SmartFusion cSoC: Mixed Signal Power Manager – Customizing the MPM Reference Design

Table of Contents

Introduction

The SmartFusion® customizable system-on-chip (cSoC) FPGA devices integrate the FPGA technology with hardened ARM® Cortex™-M3 processor-based microcontroller subsystem (MSS) and programmable high-performance analog blocks built on a low power flash semiconductor process. The MSS consists of hardened blocks, such as a 100 MHz ARM Cortex-M3 processor, peripheral DMA (PDMA), embedded nonvolatile memory (eNVM), embedded SRAM (eSRAM), embedded FlashROM (eFROM), external memory controller (EMC), watchdog timer, system registers, Phillips Inter-Integrated circuit (I2C), SPI, 10/100 Ethernet controller, real-time counter (RTC), general purpose input output (GPIO) block, fabric interface controller (FIC), and in-application programming (IAP). The programmable analog block contains the analog compute engine (ACE) and analog front-end (AFE) consisting of ADCs, DACs, active bipolar prescalers (ABPS), comparators, current monitors, and temperature monitor circuitry.

These unique features make the SmartFusion cSoC an ideal choice for Power management solutions.

Power management of boards or of a complete system is one of the main challenges in designing a high availability system. To create a reliable system, you need to use complex devices, often mixing FPGAs, microprocessors, ASICs and ASSPs on the same board. Many of these devices require different power supplies to achieve their full functionality.

Power management mainly deals with:

- The power-up/power-down sequence of different devices on the board or complete system
- Power supplies output voltage monitoring and gives an indicator of a fault
- Maintains constant supply voltages by trimming output of the voltage regulators as per device power requirements

This application note describes a power management solution called Mixed Signal Power Manager (MPM) reference design using the SmartFusion cSoC. The MPM reference design has been enhanced as per the user requirements for adding ACE services to monitor additional power supply rails. The MPM reference design supports both analog point of loads (APOL) and digital point of loads (DPOL).

Power Management Overview

Power Management helps control the power system of a board or a complete system. To create a reliable system, you will need to use complex devices, often mixing FPGAs, microprocessors, ASICs, and ASSPs on the same board. Many of such devices require different power supplies to achieve their full functionality. Often these supplies must be applied in a predetermined and consistent order known as power sequencing. Proper sequencing will also limit any inrush current effects that can put strain on the system power supplies.

Figure 1 on page 4 gives the detailed description about how power management is done in an example system which runs with multiple applications.

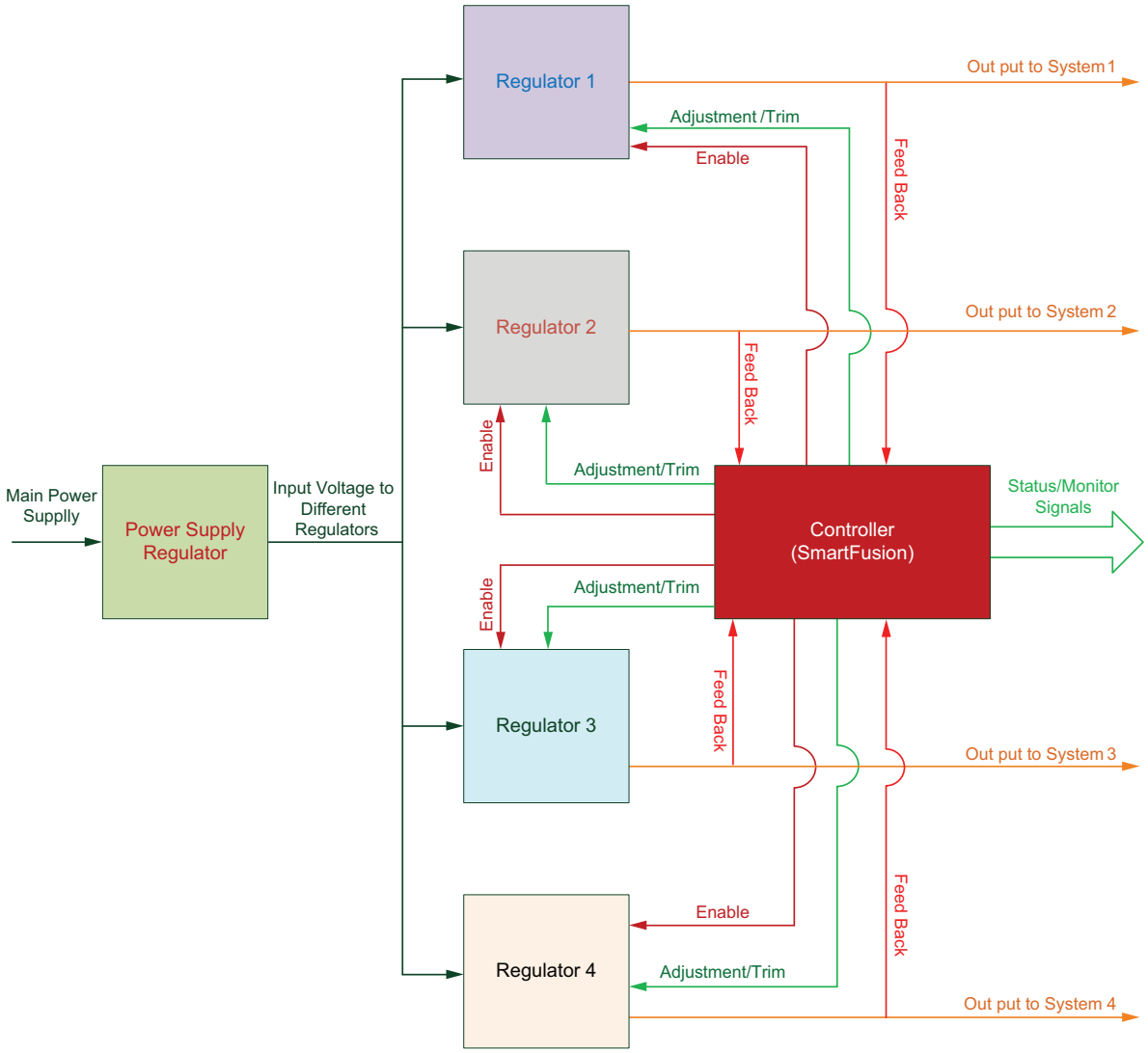


***Figure 1* • Block Diagram of a System Showing Power Management**

Consider an example system that requires different voltages to drive different functional blocks and hence it is not always possible to hook up the main source directly to power functional blocks. Power management here plays a major role in driving all the blocks with the required voltage without any fluctuations. Hence, the voltage of different voltage rails is monitored very closely all the time to avoid any damage to the system due to voltage ramp ups. Voltage regulators do the job of maintaining the voltage levels to the required value without any fluctuations.

Voltage Regulator

The voltage regulator in a typical system is used to accomplish different goals like step-down the voltage among different sub-circuits that require low supply voltage, or to step-up the voltage for sub-circuits that need higher voltage.

As shown in Figure 2, a regulator consists of four signals:

- Input Voltage (V_{in})
- Enable (E_n)
- Output Voltage (V_{out})
- Feedback/Adjust/Trimming signal

The enable and the trimming signals come from the controller/PLD device. The enable signal enables the voltage regulator and trimming performs small adjustments on the output voltage of a regulator or power supply (less than 10% of the output) by driving the trim, adjust, or feedback pin of the regulator to maintain constant voltage.

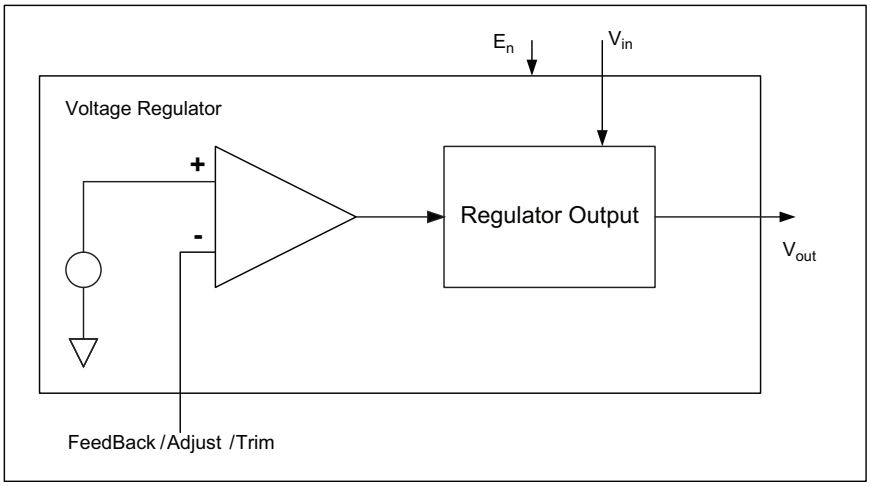


***Figure 2* • Voltage Regulator**

The trimming signal is set in two ways:

- Open loop trimming
- Closed loop trimming

Open Loop Trimming

As mentioned above, the Trim pin is used to adjust the output voltage of the regulator. In open loop trimming there are mainly three ways to set the Trim signal.

By using Discrete Components on the Board

In this method, the output of the regulator is set to particular value and it is not possible to adjust the output unless you change the discrete components.

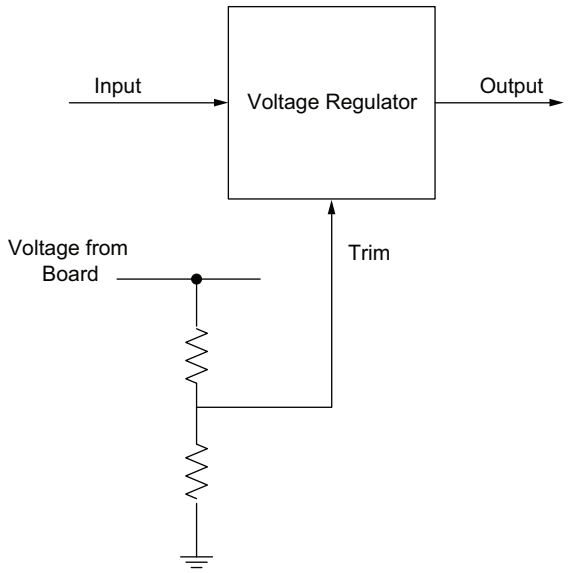


Figure 3 **• Trim Value Set By the Discrete Components on the Board**

By using Potentiometer

In this method the output of the regulator can be adjusted by changing the potentiometer manually.

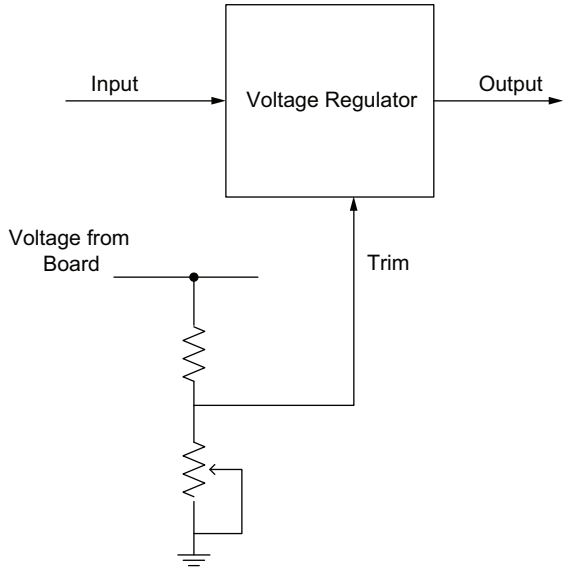


Figure 4 **• Trim Value Set By the Potentiometer**

By Setting the Trim Signal using the Controller at the Initialization Time

As shown in Figure 5, the trim input is set by the controller that puts out a pulse-width modulated (PWM) signal that acts as a DAC when fed through a low pass filter such as an RC network. The feedback pin value is never adjusted and it is set once at the time of system initialization to a fixed value until a reset or power cycle occurs in a system.

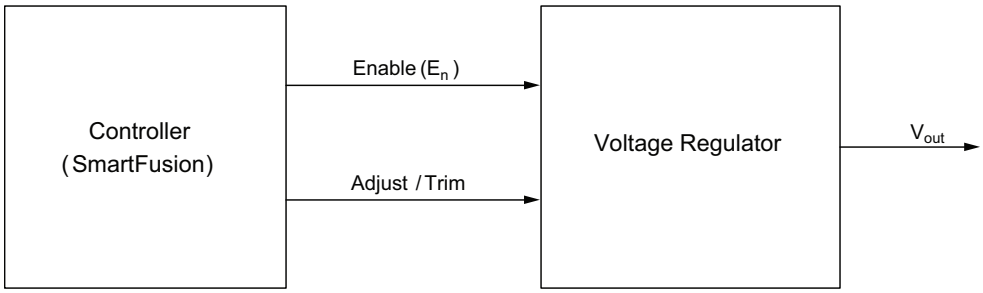


Figure 5 **• Trim Value Set By the Controller**

Closed Loop Trimming

In closed loop trimming, the trim input of the power supply is dependent upon the output voltage of the voltage regulator. In closed-loop trimming, the controller constantly scans (once per loop) the output voltage of the regulator, and actively adjusts the regulator feedback voltage to drive the regulator to some target output voltage.

Closed-loop trimming is Active mode; it is continually operating. The algorithm for trimming is linear. The main function of the algorithm is determining the trimming voltage (V_{trim}) depending upon the output voltage of regulator and target voltage.

Here, trimming voltage (V_{trim}) is the signal which is fed to the regulator feedback pin and target voltage is the voltage level that user wants at the regulator output.

