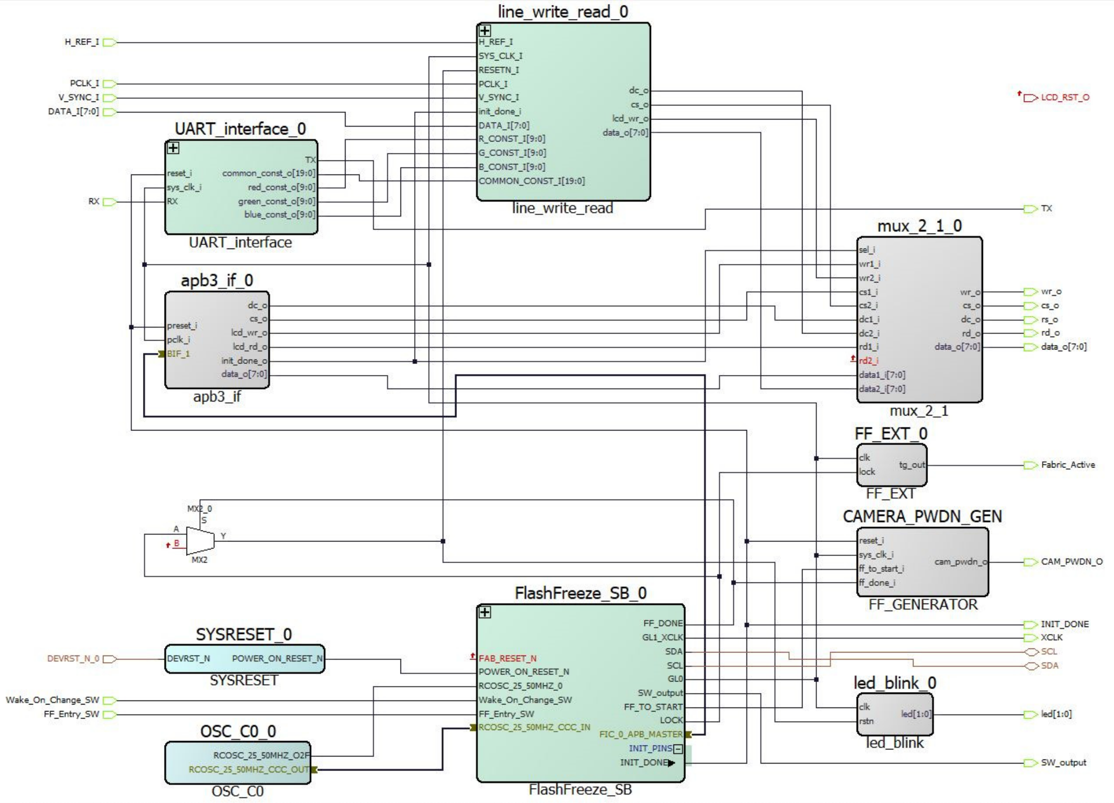
Figure 2 • **LCD and Camera Sensor Boards**



3 Video Demo Design

The following figure shows the top-level video demo design in Libero.

Figure 3 • **Top-Level Hardware Implementation (Video Demo)**



The video demo design interfaces the Camera Sensor OV7725 and LCD with the SmartFusion2 SoC FPGA. The design also includes the Flash*Freeze feature.

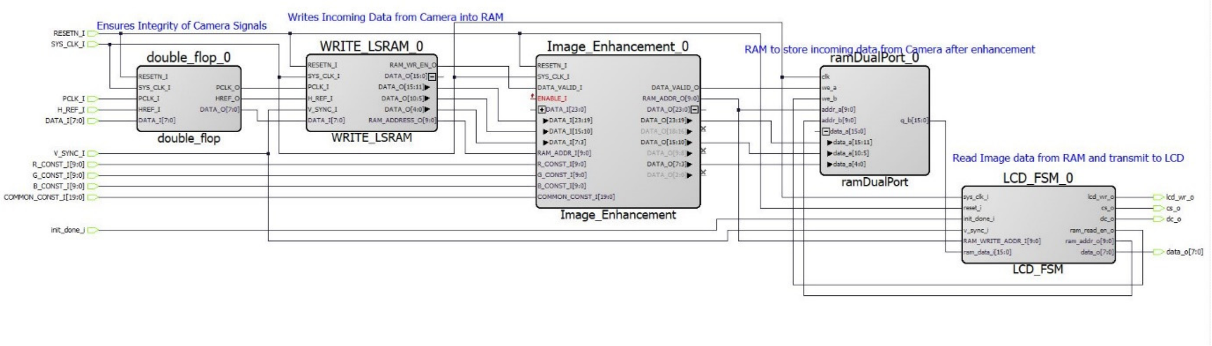
The Top module contains the following modules:

- Line_write_read Module, page 4
- FlashFreeze_SB Module, page 6
- UART_interface, page 9
- APB3_if (APB slave), page 10
- FF_EXT, page 10
- FF_GENERATOR, page 11
- Mux_2_1, page 10
- OSC_C0 IP, page 11

3.1 Line_write_read Module

Figure 4, page 5 shows the components of the Line_read_write module. Line_write_read module acquires the data from the OV7725 Camera sensor, performs image scaling as well as image enhancement and sends data to LCD Screen. OV7725 camera has an array size of 640 x 480 and is configured to operate in RGB 565 format. LCD ILI9488 has a display resolution of 480 x 320. WRITE_LSRAM module down samples the 640x480 camera data to 480x320 LCD resolution. After down sampling the image data goes through image enhancement that can be configured through UART. Image Enhancement block writes the data in the Dual Port RAM. LCD_FSM reads the data from dual port RAM and sends it to the LCD display as per the LCD interface protocol.

Figure 4 • **Line_write_read SD**



The Line_write_read module contains the following components:

- WRITE_LSRAM_0, page 5
- Image Enhancement, page 5
- LCD_FSM, page 5

3.1.1 WRITE_LSRAM_0

Signals from the Camera PLK_I, H_REF, and DATA_I(8) are transferred through the Double Flip-Flop synchronizer circuit to ensure the integrity of these signals. Input data signal from the camera DATA_I is considered valid when the H_REF signal is high. Data from the camera is in the form of RGB 5:6:5 format and is split across 2 clock cycles. Down sampling from 640 pixels per horizontal line to 480 pixels per horizontal line is achieved by skipping a pixel for every 4 pixels. Down sampling from 480 lines to 320 lines is achieved by skipping one horizontal line for every 3 lines.

3.1.2 Image Enhancement

This module performs modification on the pixel output data from the WRITE_LSRAM based on the user-controlled data provided by the UART block. User-controlled data include contrast, brightness, and RGB color adjustment values.

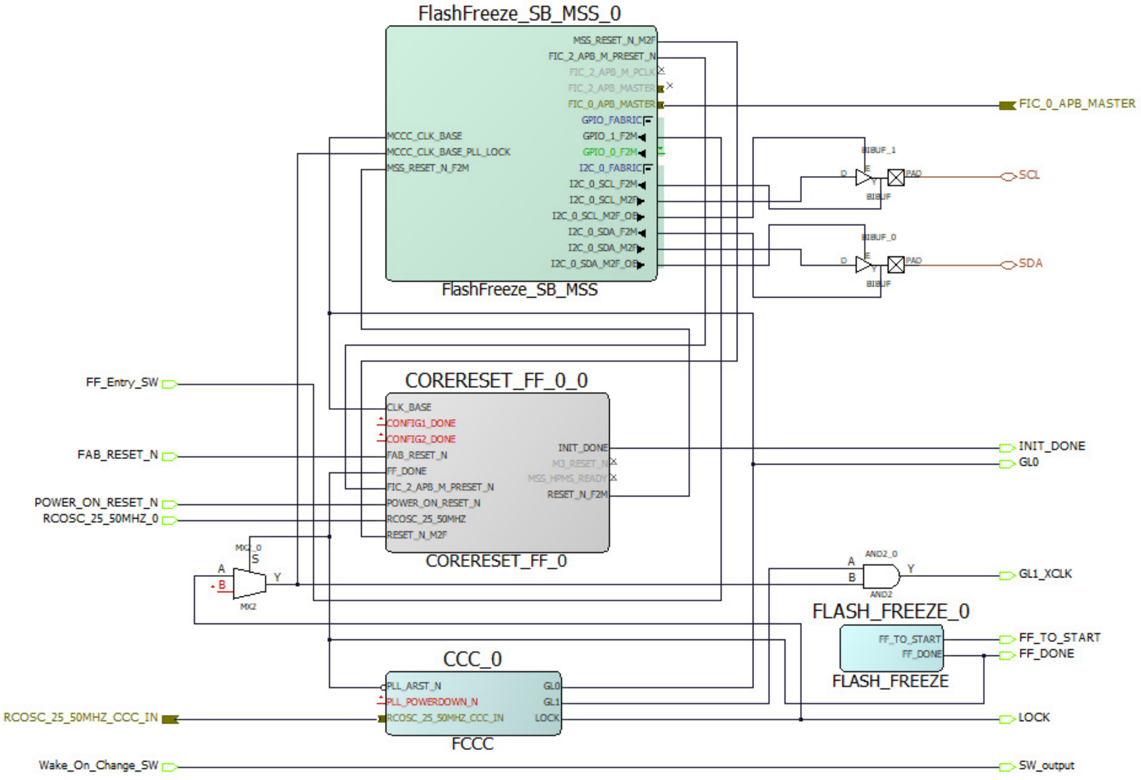
3.1.3 LCD_FSM

The LCD_FSM module reads data from the dual port RAM and sends it to the LCD display. The module starts reading the dual port RAM when the write address in WRITE_LSRAM reaches 160. The LCD_FSM also resets the registers of the LCD based on the V_Sync signal of the camera which indicates the start of new frame data. LCD_FSM provides data to the LCD based on the interface protocol.

3.2 FlashFreeze_SB Module

Figure 5, page 6 shows the components of the FlashFreeze_SB_MSS module. The FlashFreeze_SB_MSS module configures the Microcontroller Sub System (MSS). The I2C peripheral in MSS is used to configure camera registers. MSS is the APB master which is connected to the APB slave (apb3_if) module to initialize LCD registers during startup. The Flash_FREEZE module provides the feature of low power FF mode to the FPGA. The FF_Entry_SW signal provides the hardware interrupt for FF to MSS. When interrupt comes MSS executes the instruction related to FF firmware through which FPGA goes into the flash freeze mode. MSS is in an infinite loop when the FPGA fabric is in the Flash*Freeze mode.

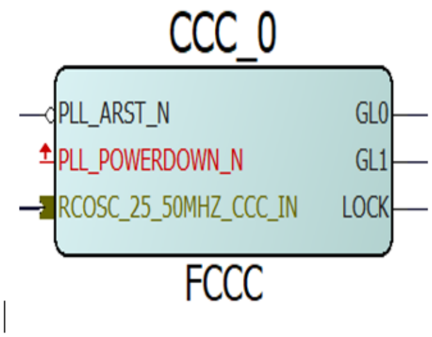
Figure 5 • **FlashFreeze_SB_0 Module**

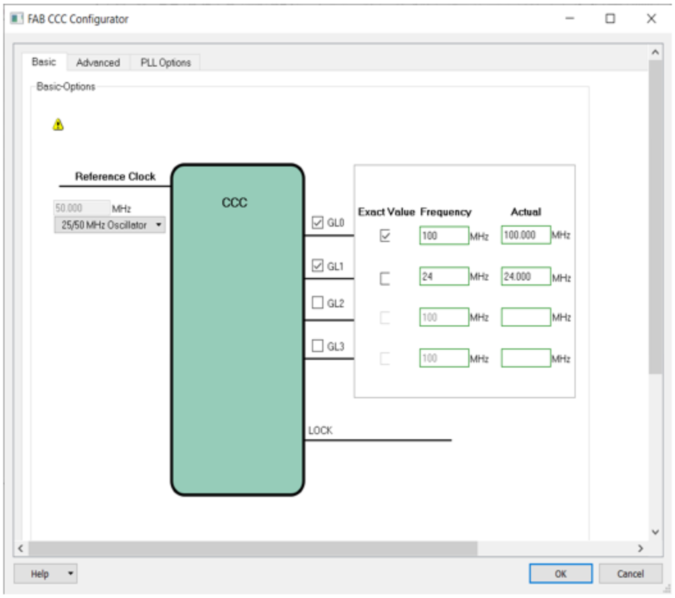


3.2.1 CCC IP

CCC IP is available in Libero SoC -> Clock and Management IP catalog. CCC IP acquires the 25/50 MHZ clock from the on-chip oscillator IP and generates the GL0 clock of 100 MHz and GL1 of 24 MHz clock. All the modules inside FPGA works on GL0 clock. The following figure shows the configuration of the CCC IP.

Figure 6 • **CCC IP Configuration**





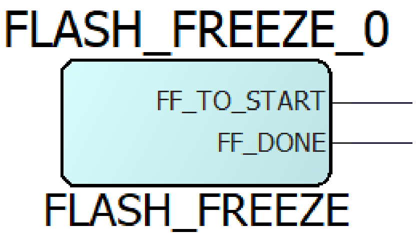
Reference Clock: 50.000 MHz, 25/50 MHz Oscillator

	Exact Value	Frequency	Actual
☑ GL0	☑	100 MHz	100.000 MHz
☑ GL1	□	24 MHz	24.000 MHz
□ GL2	□	100 MHz	MHz
□ GL3	□	100 MHz	MHz

3.2.2 FLASH_FREEZE IP

FLASH FREEZE IP is available in Libero SoC -> Catalog. SmartFusion2 SoC FPGA devices provide an ultra-low static power solution through Flash*Freeze technology. Flash*Freeze mode entry retains all the Static Random Access Memory (SRAM) and registers information. Flash*Freeze mode exit achieves rapid recovery to active mode (approximately 13 μs). The following figure shows the FLASH FREEZE IP.