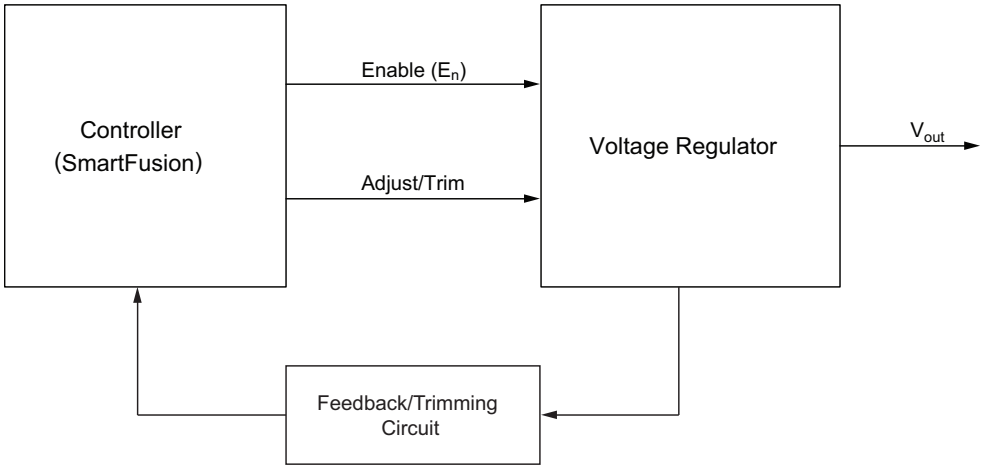


Figure 6 **• Block Diagram of Closed Loop Trimming**

The input V_{in} and the trimming signal should be analog signal. There are several techniques used to generate trimming signals, two of them are:

- Digital to analog converter (DAC)
- Pulse width modulator (PWM)

DAC

The DAC is a device that converts a digital signal to analog signal. Digital-to-analog conversion can degrade a signal, hence, the conversion details are normally chosen in a manner that the errors are negligible. Sigma-Delta DAC is one type of DAC which is driven by pulse density modulated signal, created with a low-pass filter, step, and negative feedback loop. The analog low-pass filter at the output attenuates the noise at higher frequencies.

The 8-bit, 16-bit, or 24-bit unsigned binary digital input word to be converted is fed to an all-digital first-order sigma-delta modulator. Sigma-delta modulation is a subset of the class of duty factor modulation methods, as is PWM. In all of these, the output duty factor approximates the input to the modulator over some time period. Typically, the input signal changes slowly, and the output of the modulator is clocked at a higher sample rate, but with a lower output resolution. In the limiting (and commonly used) case, the output only has one-bit of resolution.

PWM

The PWM is a general purpose, multi-channel module for motor control, tone generation, battery charging, heating elements, fine tuning of power supply output levels, etc. It is the simplest DAC type. A stable current or voltage is switched into an analog low-pass filter with a duration determined by the digital input code.

In General Purpose PWM mode, duty cycle updates can be performed asynchronously or synchronously, based on parameter selection.

The Low Ripple DAC mode creates a minimum period pulse train whose High/Low average is that of the chosen duty cycle. When used with a low-pass filter (such as a simple RC circuit), a DAC can be created with far better bandwidth and ripple performance than what a standard PWM algorithm can achieve. This type of DAC is ideally suited for fine tuning of power supply levels.

The Low Ripple DAC mode is intended to drive a low-pass filter, typically a single-pole RC filter. Narrow pulses of constant width are spread evenly over time such that the average voltage is equal to the duty cycle. The output of the filter is then a DC voltage directly proportional to the duty cycle. This type of pulse train allows for much lower ripple at the output of the filter, and benefits from either higher bandwidth and/or smaller R and C values.

The PWM holds the frequency constant and varies the pulse width (t_{ON}) to adjust the output voltage. The average power delivered is proportional to the duty cycle, D, making this an efficient way to provide power to a load.

$$D = \frac{t_{ON}}{t_{ON} + t_{OFF}}$$

Figure 7 gives the information about how the PWM output is averaged to a varying DC voltage. In the following diagram, assume that the Main Power Supply is of 12 V. At reset, the PWM duty cycle, or level out value, is 100% and the voltage increases to the rail of 12 volts. The PWM duty cycle/level out value changes to 75% and then 50%, and the output of the RC filter follows this by dropping to 8 volts and then 6 volts. The generated ripple voltage is a function of the RC circuit values, the system clock period, and the PWM duty cycle. In Low Ripple DAC mode, the pulse width is effectively reduced to a 1 clock cycle period, significantly reducing the ripple at the output of a low-pass filter. Using Low Ripple DAC mode has the added benefit of requiring a smaller time constant for the filter, which allows for smaller R and C components to be used.

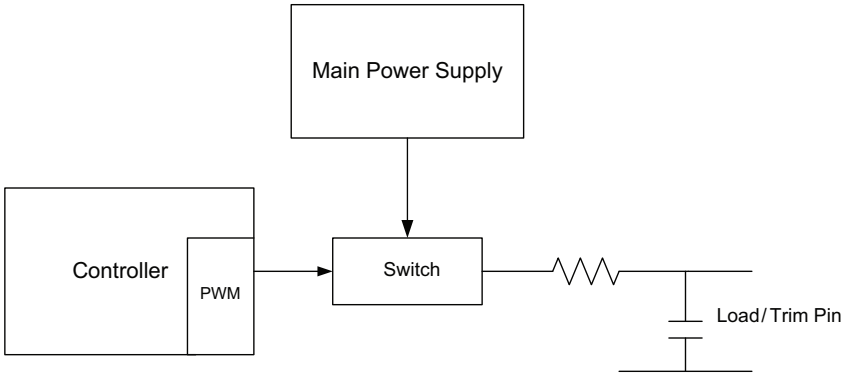


***Figure 7* • Averaging of PWM Output**

The purpose of trimming is to perform small adjustments on the output voltage of a regulator or power supply (less than 10% of the output) by driving the trim, adjust, or feedback pin of the regulator.

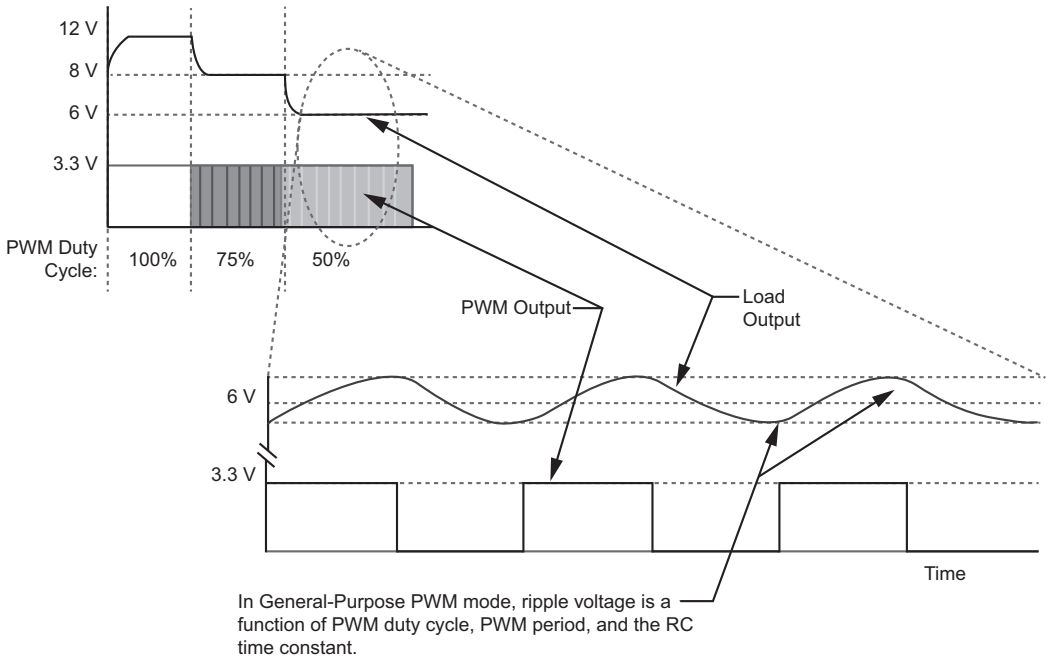


***Figure 8* • Voltage Waveform Of Voltage Across Load**

Trimming is known to provide a voltage regulator with a trim input to which a control signal can be supplied to trim the output voltage of the voltage regulator. Such trimming can be used to adjust the output voltage to a desired value. This compensates for changes of the power supply characteristics due to temperature changes or aging, in which, the output voltage of the power supply is raised or lowered from the set point. This is referred to as an active trim or an active DC output control of the voltage regulator.

The main function of the power management solution (MPM) using the SmartFusion are power sequencing, power monitoring, and power supply margining.

MPM Design Description

Figure 9 shows the block diagram of the MPM reference design using the SmartFusion cSoC.

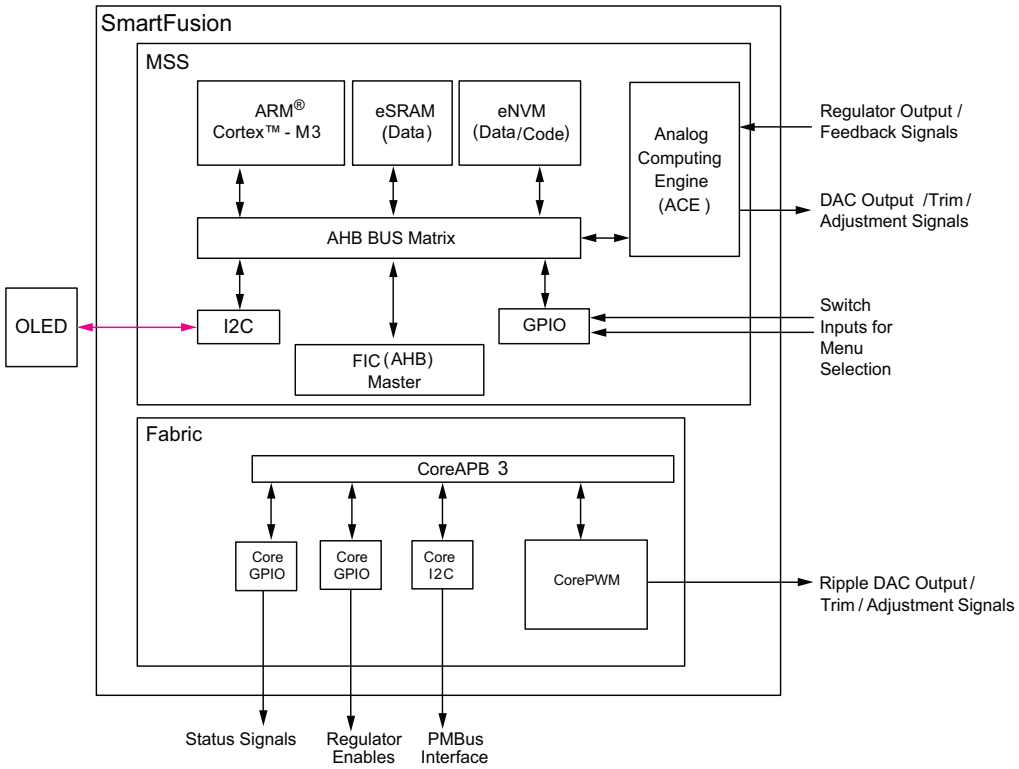


***Figure 9* • Block Diagram of the MPM Reference Design**

The MPM reference design delivers superior power monitoring, power sequencing, closed-loop trimming, and power-up and power-down control of external power supplies (external regulators).

Hardware Implementation Details

The FPGA fabric is used to implement the CorePWM and CoreGPIOs(2). The CorePWM is configured in the low-ripple DAC mode that generates the trim/adjust signals to voltage regulators. CorePWM supports 32 individual outputs. CoreGPIOs are used to implement the status signals like over voltage, under voltage, and enable signals to different voltage regulators.

Figure 10 depicts the flow of the Hardware implementation. The two CoreGPIOs and the CorePWM are connected to the Cortex-M3 processor through the APB Interface. The CorePWM continuously generates the duty cycle and performs trimming. The CoreGPIOs are used for enabling the regulators and the LEDs.

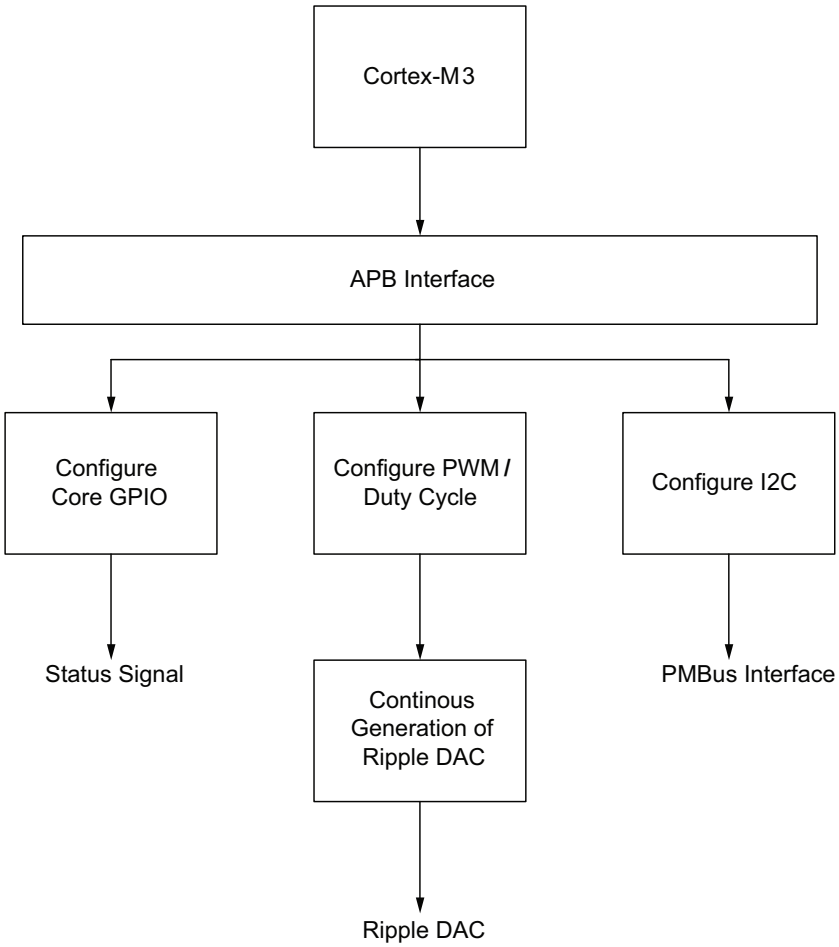


Figure 10 **• Block Diagram Representing Hardware Flow**

Analog Computing Engine (ACE)

The SmartFusion inbuilt ACE is used to generate the trim/adjust signals to different voltage regulators using the DAC. ACE is used to monitor the output of voltage regulators using the ADC.

Software Implementation Details

The software design performs the following operations:

- Initializes and configures the SmartFusion ACE to monitor the voltage levels of MPM channels
- Sets the MPM configuration registers through I2C
- Logs the MPM status of multiple channels to the eNVM
- Controls the CorePWM or on chip DAC to trim output voltage of different regulators
- Controls CoreGPIOs to enable regulators and to generate status signal
- Checks the MPM status and shuts down the system if the either of OFF, OV2 or UV2 is observed
- At regular intervals, each MPM channel information is displayed on OLED

The MPM solution block diagram is shown in Figure 11. As depicted, the Voltage regulator is connected through a three pin jumper. This jumper is used to select the trim type, either SDD or PWM. When the jumper is connected to the ACE pin and the output pin then it will result in SDD trimming type and the other way results in PWM trimming type. In the similar manner the other three regulators are also connected.

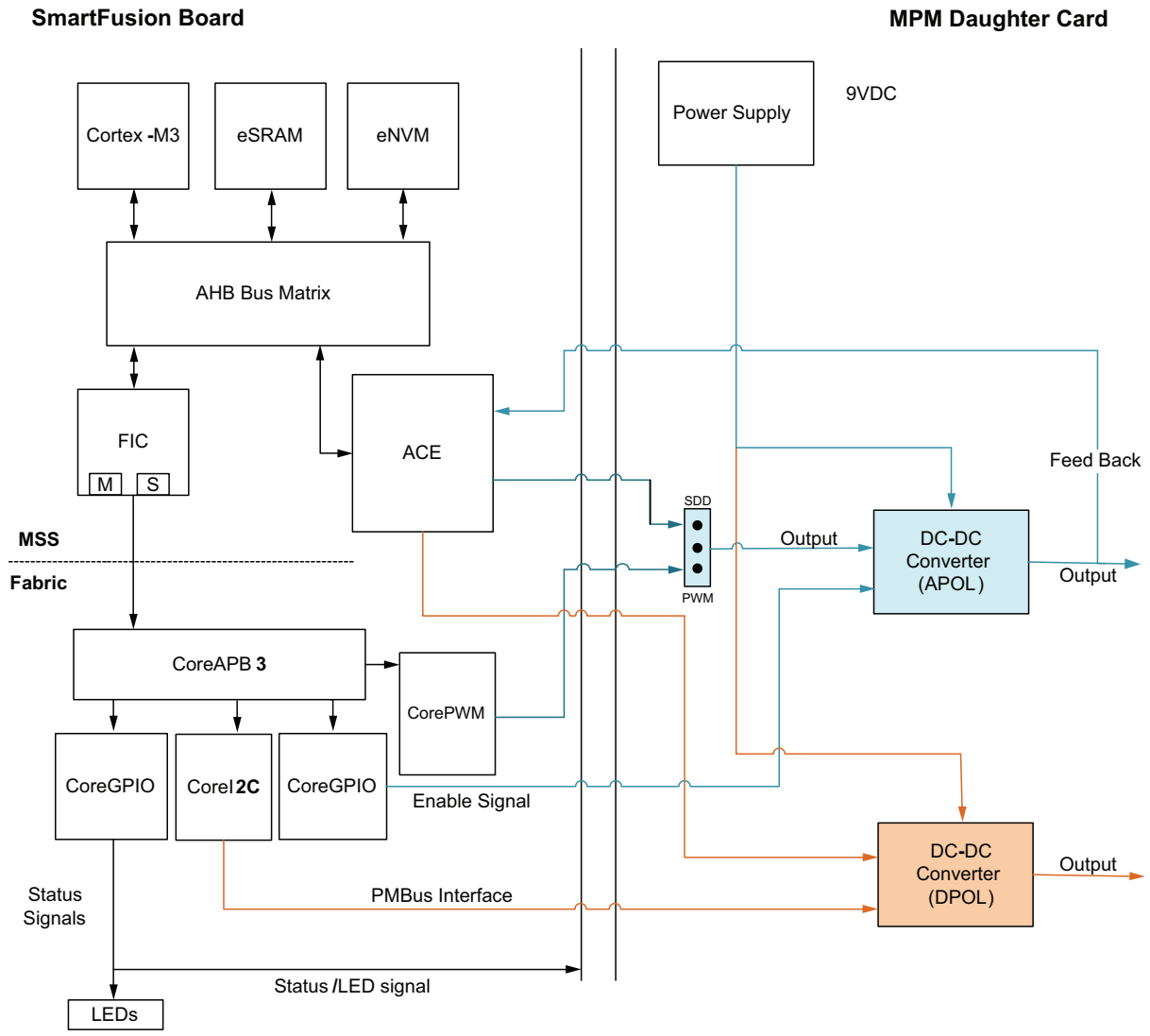


Figure 11 • **Block Diagram Of Complete Set Up Of MPM**

Trimming value is increased if the voltage is more than the nominal and it is decreased if the voltage is less than the nominal. This can be implemented using any of the following:

- Linear function or
- PI controller function or
- Using the equation:

$$Y=mX+c \qquad \textit{EQ 1}$$

where:

'Y' is the output duty cycle,

'X' is the step size that varies with the state of the channel,

'm' is the slope, and

'c' is constant

In the current design, the power ramp has been implemented using the above equation. Figure 12 and Figure 13 on page 12 are the graphs that are obtained with the implementation of above equation.

The name Channel<n>_M<p>_C<q> gives the following details:

'n' represents the channel number,

'p' gives the slope value, and

'q' gives the constant value

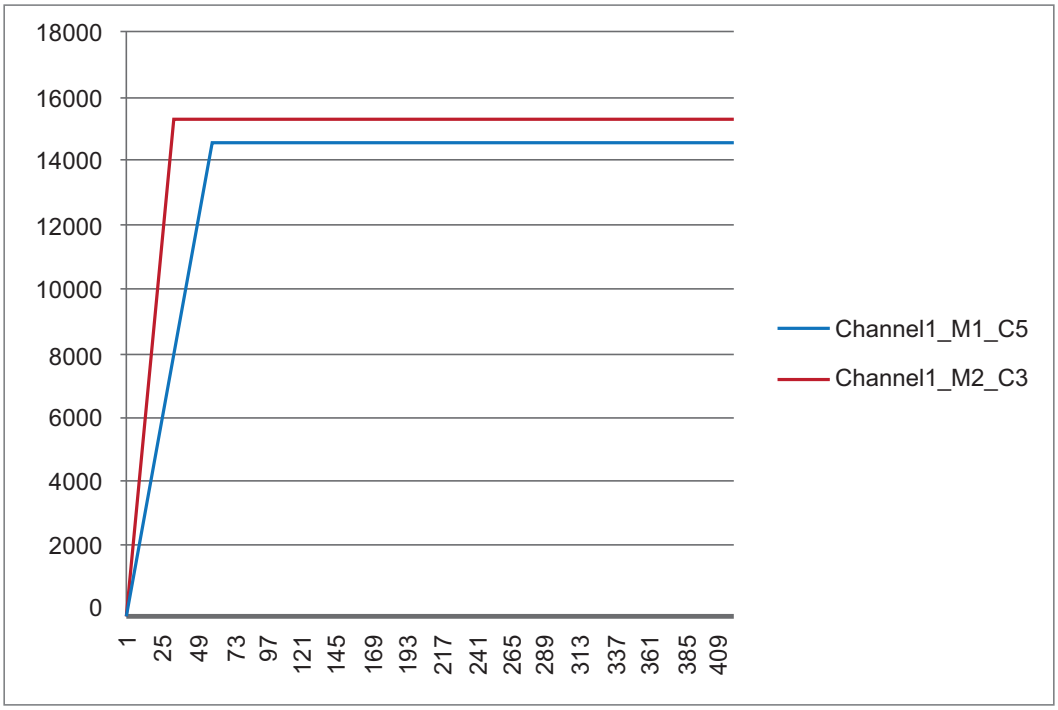


Figure 12 **• Graphical Representation Of Power Ramp Of Channel 1**

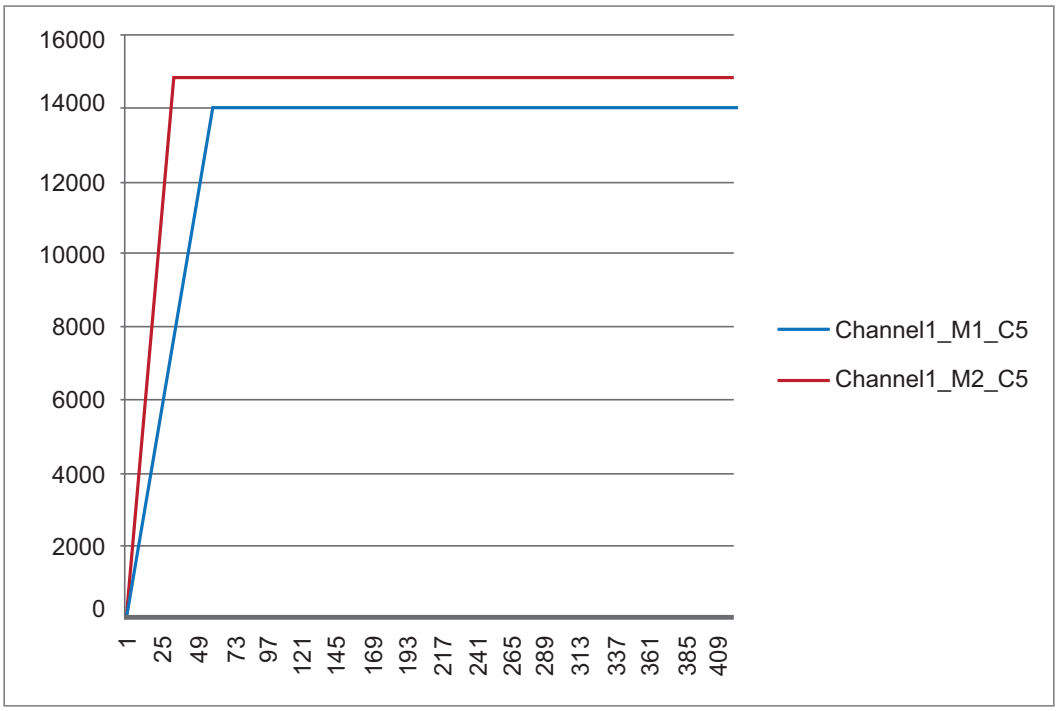


Figure 13 **• Graphical Representation Of Power Ramp Of Channel 2**

The MPM solution has the following components:

- Design files: The Hardware design files contain configuration of hardware components like PWM channels, ACE, GPIOs etc. The Software design files contain C implementation for the initialization of PWM, enumeration of all ACE channels, and power management algorithm, shutting down the system when any fault condition is observed.
- GUI: The GUI enables you to configure power management and drive output signals as the monitored voltages meet or deviate from the user-programmed operating limits. MPM design is programmed into the device through an easy-to-use standalone GUI tool.
- MPM power management demo daughter card (MPM-DC).

The complete flow of the MPM reference design is explained in the Figure 14.

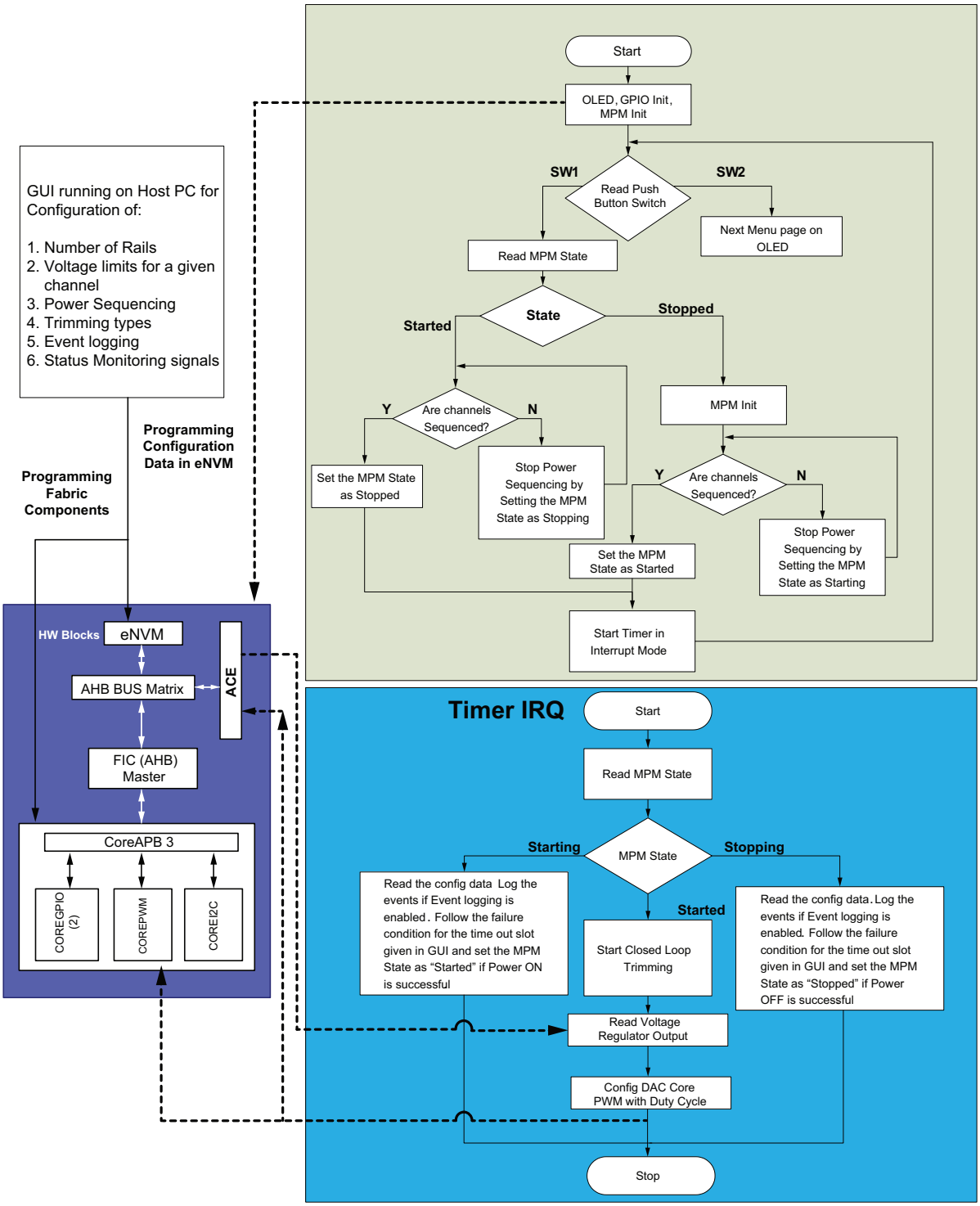


***Figure 14* • Flowchart For Complete MPM Data Flow**

Adding a New APOL Channel to MPM Reference Design

You can enhance the MPM reference design to add more channels as per the application requirements. The steps listed below explain the procedure on adding a new channel for the MPM reference design using the Libero Flow. Adding new channels can be implemented only with the ripple DAC implementation of CorePWM, and it is not possible to add new channel using on chip DAC, as all the available DAC channels are used in the existing MPM reference design. Adding a new channel to the existing MPM reference involves below steps:

1. Adding a monitoring channel to ADC to sample the regulator output
2. Adding extra channel to CoreGPIO to generate enable signal
3. Adding extra channel to CorePWM to generate ripple DAC

Adding a Monitoring Channel to ADC to Sample the Regulator Output

Adding a new channel to ADC to sample the regulator output is required for closed loop control of the MPM. The number of channels or regulators that can be controlled in closed loop is based on the number of ADC channels available on the device. The Table1 below gives the available Analog input channels for SmartFusion cSoC Device for closed loop control of MPM.