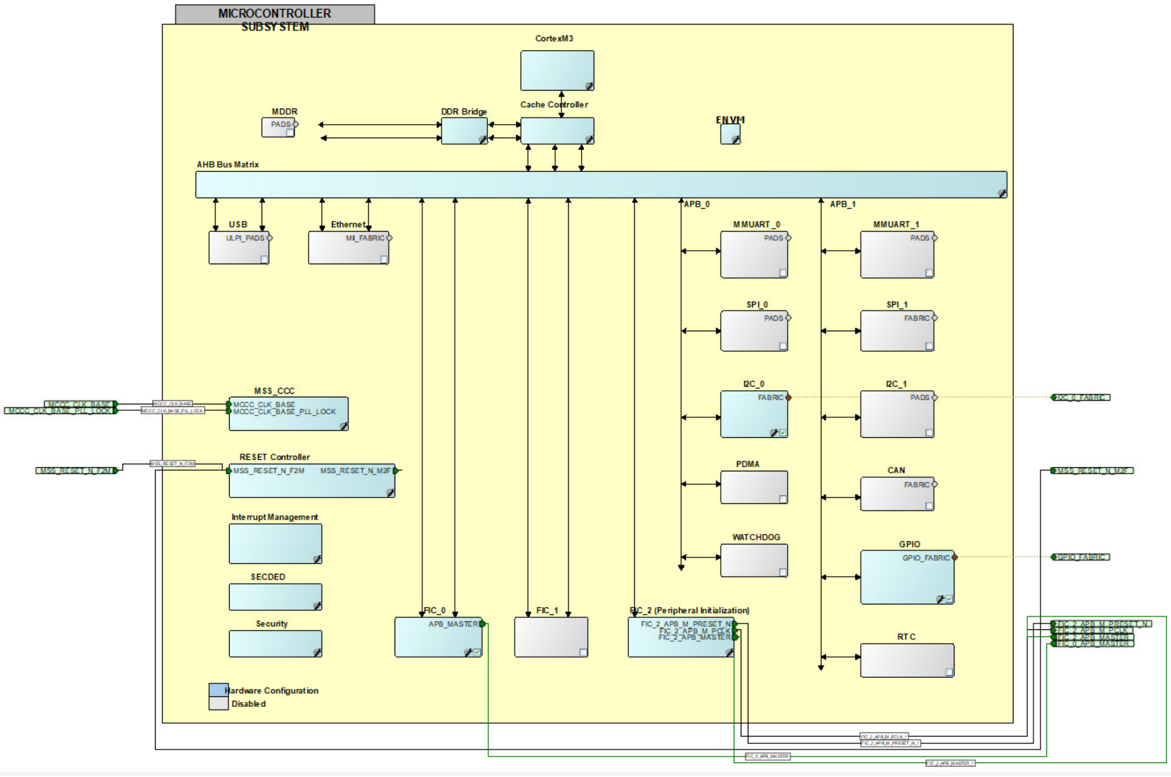
Figure 7 • **Flash_Freeze IP**



3.2.3 MSS IP

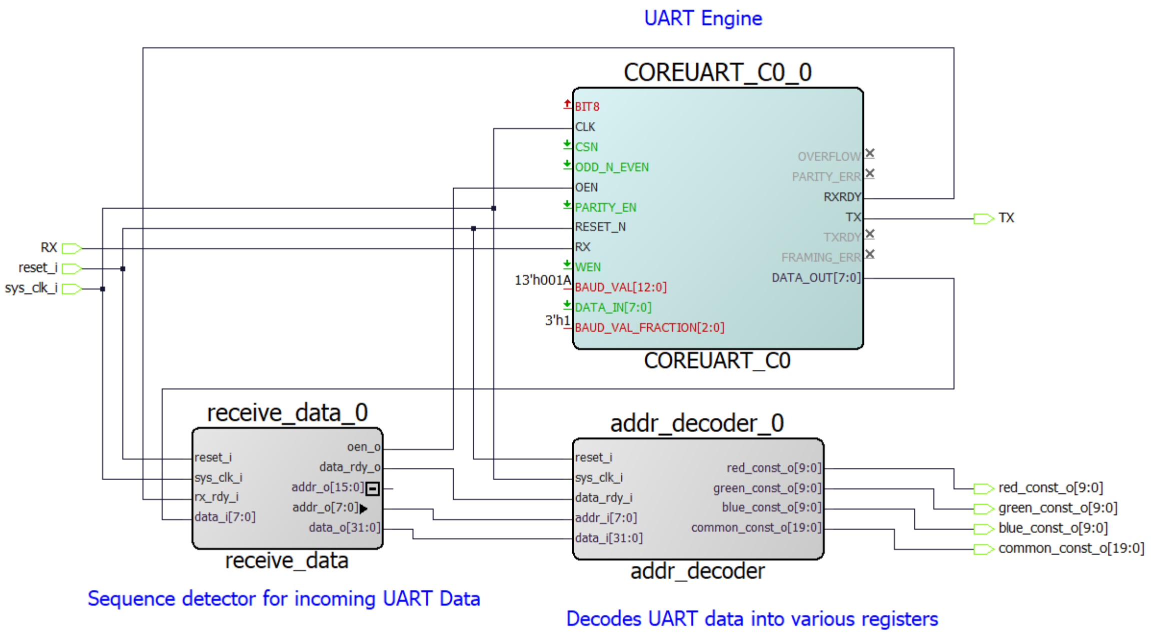
SmartFusion2 Microcontroller Subsystem (MSS) IP is available in Libero SoC -> Catalog -> Processors. The MSS Component Configurator represents a graphical block diagram of the SmartFusion2 Microcontroller Subsystem. Each of MSS sub-blocks can be enabled or disabled as per the application requirements. This design uses FIC_0 for APB interface and I2C_0 to enable camera configuration interface and GPIO to acquire external signal.

Figure 8 • **MSS Component Configuration**



3.3 UART_interface

Figure 9 • **UART_interface SD**



UART Engine

Sequence detector for incoming UART Data

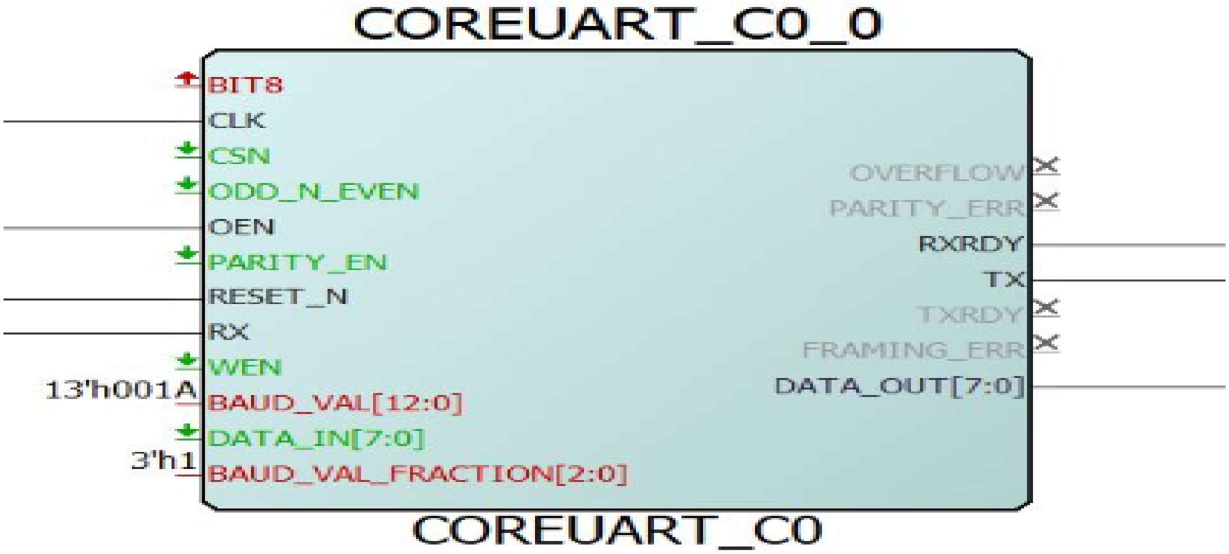
Decodes UART data into various registers

UART_interface SmartDesign performs the task of communication between the SmartFusion2 FPGA and the PC with PIC microcontroller as a bridge. Based on the data from Hello_FPGA GUI, receive_data module generates the address and the data. Addr_decoder module generates the values of brightness, contrast, and color and provides it to the Image_enchancement module.