Table 1 • **Analog Input Channels For SmartFusion cSoC Device**

	A2F500(FG484)	A2F200	A2F060
ABPS	10	8	2
Direct ADC i/ps	12	8	11
CM0	5	4	1
TM0	5	4	1
Total Analog i/p	32	24	15
DACs	3	2	1

Use the following steps to add a monitoring channel. For design files of this application note, refer to "Appendix A" on page 38, in the Libero HW project.

1. Open the SmartDesign MSS configurator from the Libero-Project Flow.

2. Double-click the ACE block to configure. It looks similar to Figure 15.

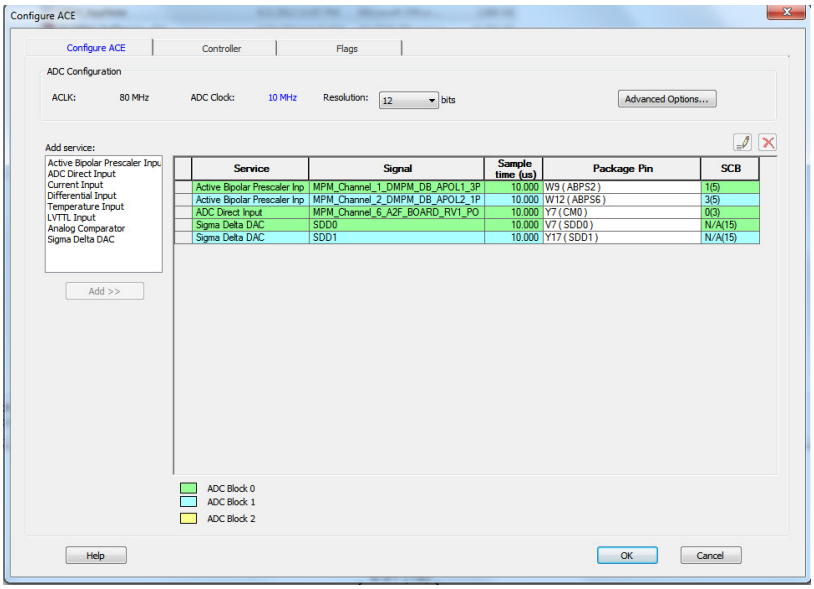


***Figure 15* • ACE Configurator Before Adding a Service**

3. Select **Active Bipolar Prescaler Input or ADC Direct Input** from the **Add service** list, and click **Add**. If the analog input value is less than the ADC voltage reference, choose the **ADC Direct Input** service for a higher accuracy for the application. An analog service configurator window opens as shown in below Figure 16, allowing you to customize the service.

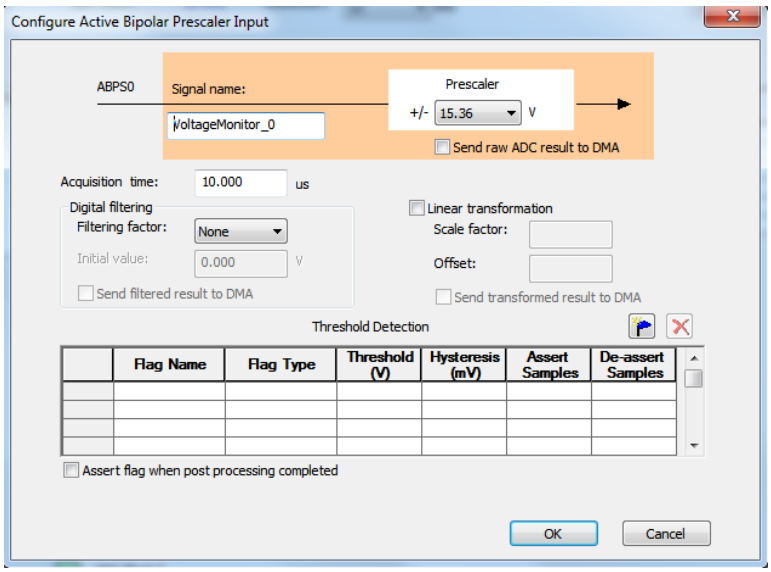


***Figure 16* • Active Bipolar Prescaler Configurator**

4. Enter the **Signal name** prefixed with 'MPM_Channel_<n>…', where '…' represents any other text needed to name the ACE channel signal and <n> is a unique MPM channel number between 1 and MPM_MAX_NUMBER_OF_CHANNELS under Signal Name option. For Example if you are adding 7th channel to the reference design, Signal name can be MPM_Channel_7_DMPM_DB_APOL3_1P5V. Assign flags UP and DOWN of flag types Over and

Under as required for application. The initial voltage values for these thresholds are not important and can be configured to any valid value.

The MPM driver dynamically reprograms these on the fly when managing the channels. For hysteresis-based thresholds, the MPM GUI specified hysteresis value is used. State filtered (assert/deassert samples) based thresholds are used as is in the ACE configuration. Channels meeting these criteria that are not given 'out of range' threshold ± hysteresis configurations via the MPM GUI are recognized and managed by the MPM firmware. After filling up all fields, Active Bipolar Prescaler Input Dialog Box looks like Figure 17.

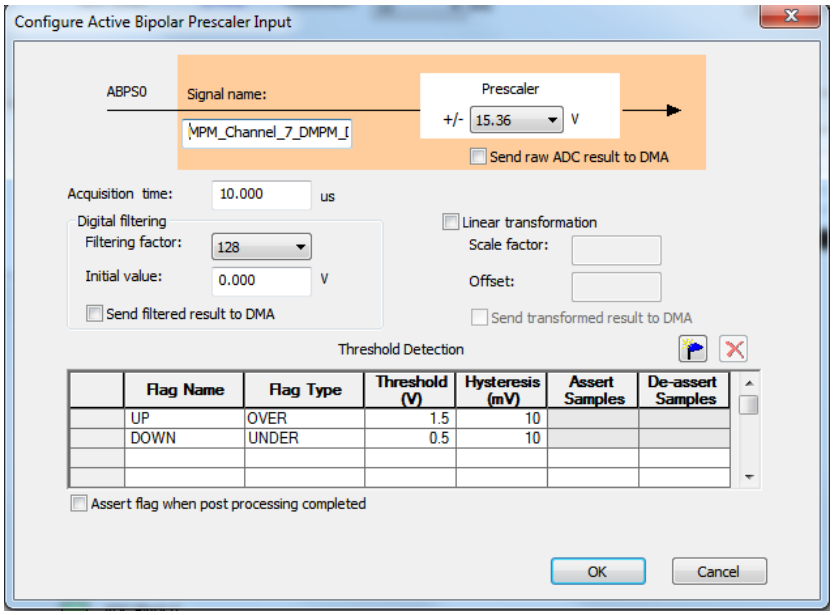


Figure 17 • **Active Bipolar Prescaler Configurator with Data**

5. Click **OK**. The ACE configurator looks like Figure 18. Assign the Package Pin value.

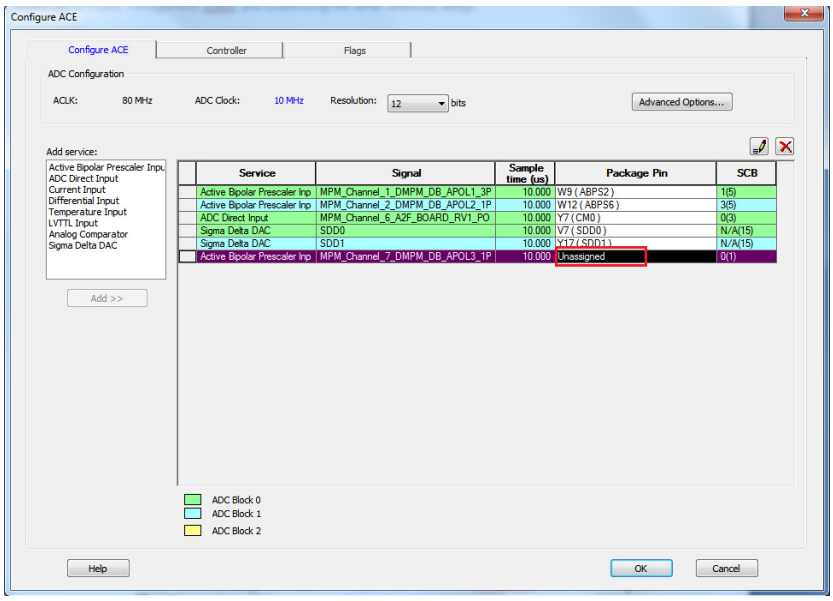


Figure 18 • **ACE Configurator After Adding a Service**

6. Typically, you will know with which analog pad the service needs to be assigned to based on your board design. The package pin assignment also determines which ABPS is selected. The Active Bipolar Prescaler Input (ABPS) service senses the voltage on an input pad and scales it to fit the range of the ADC (nominally 0 V to 2.56 V, if the internal voltage reference, Varef, is used). Currently, assign the value to Y11(ABPS7).The ACE configurator looks like in Figure 19.

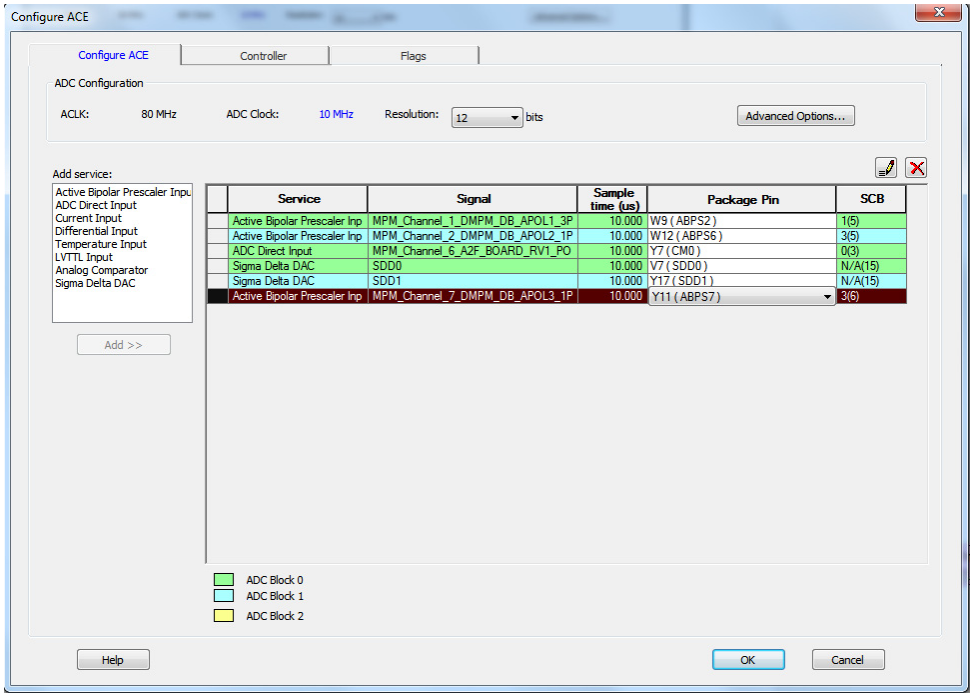


***Figure 19* • ACE Configurator After Adding a Service With Package Pin**

7. After assigning the pin value, click the **Controller** tab. Then click ADC1_MAIN in the Procedures box. As shown in Figure 20 on page 18, open the **Controller** tab in ACE configurator to configure Sample sequencing. The Sampling Sequence determines the order in which analog channels are sampled for a particular ADC.

The **Operating sequence entry** can be specified as **Auto** or **Manual**. Select **Auto**. As shown in Figure 20, the **Available signals** list shows the signals that are assigned to a specific ADC. You can use this list of signals to indicate which one you want to sample in Auto mode by moving these signals to and from the **Sampling rate**. Click on the ADC<n>_MAIN in which the newly added signal is and move it from the **Available signals** list to **Sampling rate** table. Click **Calculate Sequence and Actual Rate**. This creates a basic round robin sequence of your selected signals. Click **OK**.

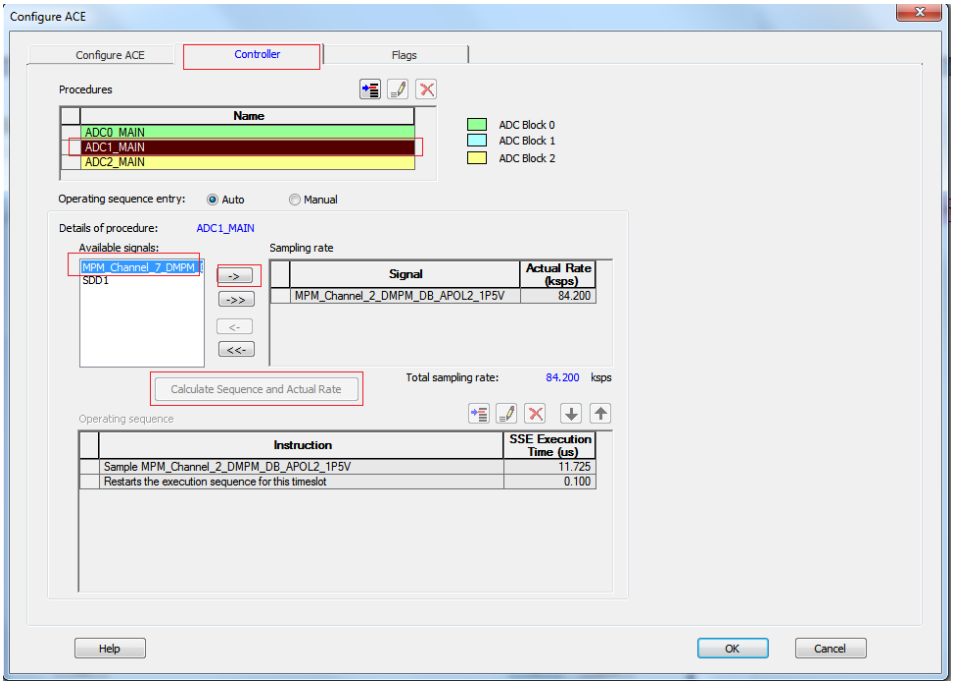


Figure 20 **• Controller Tab of ACE Configurator**

8. The MSS configurator appears as shown in Figure 21. You can observe the added signal MPM_Channel_7_DMPM_DB_APOL3_1P5V.

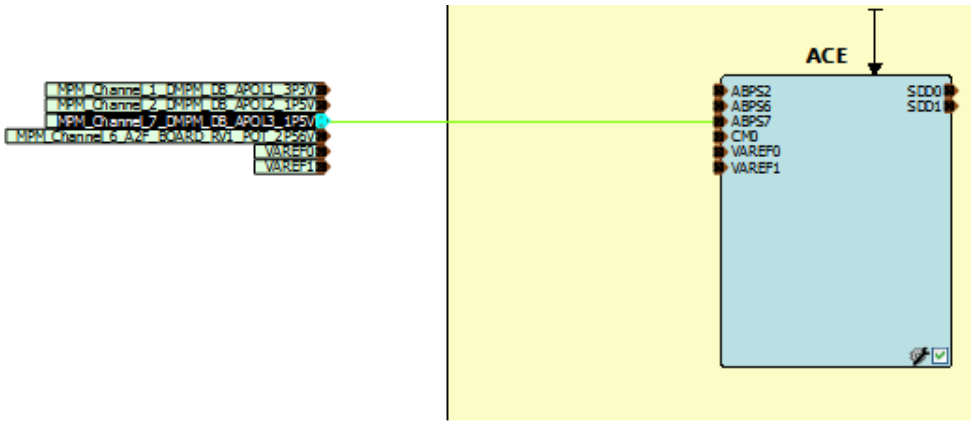


Figure 21 **• MSS Configurator With Added Signal**

Adding a Channel to CoreGPIO to Generate Enable Signal

Use the following steps to add a channel to CoreGPIO to generate enable signal in the Libero HW project:

1. Open the SmartDesign MSS configurator from Libero-Project Flow.
2. Double click MSS I/O block to configure, it is similar to Figure 22.

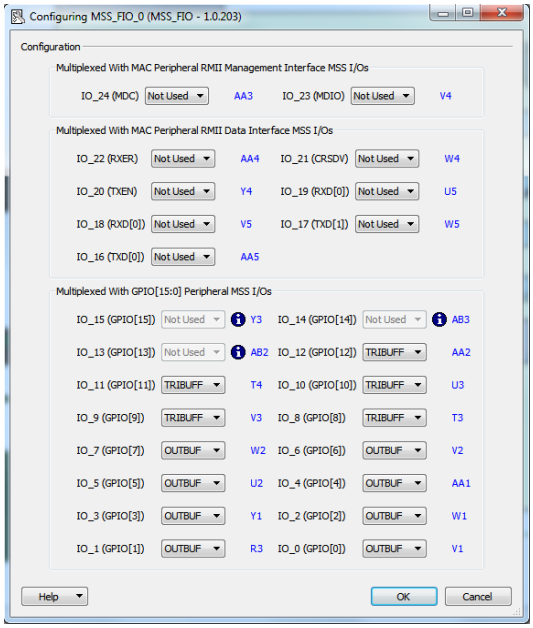


***Figure 22* • Configure MSS I/O Block**

3. Select TRIBUFF for IO_16 as shown in Figure 23.

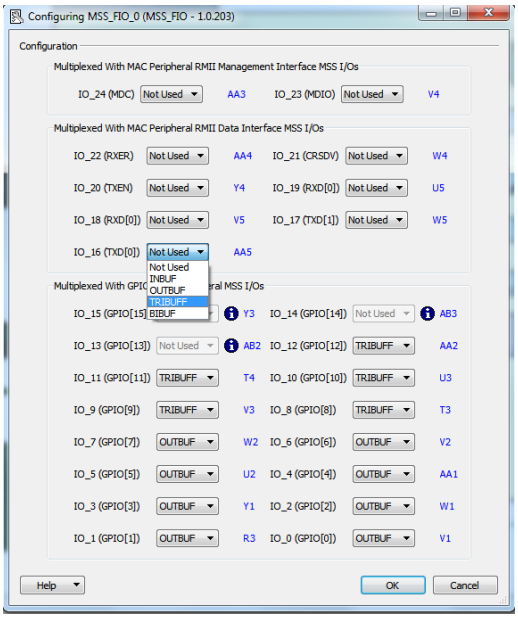


***Figure 23* • Selecting TRIBUFF for I/O_16**

4. Now MSS I/O Block in MSS configurator is visible as shown in Figure 24.

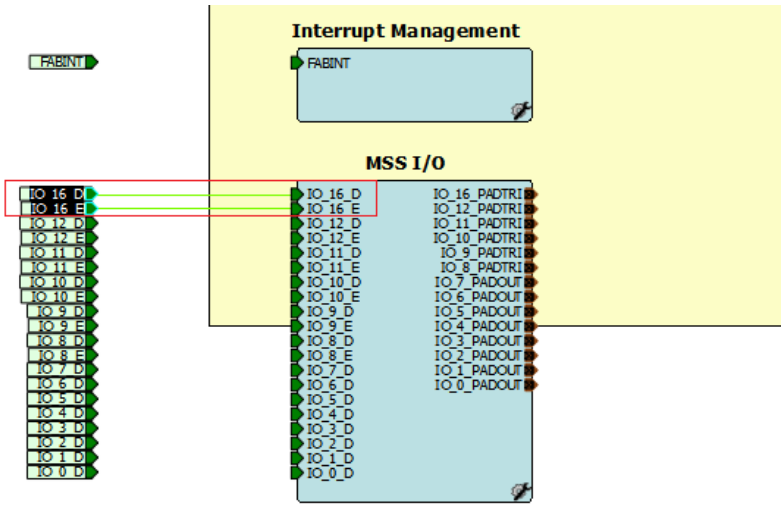


Figure 24 **• MSS I/O Block After Adding Extra I/O Port**

5. Save the changes and click Generate. Refer Changes in Softconsole section to update eNVM firmware client.
6. In Libero, right click SF_MPM_RefDesign_MSS block, select Update Instance(s) with Latest Component as shown in Figure 25.