3.3.1 COREUART_C0

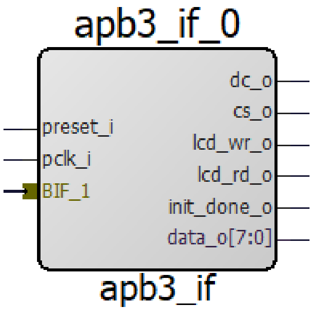
COREUART IP is available in the Libero SoC peripheral IP catalog. BAUD rate of UART is 230400, based on the operating clock frequency, BAUD_VAL, and BAUD_VAL_FRACTION values. The following figure shows the CoreUART configuration.

Figure 10 • **CoreUART C0 Configuration**



3.4 APB3_if (APB slave)

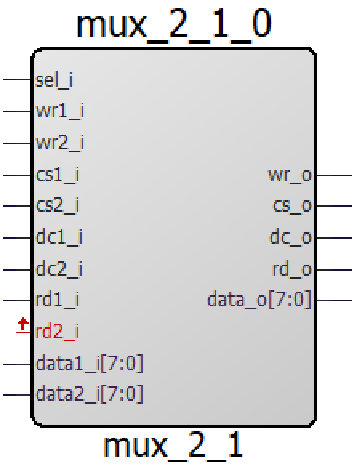
Figure 11 • **Apb3_if**



Apb3_if module implements the APB slave Interface to communicate with APB3 Master (MSS). LCD initialization is done through this module based on the control and data words provided by the MSS. After LCD initialization, MSS gives the command to the APB slave to generate init_done signal.

3.5 Mux_2_1

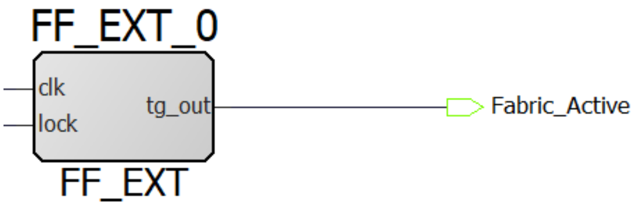
Figure 12 • **MUX_2_1**



MSS provides the select signal (Init_done) to the MUX, MSS first initializes the LCD and after initialization, MSS gives LCD control to the FPGA fabric.

3.6 FF_EXT

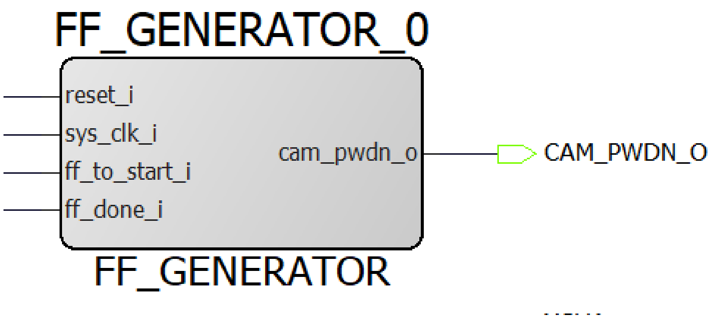
Figure 13 • **FF_EXT Module**



FF_EXT module generates the pulse based on the PPL lock signal. The output pulse is given to the PIC controller, which signifies the FPGA fabric is out of flash freeze mode.

3.7 FF_GENERATOR

Figure 14 • **FF_Generator Module**



FF_Generator module implements a power down signal for the camera module based on the flash freeze signal FF_to_start_i, which indicates the fabric is in flash freeze or active.

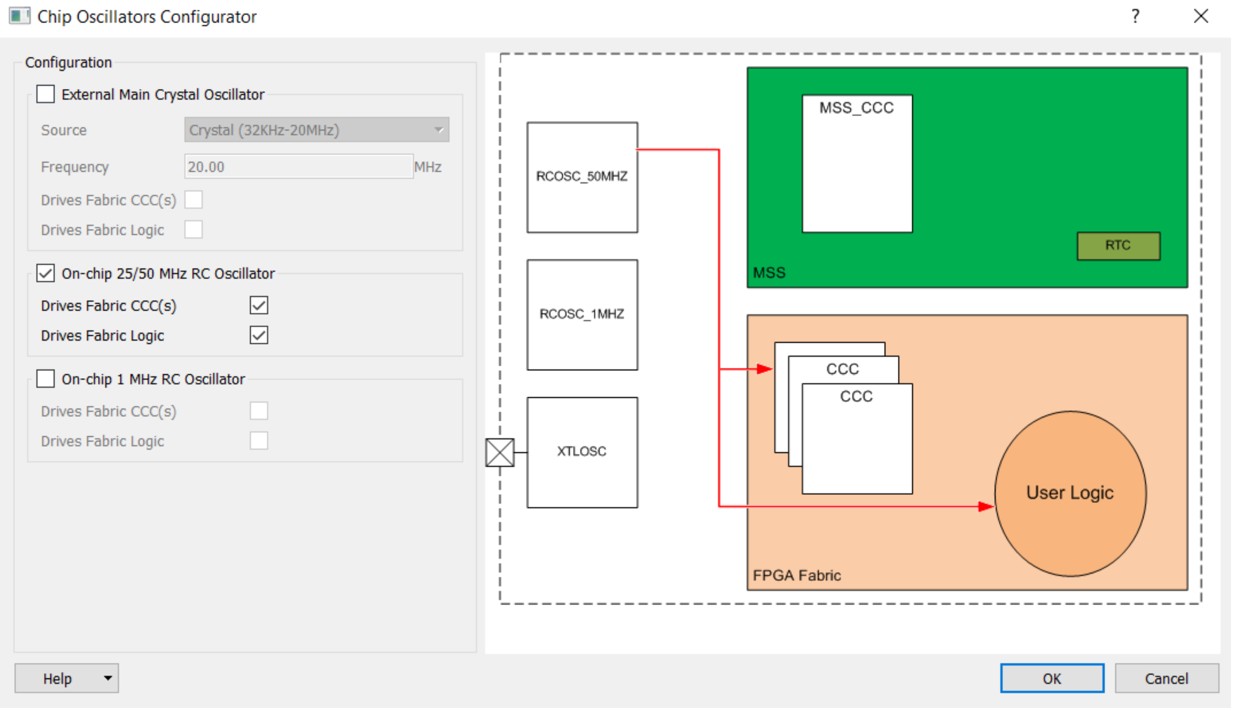
3.8 OSC_C0 IP

OSC_C0 IP is available in the Libero SoC clock and management IP catalog. There are three oscillator blocks in the SmartFusion2 device that can be used in different use models:

- On-chip 25/50 MHz RC Oscillator - This oscillator generates a 50 MHz waveform when the core supply voltage is 1.2V and 25 MHz when the supply voltage is 1.0V. The device core voltage can be selected from the Libero SoC Device Settings dialog box.
- On-chip 1 MHz RC Oscillator
- Main Crystal Oscillator

For this design we used on chip 25/50 MHz RC Oscillator. The following figure shows the configuration of the OSC_C0 IP core.