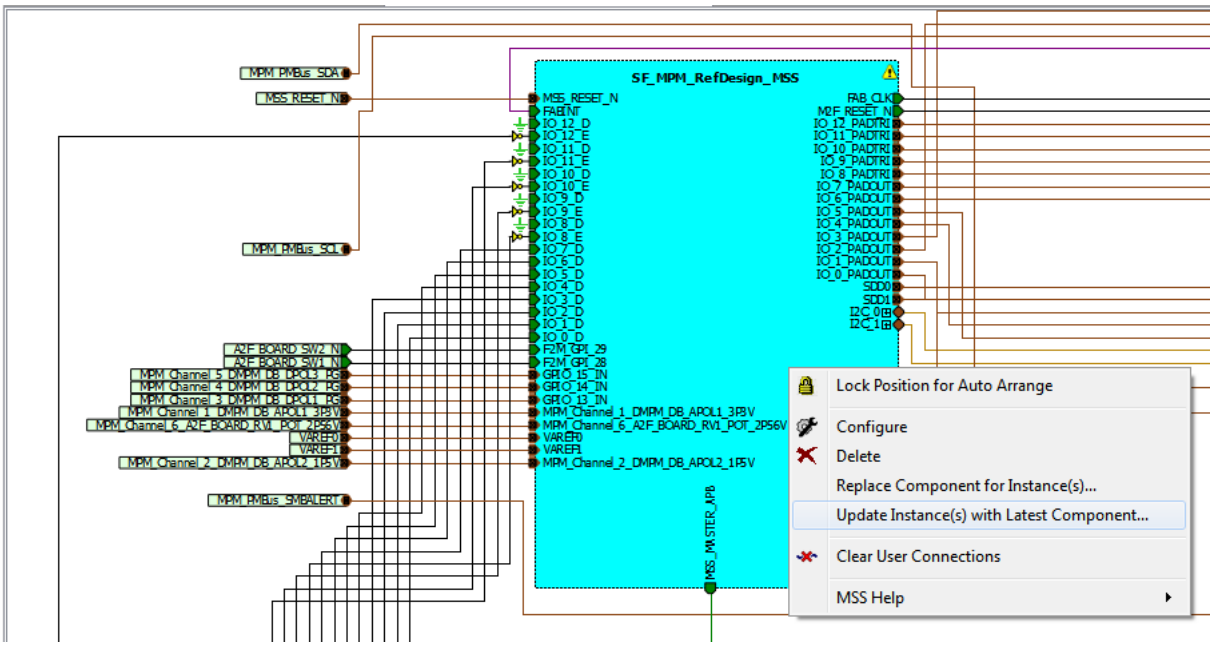


Figure 25 **• Updating MPM_MSS_0**

7. Right-click IO_16_D port of SF_MPM_RefDesign_MSS and select **Tie Low** as shown in Figure 26.

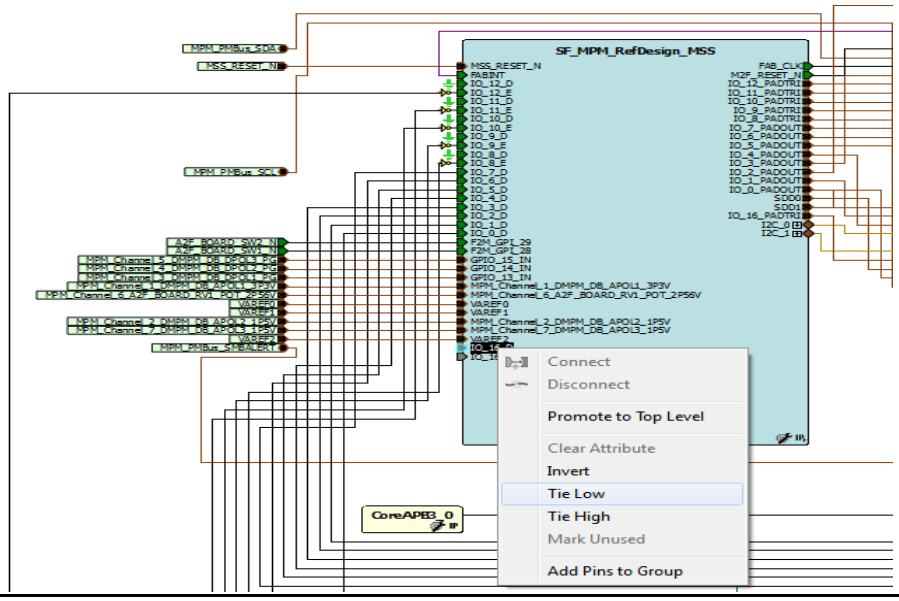


Figure 26 **• Selecting Tie Low for I/O Port**

8. Right-click IO_16_E port of SF_MPM_RefDesign_MSS and select Invert as shown in Figure 27.

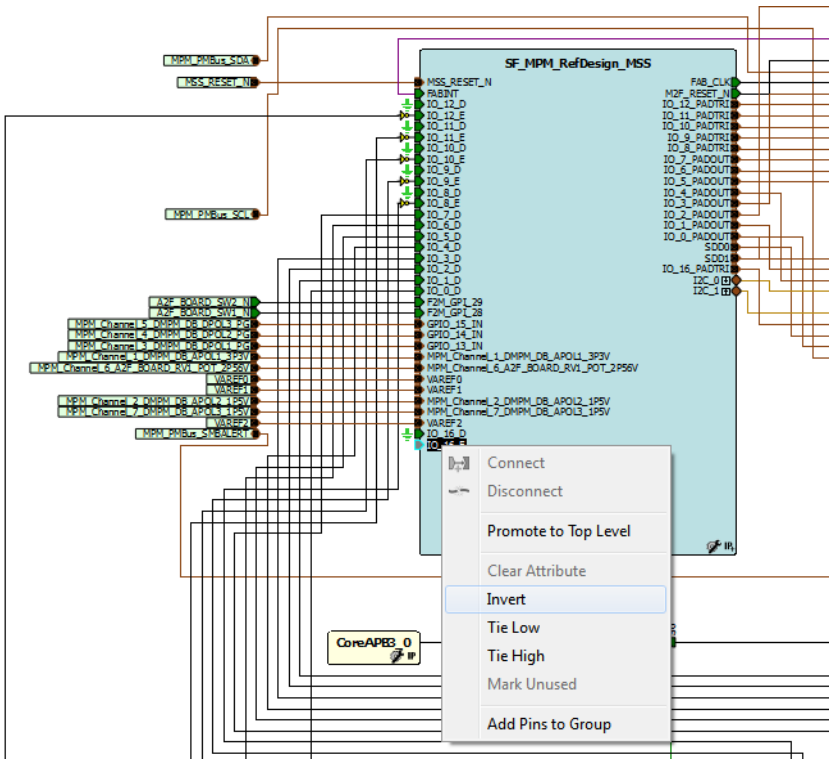


Figure 27 **• Selecting Invert for I/O Port**

9. Right-click the IO_16_PADTRI pin of SF_MPM_RefDesign_MSS and select Modify Port as shown in Figure 28.

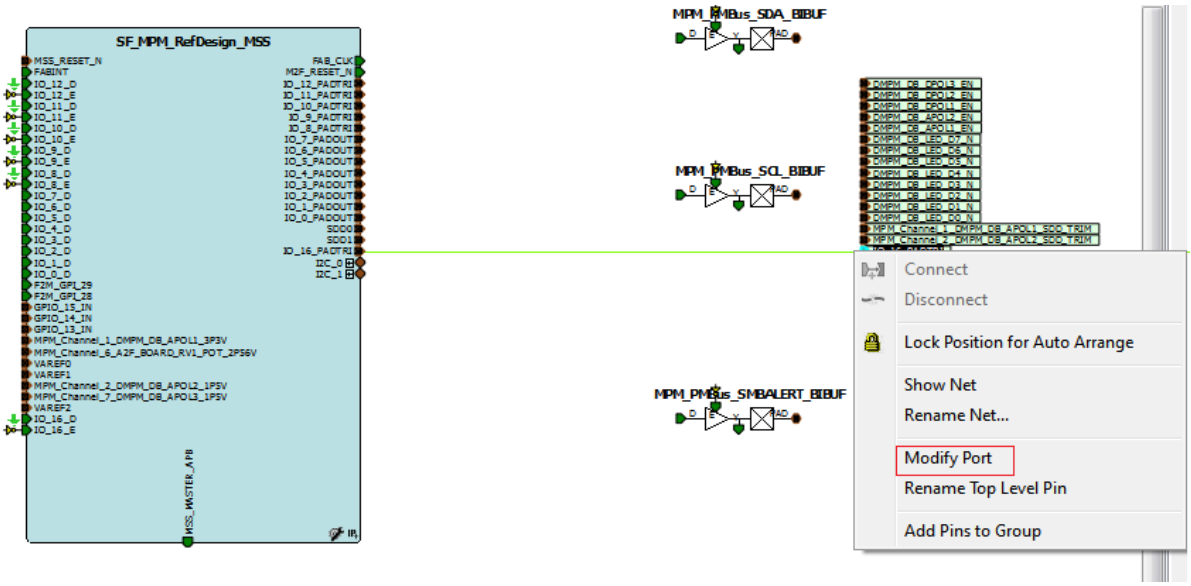


Figure 28 **• Selecting Modify Port**

10. Enter the name of the pin as DMPM_DB_APOL3_EN as shown in Figure 29 and click **OK**.

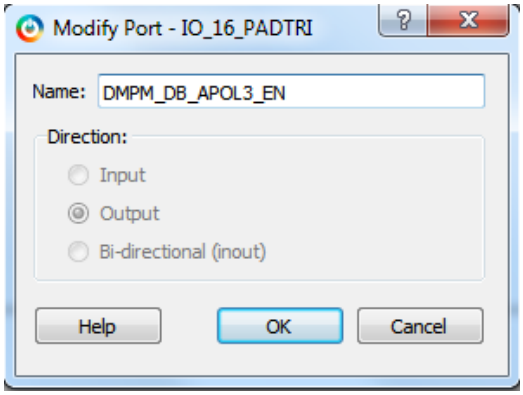


Figure 29 **• Modify Port Dialog Box**

11. Now right-click GPIO_OUT[31:0] of MPM_GPIO_Regulator_Enables and select Edit Slice as shown in Figure 30.

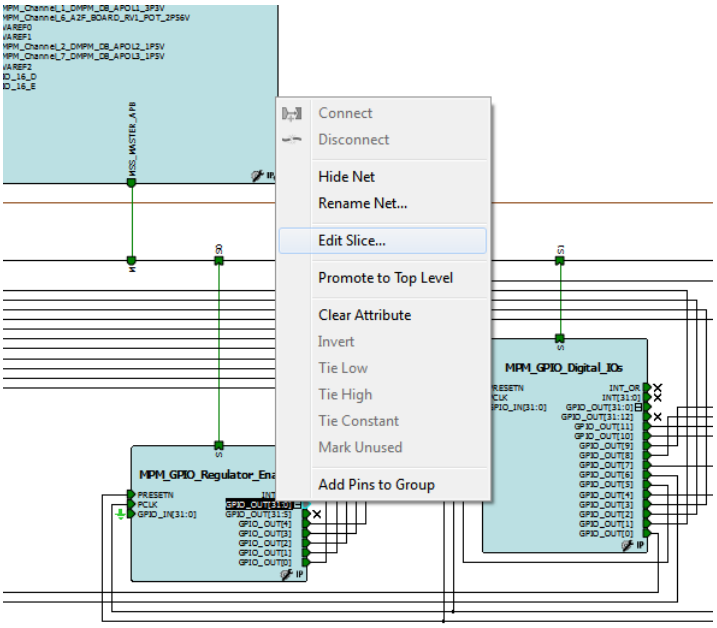


Figure 30 **• Selecting Edit Slice**

12. Click Add slice entry in the Edit Slices dialog box to add GPIO_OUT[5] and change the configuration as shown in Figure 31. Click OK.

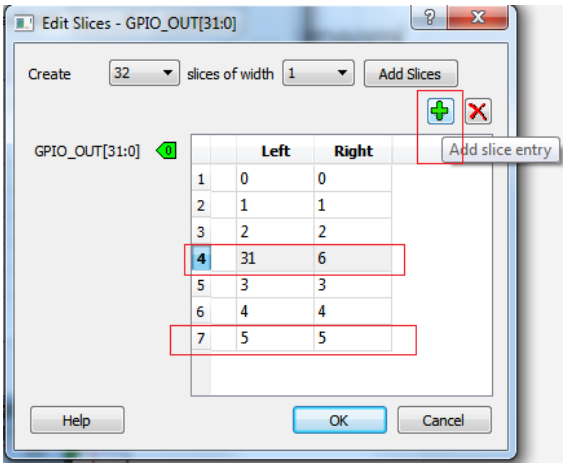


Figure 31 **• Edit Slices Dialog Box**

13. Connect GPIO_OUT[5] port of MPM_GPIO_Regulator_Enables to IO_16_E port of SF_MPM_RefDesign_MSS by using Connection Mode button as shown in Figure 32.

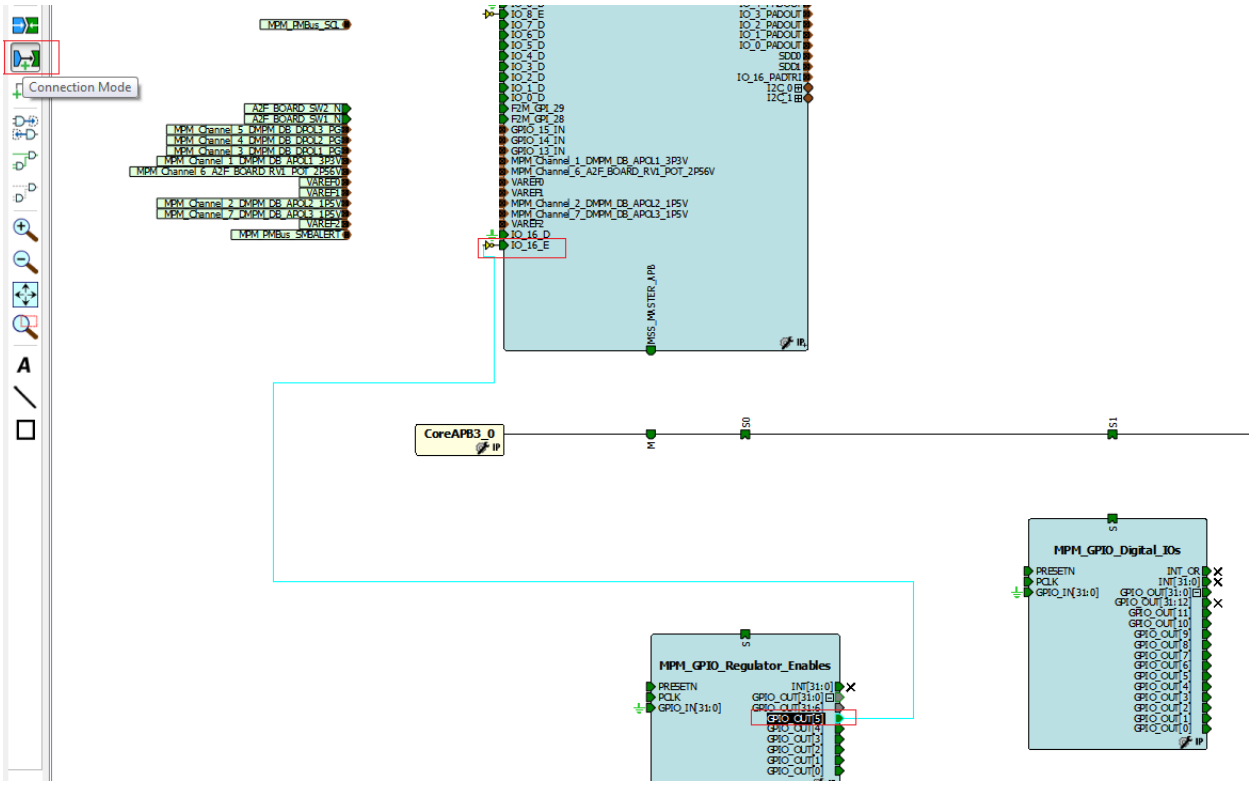


Figure 32 • **Connecting GPIO_OUT[5] and IO_16_E**

14. Right-click GPIO_OUT[31:6] of MPM_GPIO_Regulator_Enables and select Mark Unused as shown in Figure 33.

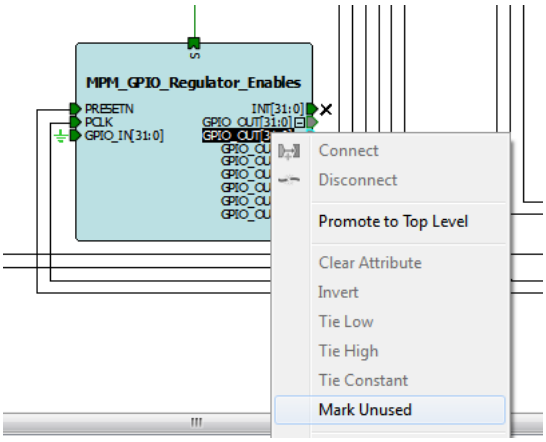


Figure 33 • **Selecting Mark Unused for GPIO_OUT[31:6]**

Adding a Channel to CorePWM to Generate Ripple DAC

Use the following steps to add a channel to CorePWM to generate ripple DAC:

1. Configure the connections of CorePWM (MPM_PWM_Trimming_Outputs). Double click CorePWM block of MPM_top and increment the Number of PWM Channels by 1 as shown in Figure 34.

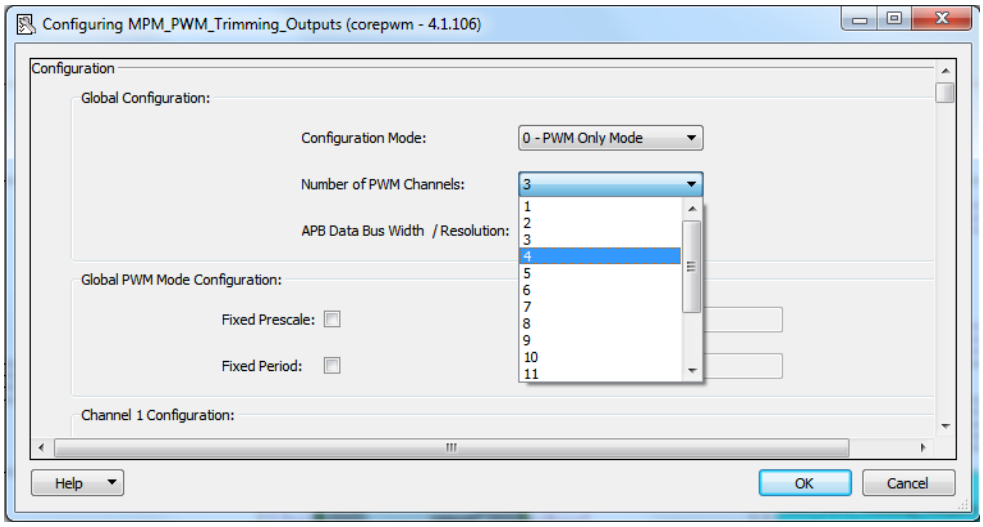


Figure 34 **• Configuring CorePWM**

2. Current design has 3 PWM channels. After adding the new PWM channel, select Low Ripple DAC mode for channel 3 as shown in Figure 35.

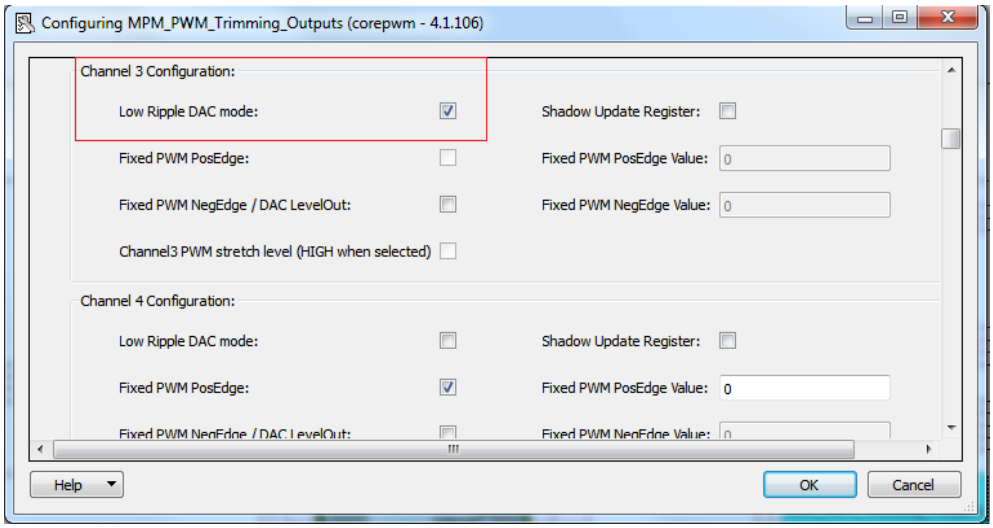


Figure 35 **• Selecting Low Ripple DAC Mode**

3. The CorePWM (MPM_PWM_Trimming_Outputs) block looks like Figure 36. Right-click the PWM[4:1] as shown in Figure 37 and select Edit Slice.

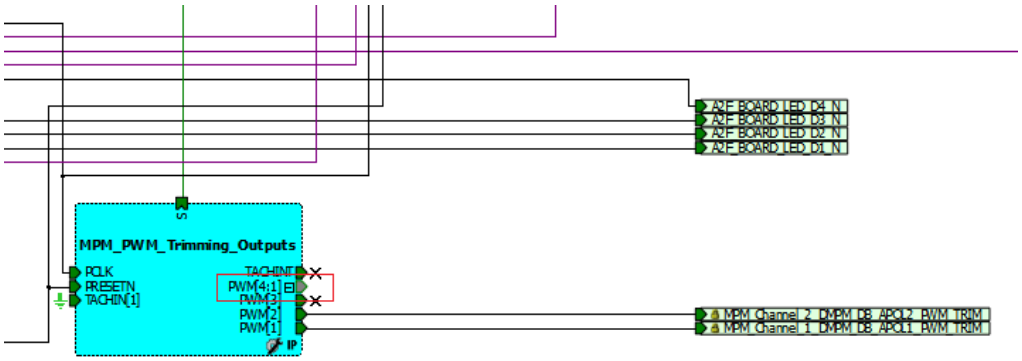


Figure 36 • **CorePWM Block After Adding PWM Channel**

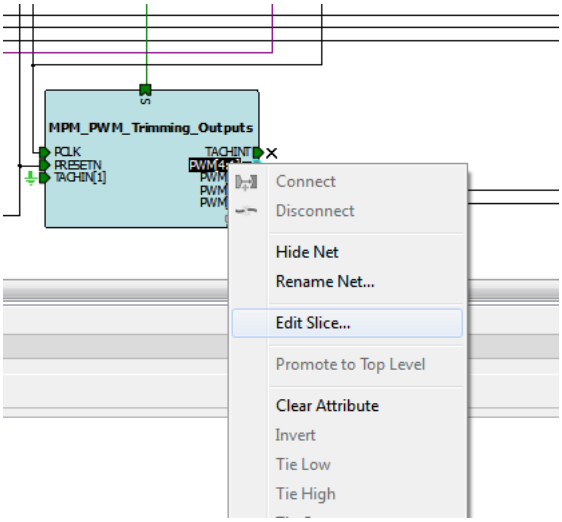


Figure 37 • **PWM Edit Slice**

4. Click Add Slices to add an extra PWM port as shown in Figure 38. Click **OK**.

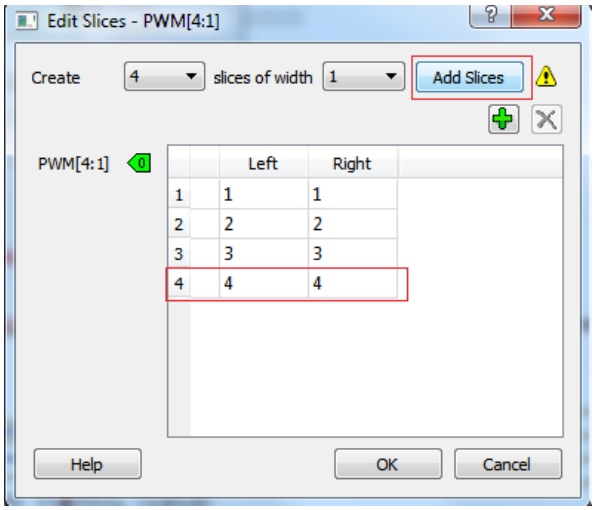


Figure 38 **• Edit Slices of CorePWM**

5. Right-click PWM[3] to clear the attribute as shown in Figure 39 and right click PWM[3] again to Promote to Top Level as shown in Figure 40 on page 27.

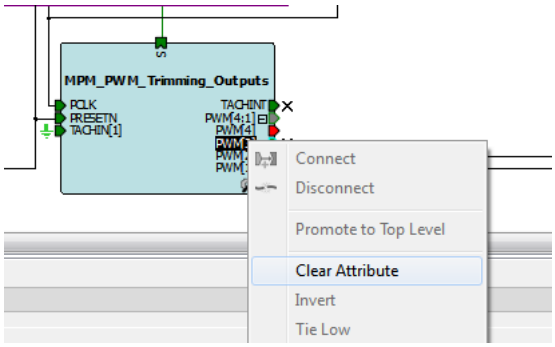


Figure 39 **• PWM[3] Clear Attribute**

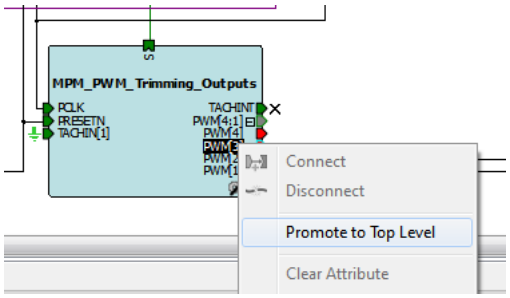


Figure 40 **• PWM[3] Promote to Top Level**

6. Modify the PWM[3] port as shown in below Figure 41. Enter the name of the port as MPM_Channel_7_DMPM_DB_APOL3_PWM_TRIM as shown in Figure 42.

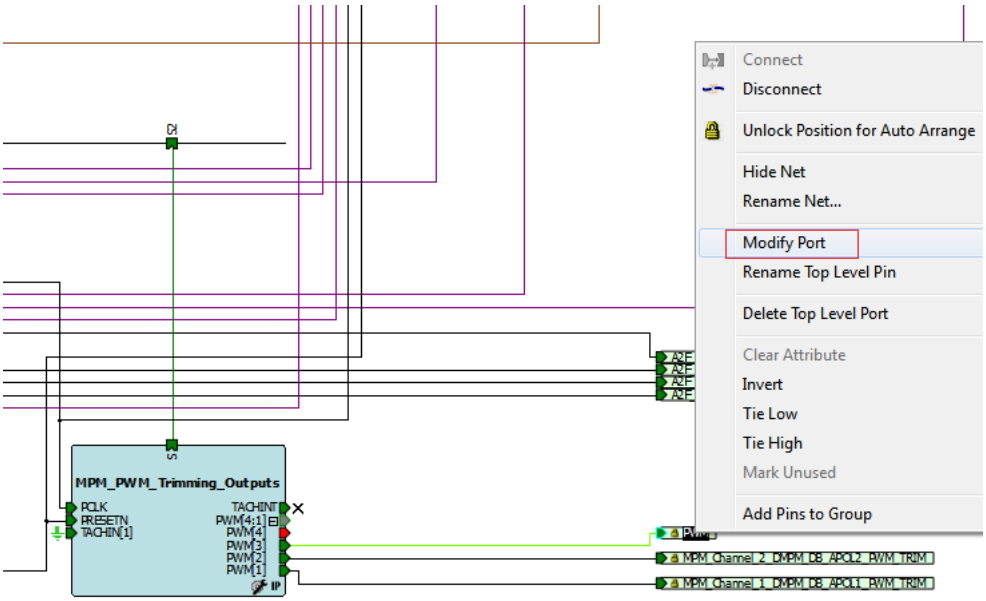


Figure 41 • **PWM[3] Select Modify Port**

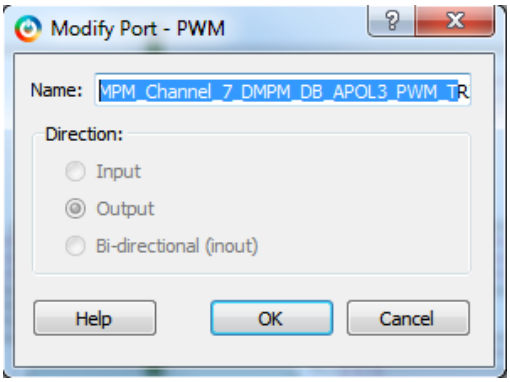


Figure 42 • **Modify Port Dialog Box**

7. Select Mark Unused for PWM[4] as shown in Figure 43.

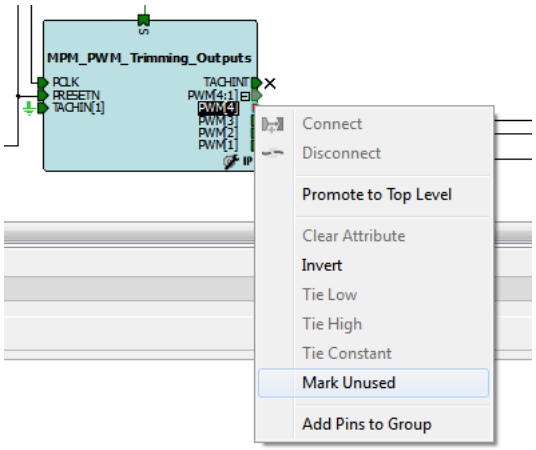


Figure 43 **• PWM[4] Mark Unused**

8. Save the design, right-click the SmartDesign canvas, and generate the component as shown in Figure 44.

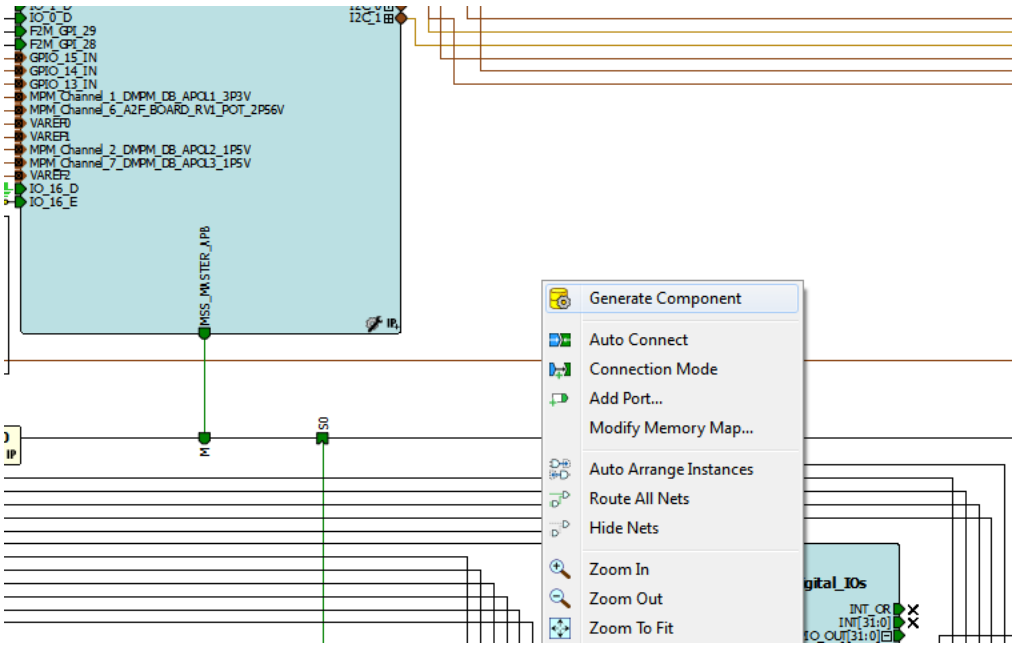
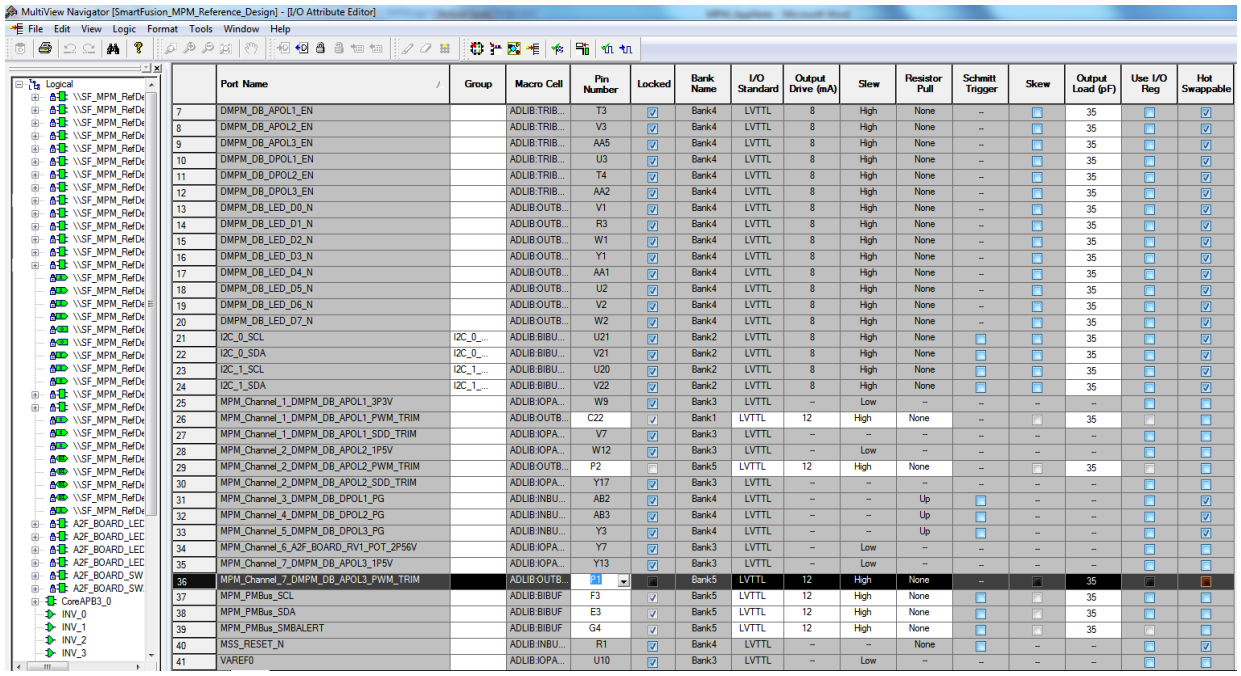


Figure 44 **• Selecting Generate Component**

9. In Libero project flow, after compiling, in I/O attribute editor, select corresponding pin number for the port MPM_Channel_7_DMPM_DB_APOL3_PWM_TRIM as shown in Figure 45. Select **Commit and Check** from the **File** menu and close the window.