Figure 15 • **OSC_C0 IP Core Configuration**



Chip Oscillators Configurator

Configuration

☐ External Main Crystal Oscillator

Source: Crystal (32KHz-20MHz)

Frequency: 20.00 MHz

Drives Fabric CCC(s) ☐

Drives Fabric Logic ☐

☑ On-chip 25/50 MHz RC Oscillator

Drives Fabric CCC(s) ☑

Drives Fabric Logic ☑

☐ On-chip 1 MHz RC Oscillator

Drives Fabric CCC(s) ☐

Drives Fabric Logic ☐

3.9 Resource Utilization

The following table lists the resource utilization of the Video demo design.

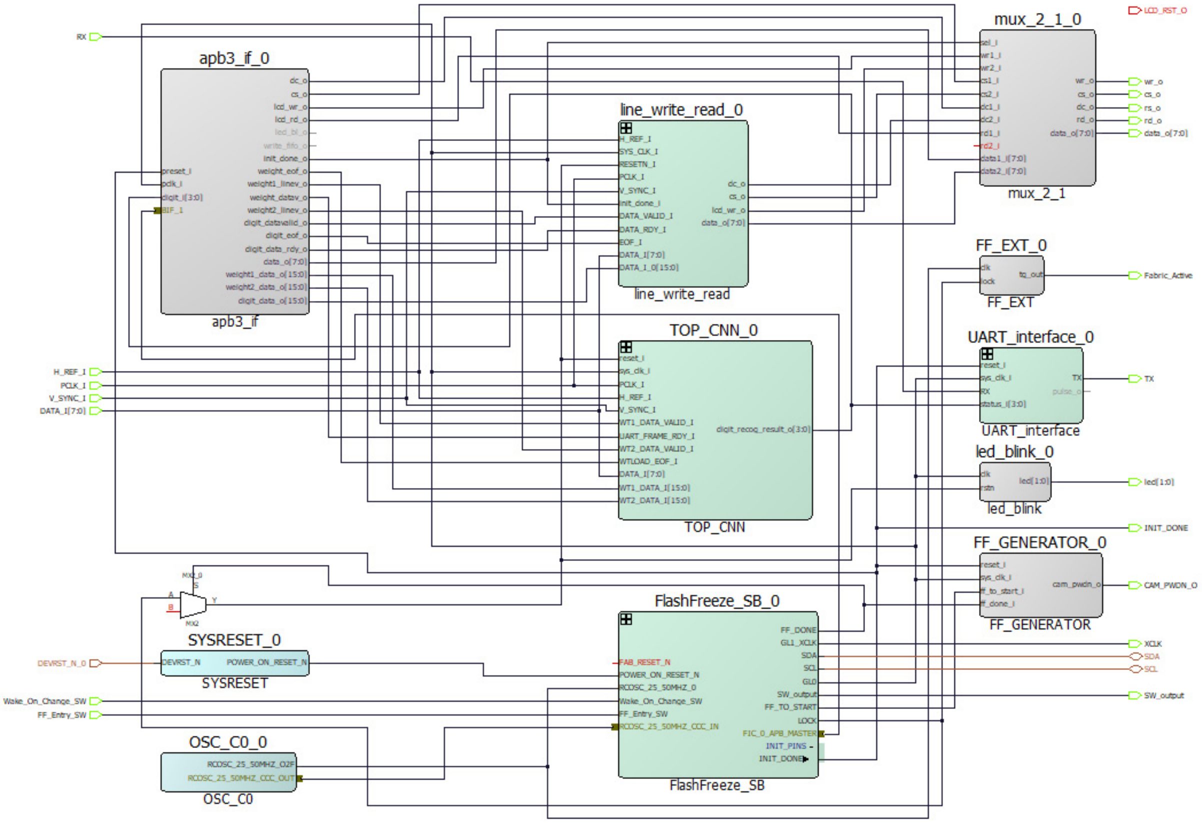
Table 1 • **Resource Utilization**

Type	Used	Total	Percentage
4LUT	558	12084	4.62
DFF	584	12084	4.83
User I/O (single-ended)	37	138	26.81
RAM1K18	1	21	4.76
MACC	3	22	13.64

4 Digit Recognition (AI) Demo Design

The following figure shows the top-level AI digit recognition demo design in Libero.

Figure 16 • **Top-Level Hardware Implementation (AI Digit Recognition)**

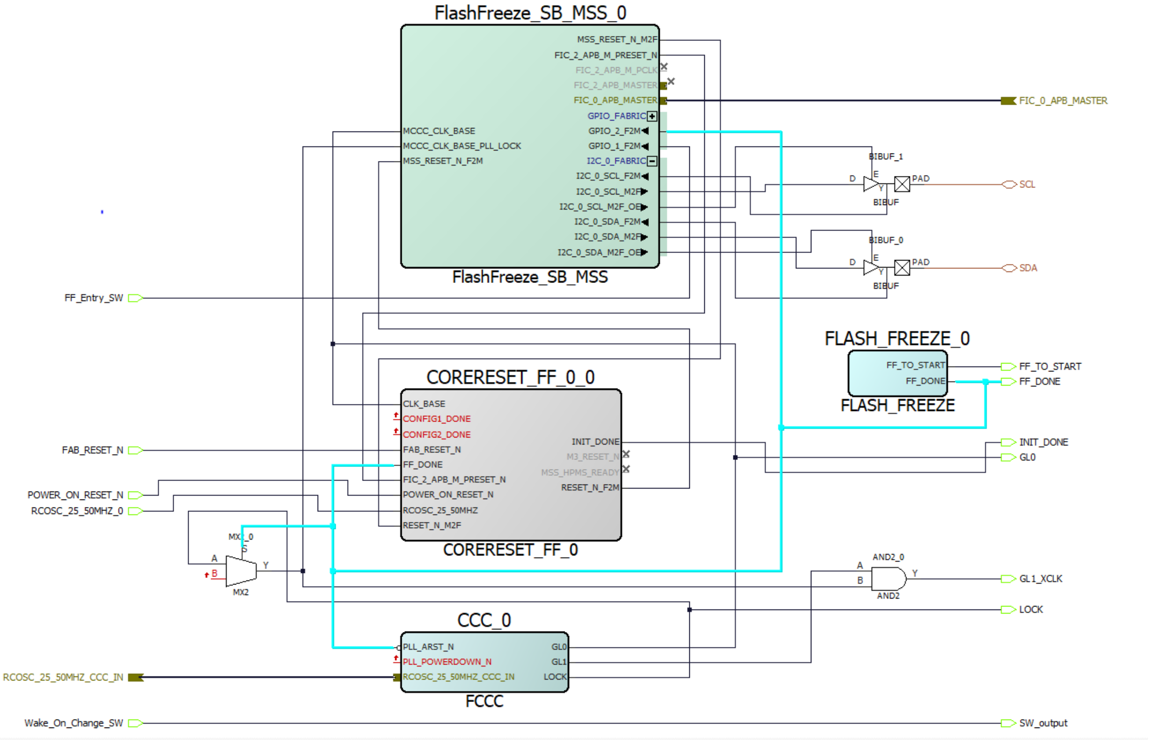


AI Digit Recognition libero project aims to show Artificial Intelligence (AI) application. The Libero design for AI digit recognition is similar to the Video Demo Libero project with below additional modules.

- FlashFreeze SmartDesign, page 14
- Apb3_if (APB slave), page 15
- Line_write_read SmartDesign, page 16
- TOP CNN, page 16

4.1 FlashFreeze SmartDesign

Figure 17 • **FlashFreeze_SB**

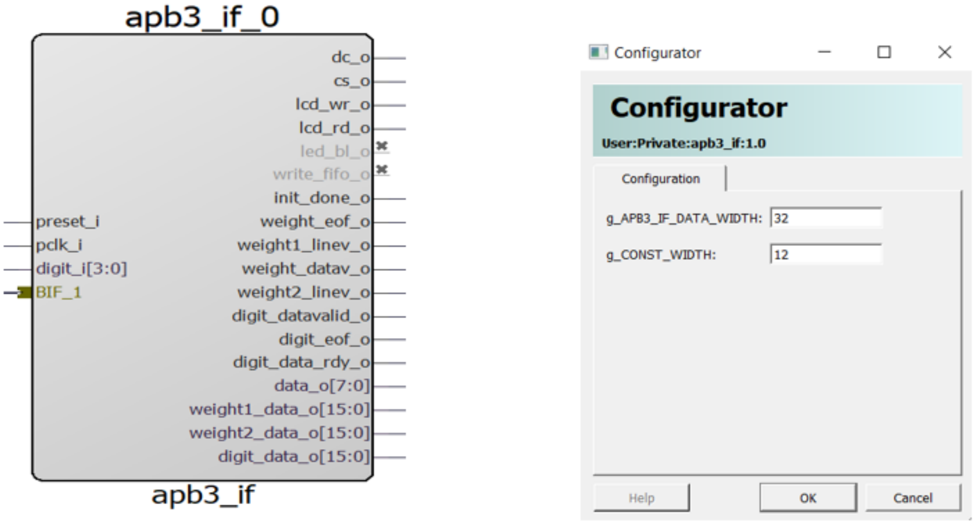


FlashFreeze_SB_MSS configures the MSS. The I2C peripheral in MSS is used to configure camera registers. MSS is the APB master which is connected to the APB slave (apb3_if) module to initialize LCD registers during startup. Flash_FREEZE module provides the feature of low power FF mode to the FPGA. The FF_Entry_SW signal provides the hardware interrupt for FF to MSS. When there is an interrupt, MSS executes the instruction related to FF firmware through which FPGA goes into the flash freeze mode. While FPGA is in FF mode, MSS will wait for the FPGA to come out of FF. MSS identifies that the FPGA is out of FF mode based on the FF_done signal interrupt. After receiving the FF_done signal interrupt the MSS can communicate with the FPGA through APB.

4.2 Apb3_if (APB slave)

Apb3_if is an APB slave module that communicates with the MSS. MSS first loads the weights into the Convolution Neural Network (CNN) SD through APB slave. After loading weights, MSS initializes the LCD display and transfers the LCD control to the FPGA fabric through init_done signal. APB slave acquires the digit recognized result from the CNN SD. The MSS loads the graphics related to the recognized digit in to the LSRAM for On Screen Display (OSD) of the recognized digit. The following figure shows the APB slave module configuration.

Figure 18 • **APB Slave Module Configuration**



Configurator

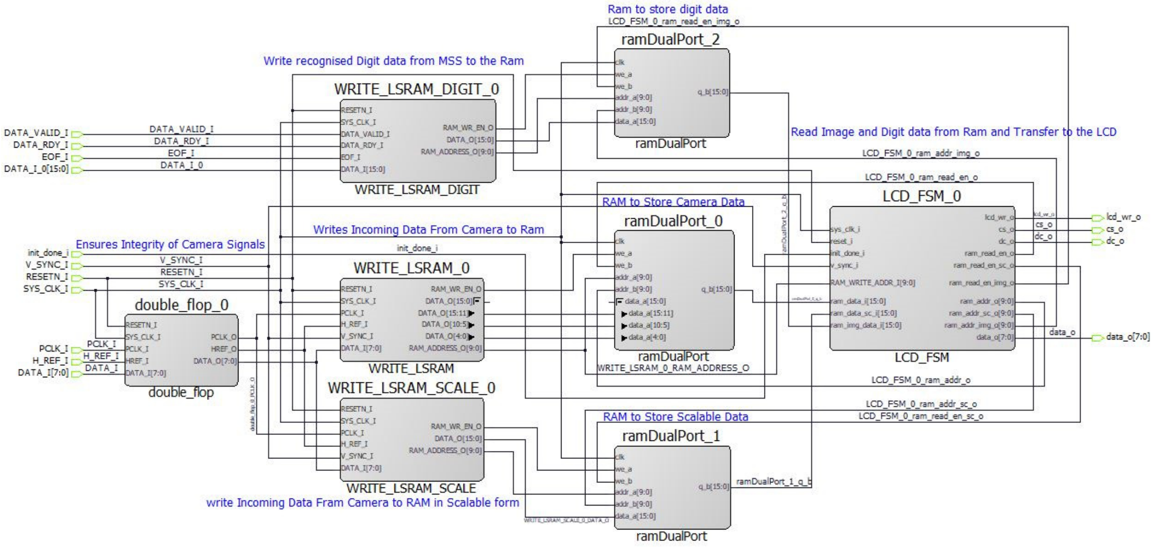
User:Private:apb3_if:1.0

g_APB3_IF_DATA_WIDTH: 32

g_CONST_WIDTH: 12

4.3 Line_write_read SmartDesign

Figure 19 • **Line_write_read SD**



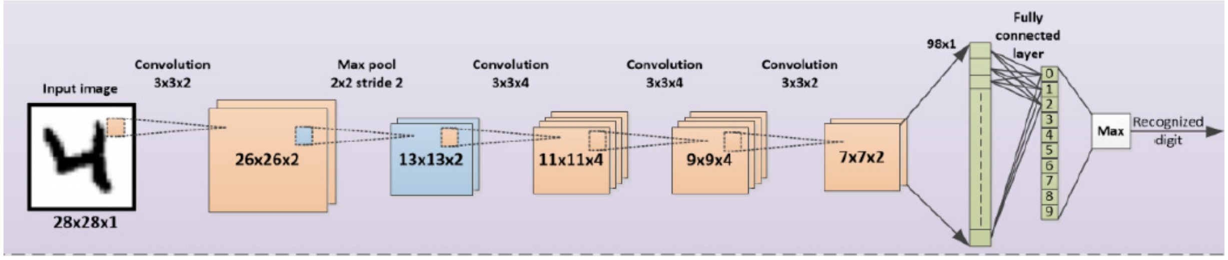
Line write read SmartDesign displays three images on the LCD:

1. Image from camera.
2. Image from camera in the scalable of size 28x28 at top left corner of LCD.
3. Recognized digit of size 80x80 at left bottom corner.

The FPGA receives video from the camera with a resolution of 640x480 at 30fps. WRITE_LSRAM perform scaling of the input video to the 480x320 video format of LCD and write scalable data on the ramDualPort_0. Similarly, WRITE_LSRAM_SCALE performs scaling of video from the camera to the 28x28, which is the resolution required by CNN. The actual scaled image passed to the CNN is displayed through the 28x28 scaled image. Recognized digit OSD data is provided by the MSS through the APB slave of size 80x80, WRITE_LSRAM_DIGIT module saves digit OSD data into the ramDualPort_2. Init_done signal from the MSS indicates the LCD is initialized and MSS gives control of LCD to the LCD_FSM module. LCD_FSM module reads data from all these RAMs and displays it on the LCD according to the interface protocol.

4.4 TOP CNN

Figure 20 • **Implemented SmartDesign block diagram**



The above figure shows the block diagram of the implemented SmartDesign. The network implemented in the demo contains 4 convolution layers, a max pool layer, and a fully connected layer. The network is trained from the standard MNIST handwritten digit database which contains the digits in 28x28 resolution images. The CNN can detect a single digit in the 28x28 image when the aspect ratio of the digit is approximately equal to the aspect ratio of the trained image data set. The network is built for only 10 classes for digits from 0 to 9 and does not use a class that shows no digit. Hence even though the camera is not pointing to a digit, the network will still output a digit that has the maximum value from the fully connected layer.

4.5 Resource Utilization

The following table lists the resource utilization of the digit recognition demo design.

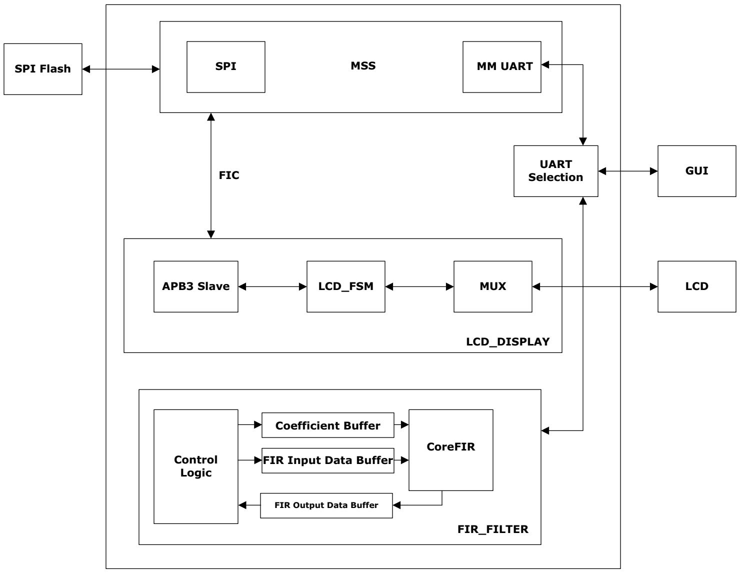
Table 2 • **Resource Utilization**

Type	Used	Total	Percentage
4LUT	7203	12084	59.61
DFF	7168	12084	59.32
User I/O (single-ended)	37	138	26.81
RAM1K18	20	21	95.24
MACC	20	22	90.91
Chip Globals	8	8	100

5 DSP (FIR) Filter Demo Design

In this DSP FIR filter demo design, the FIR filter is implemented in the fabric for Low pass, High pass, Band pass, and Band reject filtering operations. The host interface is implemented in the fabric to communicate with the host PC. A user friendly graphical user interface (GUI) generates the filter coefficients, input signals (Pass-band frequency + Stop-band frequency) and also plots the input/output waveforms and the required spectrum. Microchip CoreFIR filter IP is used to suppress the unwanted frequency components, and generate the output signals to verify the filtering operation.

Figure 21 • **DSP FIR Filter Block Diagram**

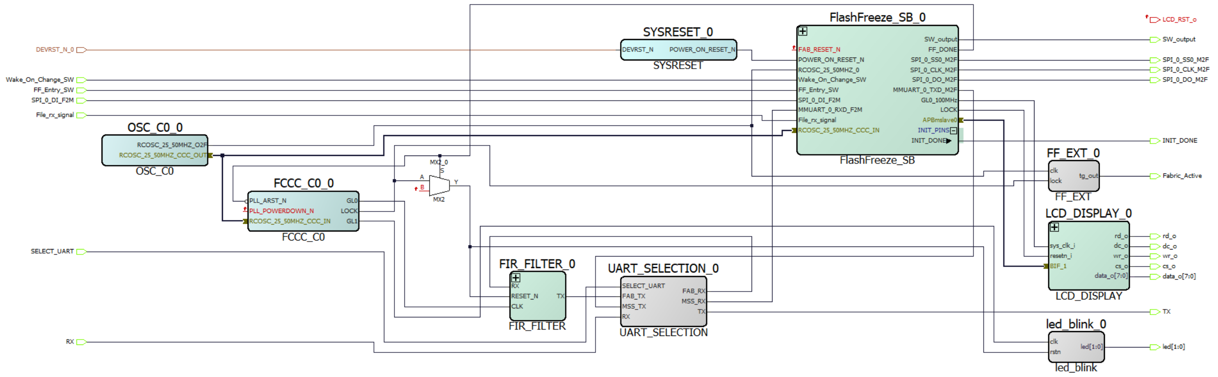


The demo design also implements an LCD display application, which displays images stored in SPI flash on an ILI9488 LCD display. The MSS UART is used to read and write the image data into the SPI flash. For this demo, the SPI flash is preloaded with Microchip logo and a Hello FPGA board image.

Both the applications use UART communication to interface with PIC32 micro-controller which is interfaced with host PC for GUI, due to which a UART_SELECTION module is used to switch between fabric UART and the MSS UART. The Fabric UART is used for DSP FIR application and the MSS UART is used for LCD display application.

Figure 22 shows the top-level DSP FIR_LCD_FF demo design in Libero.

Figure 22 • **Top-Level DSP Design**



The Top module contains the following modules:

- FlashFreeze_SB_0 Module
- FIR_FILTER_0 Module
- LCD_DISPLAY_0 Module

5.1 FlashFreeze_SB_0 Module

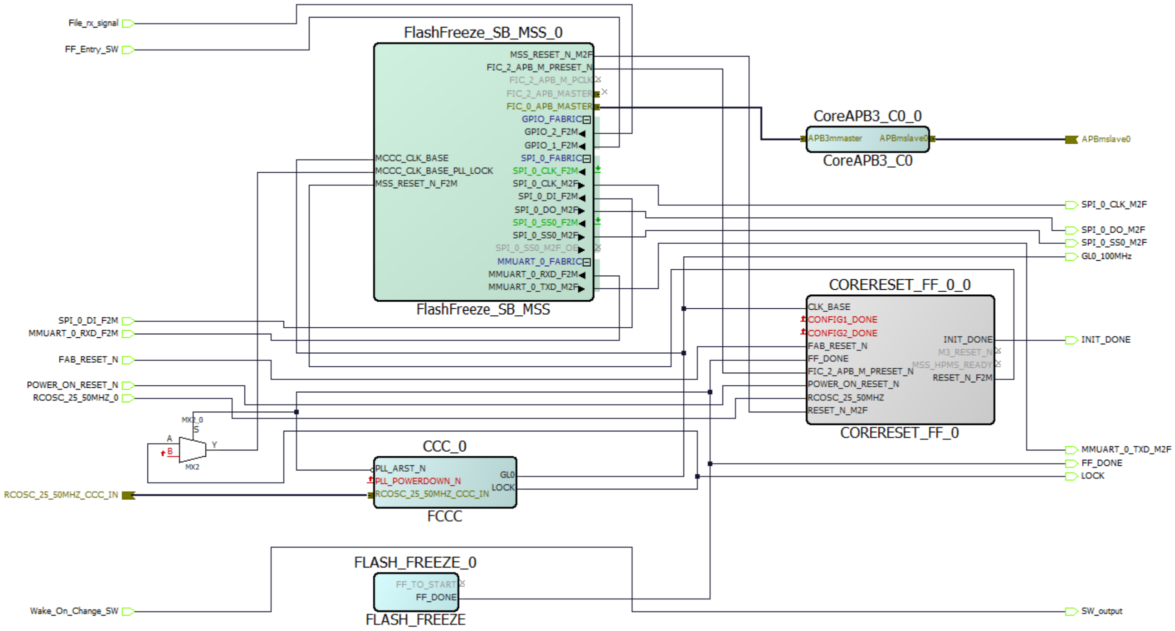
The FlashFreeze_SB_MSS module configures the Microcontroller Subsystem (MSS).

- The SPI peripheral in the MSS is used to store the images in SPI flash and to read the images from SPI flash.
- The UART peripheral in MSS is used to communicate with the GUI on host PC through PIC32 microcontroller.
- MSS is the APB master which is connected to the APB slave (apb3_if) module to initialize LCD registers during startup.
- The Flash_FREEZE module provides the feature of low power FF mode to the FPGA. The FF_Entry_SW signal provides the hardware interrupt for FF to MSS.

When a Flash*Freeze user interrupt is received by the MSS, the instruction related to Flash*Freeze is executed and the FPGA goes into the Flash*Freeze mode. The MSS is in an infinite loop when the FPGA fabric is in the Flash*Freeze mode.

Figure 23 shows the blocks inside the FlashFreeze_SB_0 module.

Figure 23 • **FlashFreeze_SB_0**



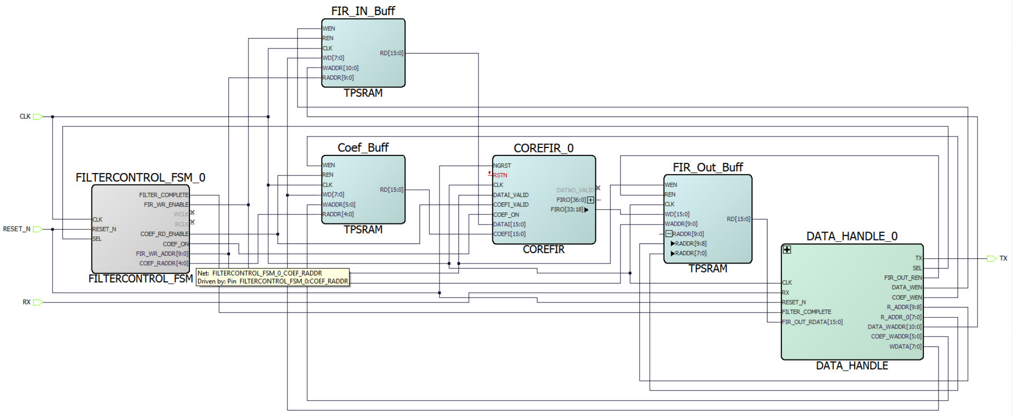
5.2 FIR_FILTER_0 Module

The FIR_FILTER_0 module implements the user logic in the fabric. This module implements the following finite-state machines:

- Data Handling: Implements and controls operations like loading the filter input data to the corresponding input data buffer and loading filter coefficients to the corresponding coefficient memory buffers.
- Filter Control: Controls the FIR filter operation. Loads the filtered data to the corresponding output buffer.
- CoreFIR IP: The Core FIR IP is used in Re-loadable coefficient mode to support Low pass, High pass, Band pass, and Band reject filters.
- TPSRAM IP: The TPSRAM IP is used to implement Filter coefficient buffer, Input signal data buffer, Output signal buffer.

Figure 24 figure shows the blocks inside the FIR_FILTER_0 module.

Figure 24 • **FIR_FILTER Module**



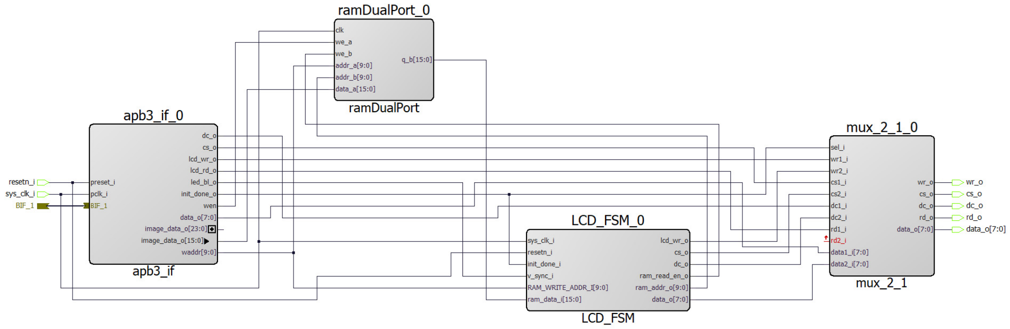
5.3 LCD_DISPLAY_0 Module

The LCD_DISPLAY_0 module implements the following functionalities:

- apb3_if: Apb3_if module implements the APB slave Interface to communicate with APB3 Master (MSS). LCD initialization is done through this module based on the control and data words provided by the MSS. After LCD initialization, MSS gives the command to the APB slave to generate init_done signal. After the init_done signal, the MSS reads the images from SPI flash and stores the data into the dualport ram through abp3_if module.
- LCD_FSM: The LCD_FSM module reads data from the dual port RAM and sends it to the LCD display. The module starts reading the dual port RAM when the write address in WRITE_LSRAM reaches 160. The LCD_FSM also resets the registers of the LCD based on the V_Sync signal of the camera which indicates the start of new frame data. LCD_FSM provides data to the LCD based on the interface protocol.
- mux_2_1: MSS provides the select signal (Init_done) to the MUX, MSS first initializes the LCD and after initialization, MSS gives LCD control to the FPGA fabric.

Figure 25 shows the blocks inside the LCD_DISPLAY_0 module.

Figure 25 • **LCD_DISPLAY_0 module**



5.4 Resource Utilization

The following table lists the resource utilization of the DSP demo design.

Table 3 • **Resource Utilization**

Type	Used	Total	Percentage
4LUT	1430	12084	11.83
DFF	2335	12084	19.32
Logic Element	2540	12084	21.02