	Port Name	Group	Macro Cell	Pin Number	Locked	Bank Name	I/O Standard	Output Drive (mA)	Slew	Resistor Pull	Schmitt Trigger	Skew	Output Load (pF)	Use I/O Reg	Hot Swappable
7	DMPM_DB_APOL1_EN		ADLIB:TRIB...	T3	✓	Bank4	LVTTL	8	High	None	--		35		✓
8	DMPM_DB_APOL2_EN		ADLIB:TRIB...	V3	✓	Bank4	LVTTL	8	High	None	--		35		✓
9	DMPM_DB_APOL3_EN		ADLIB:TRIB...	AA5	✓	Bank4	LVTTL	8	High	None	--		35		✓
10	DMPM_DB_DPOL1_EN		ADLIB:TRIB...	U3	✓	Bank4	LVTTL	8	High	None	--		35		✓
11	DMPM_DB_DPOL2_EN		ADLIB:TRIB...	T4	✓	Bank4	LVTTL	8	High	None	--		35		✓
12	DMPM_DB_DPOL3_EN		ADLIB:TRIB...	AA2	✓	Bank4	LVTTL	8	High	None	--		35		✓
13	DMPM_DB_LED_D0_N		ADLIB:OUTB...	V1	✓	Bank4	LVTTL	8	High	None	--		35		✓
14	DMPM_DB_LED_D1_N		ADLIB:OUTB...	R3	✓	Bank4	LVTTL	8	High	None	--		35		✓
15	DMPM_DB_LED_D2_N		ADLIB:OUTB...	W1	✓	Bank4	LVTTL	8	High	None	--		35		✓
16	DMPM_DB_LED_D3_N		ADLIB:OUTB...	Y1	✓	Bank4	LVTTL	8	High	None	--		35		✓
17	DMPM_DB_LED_D4_N		ADLIB:OUTB...	AA1	✓	Bank4	LVTTL	8	High	None	--		35		✓
18	DMPM_DB_LED_D5_N		ADLIB:OUTB...	U2	✓	Bank4	LVTTL	8	High	None	--		35		✓
19	DMPM_DB_LED_D6_N		ADLIB:OUTB...	V2	✓	Bank4	LVTTL	8	High	None	--		35		✓
20	DMPM_DB_LED_D7_N		ADLIB:OUTB...	W2	✓	Bank4	LVTTL	8	High	None	--		35		✓
21	I2C_0_SCL	I2C_0_...	ADLIB:BIBU...	U21	✓	Bank2	LVTTL	8	High	None			35		✓
22	I2C_0_SDA	I2C_0_...	ADLIB:BIBU...	V21	✓	Bank2	LVTTL	8	High	None			35		✓
23	I2C_1_SCL	I2C_1_...	ADLIB:BIBU...	U20	✓	Bank2	LVTTL	8	High	None			35		✓
24	I2C_1_SDA	I2C_1_...	ADLIB:BIBU...	V22	✓	Bank2	LVTTL	8	High	None			35		✓
25	MPM_Channel_1_DMPM_DB_APOL1_3P3V		ADLIB:IOPA...	W9	✓	Bank3	LVTTL	--	Low	--	--	--	--		
26	MPM_Channel_1_DMPM_DB_APOL1_PWM_TRIM		ADLIB:OUTB...	C22	✓	Bank1	LVTTL	12	High	None	--		35		
27	MPM_Channel_1_DMPM_DB_APOL1_SDD_TRIM		ADLIB:IOPA...	V7	✓	Bank3	LVTTL	--	--	--	--	--	--		
28	MPM_Channel_2_DMPM_DB_APOL2_1P5V		ADLIB:IOPA...	W12	✓	Bank3	LVTTL	--	Low	--	--	--	--		
29	MPM_Channel_2_DMPM_DB_APOL2_PWM_TRIM		ADLIB:OUTB...	P2		Bank5	LVTTL	12	High	None	--		35		
30	MPM_Channel_2_DMPM_DB_APOL2_SDD_TRIM		ADLIB:IOPA...	Y17	✓	Bank3	LVTTL	--	--	--	--	--	--		
31	MPM_Channel_3_DMPM_DB_DPOL1_PG		ADLIB:INBU...	AB2	✓	Bank4	LVTTL	--	--	Up		--	--		✓
32	MPM_Channel_4_DMPM_DB_DPOL2_PG		ADLIB:INBU...	AB3	✓	Bank4	LVTTL	--	--	Up		--	--		✓
33	MPM_Channel_5_DMPM_DB_DPOL3_PG		ADLIB:INBU...	Y3	✓	Bank4	LVTTL	--	--	Up		--	--		✓
34	MPM_Channel_6_A2F_BOARD_RV1_POT_2P56V		ADLIB:IOPA...	Y7	✓	Bank3	LVTTL	--	Low	--	--	--	--		
35	MPM_Channel_7_DMPM_DB_APOL3_1P5V		ADLIB:IOPA...	Y13	✓	Bank3	LVTTL	--	Low	--	--	--	--		
36	MPM_Channel_7_DMPM_DB_APOL3_PWM_TRIM		ADLIB:OUTB...	P1		Bank5	LVTTL	12	High	None	--		35		
37	MPM_PMBus_SCL		ADLIB:BIBUF	F3	✓	Bank5	LVTTL	12	High	None			35		
38	MPM_PMBus_SDA		ADLIB:BIBUF	E3	✓	Bank5	LVTTL	12	High	None			35		
39	MPM_PMBus_SMBALERT		ADLIB:BIBUF	G4	✓	Bank5	LVTTL	12	High	None			35		
40	MSS_RESET_N		ADLIB:INBU...	R1	✓	Bank4	LVTTL	--	--	None		--	--		✓
41	VAREF0		ADLIB:IOPA...	U10	✓	Bank3	LVTTL	--	Low	--	--	--	--		

***Figure 45* • Selecting Pin Numbers in the I/O Attribute Editor**

10. Perform the place and route and program the FPGA.

Adding a New DPOL Channel to MPM Reference Design

The DPOL for the channel must be connected to the MPM PMBus and each DPOL on the MPM PMBus must obviously have a unique I2C/PMBus slave address. MPM GUI uses I2C/PMBus slave address to get the details of the DPOL. Follow the below steps to add the new DPOL channel. Assuming you are adding new channel of DPOL as 8th channel.

1. Open the SmartDesign MSS configurator from Libero-Project Flow and double click GPIO block to configure and select Input for GPIO_16 for the **Power Good (PG) DPOL output signal** as shown in Figure 46.

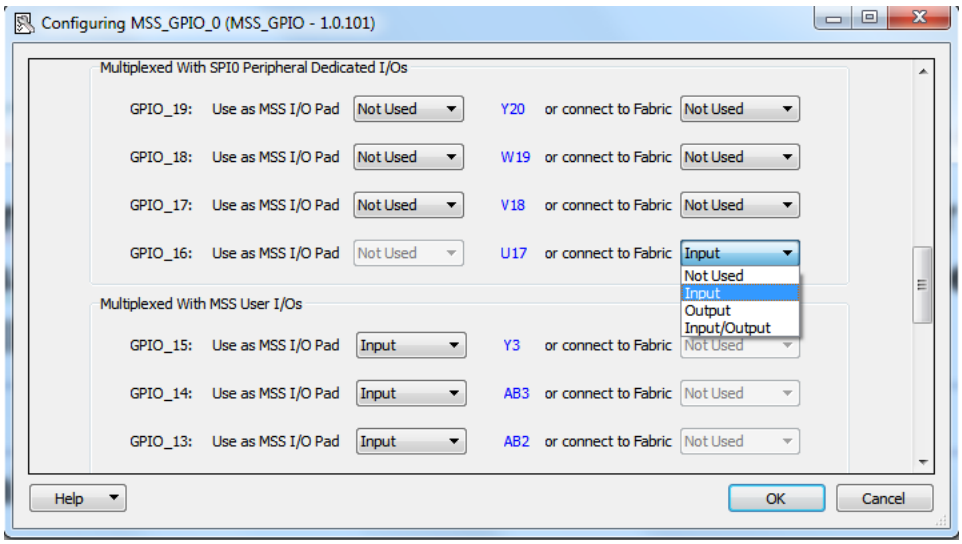
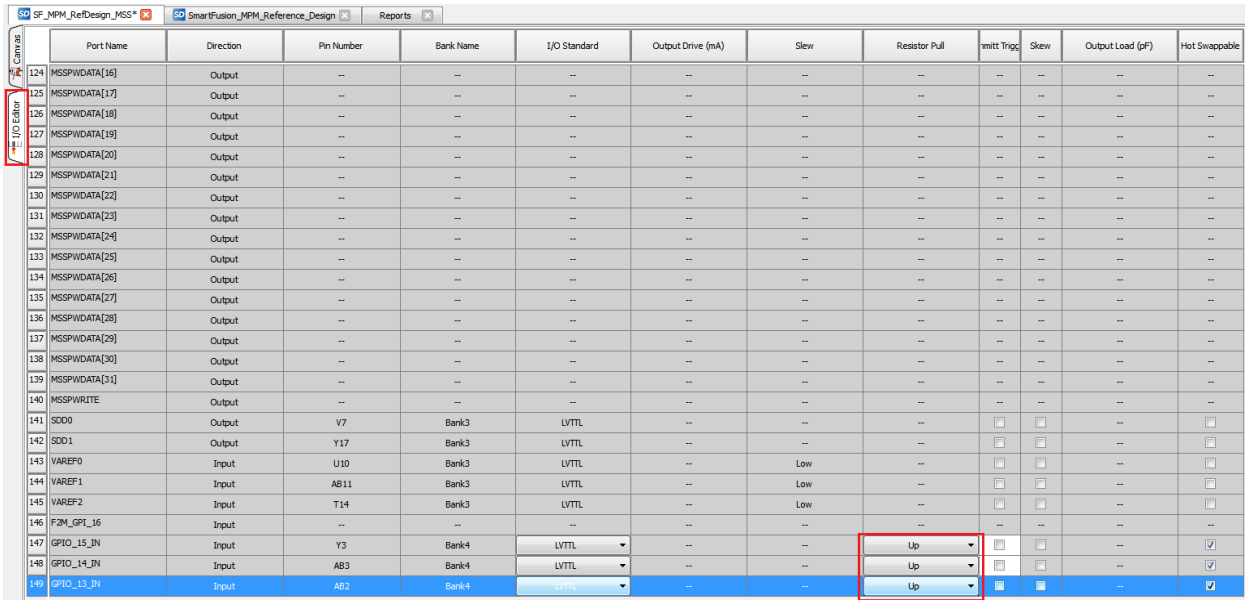


***Figure 46* • Configuring MSS GPIO**

2. Ensure that resistor pull up for GPIO_13, GPIO_14, and GPIO_15 ports in I/O Editor of MSS configurator as shown in Figure 47.



	Port Name	Direction	Pin Number	Bank Name	I/O Standard	Output Drive (mA)	Slew	Resistor Pull	mitt Trigg	Skew	Output Load (pF)	Hot Swappable
124	MSSPWDATA[16]	Output	--	--	--	--	--	--	--	--	--	--
125	MSSPWDATA[17]	Output	--	--	--	--	--	--	--	--	--	--
126	MSSPWDATA[18]	Output	--	--	--	--	--	--	--	--	--	--
127	MSSPWDATA[19]	Output	--	--	--	--	--	--	--	--	--	--
128	MSSPWDATA[20]	Output	--	--	--	--	--	--	--	--	--	--
129	MSSPWDATA[21]	Output	--	--	--	--	--	--	--	--	--	--
130	MSSPWDATA[22]	Output	--	--	--	--	--	--	--	--	--	--
131	MSSPWDATA[23]	Output	--	--	--	--	--	--	--	--	--	--
132	MSSPWDATA[24]	Output	--	--	--	--	--	--	--	--	--	--
133	MSSPWDATA[25]	Output	--	--	--	--	--	--	--	--	--	--
134	MSSPWDATA[26]	Output	--	--	--	--	--	--	--	--	--	--
135	MSSPWDATA[27]	Output	--	--	--	--	--	--	--	--	--	--
136	MSSPWDATA[28]	Output	--	--	--	--	--	--	--	--	--	--
137	MSSPWDATA[29]	Output	--	--	--	--	--	--	--	--	--	--
138	MSSPWDATA[30]	Output	--	--	--	--	--	--	--	--	--	--
139	MSSPWDATA[31]	Output	--	--	--	--	--	--	--	--	--	--
140	MSSPWRITE	Output	--	--	--	--	--	--	--	--	--	--
141	SDD0	Output	V7	Bank3	LVTTL	--	--	--			--	
142	SDD1	Output	Y17	Bank3	LVTTL	--	--	--			--	
143	VAREF0	Input	U10	Bank3	LVTTL	--	Low	--			--	
144	VAREF1	Input	AB11	Bank3	LVTTL	--	Low	--			--	
145	VAREF2	Input	T14	Bank3	LVTTL	--	Low	--			--	
146	F2M_GPI_16	Input	--	--	--	--	--	--	--	--	--	--
147	GPIO_15_IN	Input	Y3	Bank4	LVTTL	--	--	Up			--	✓
148	GPIO_14_IN	Input	AB3	Bank4	LVTTL	--	--	Up			--	✓
149	GPIO_13_IN	Input	AB2	Bank4	LVTTL	--	--	Up			--	✓

***Figure 47* • I/O Editor**

3. Follow the section Adding a Channel to CoreGPIO to Generate Enable Signal to generate the enable signal for the new channel of DPOL. While adding new channel to CoreGPIO, you have to perform the following steps.
   - Select TRIBUFF for IO_17 of MSS I/O block.
   - Refer "Changes in Softconsole" on page 36 to update eNVM firmware client.
   - Select Tie Low for IO_17_D port of SF_MPM_RefDesign_MSS
   - Select Invert for IO_17_E port of SF_MPM_RefDesign_MSS.
   - Modify the IO_17_PADTRI pin of SF_MPM_RefDesign_MSS and enter the name as DMPM_DB_DPOL4_EN.
   - To add GPIO_OUT[6] to MPM_GPIO_Regulator_Enables, configure GPIO_OUT[31:0] as shown in Figure 48.

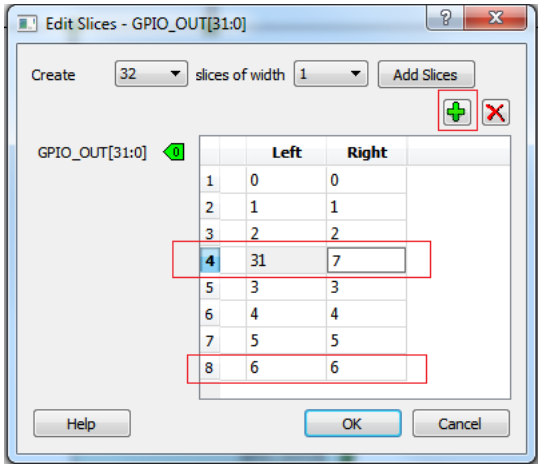


Figure 48 **• Configuring GPIO_OUT[31:0]**

– Connect GPIO_OUT[6] port of MPM_GPIO_Regulator_Enables to IO_17_E port of SF_MPM_RefDesign_MSS. Now the CoreGPIO (MPM_GPIO_Regulator_Enables) block is similar to Figure 49.

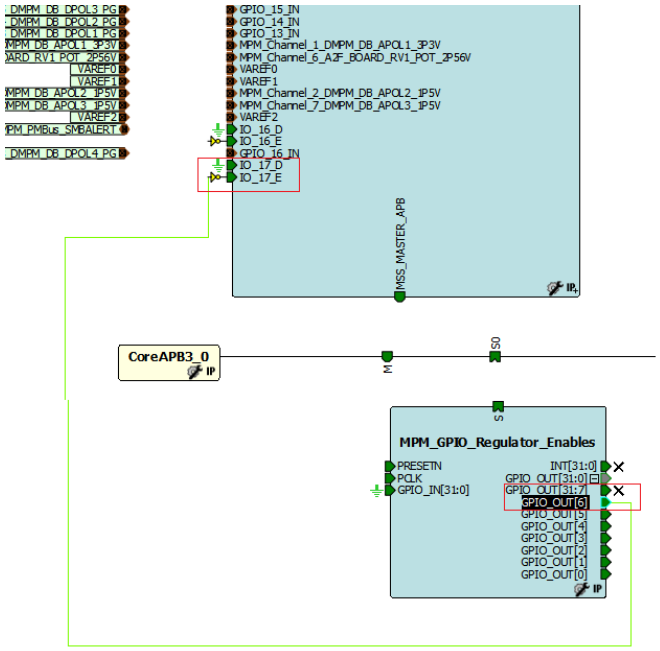


Figure 49 **• CoreGPIO Block After Configuration**

4. Right-click the F2M_GPI_16 pin of SF_MPM_RefDesign_MSS and select Promote to Top Level as shown in Figure 50.

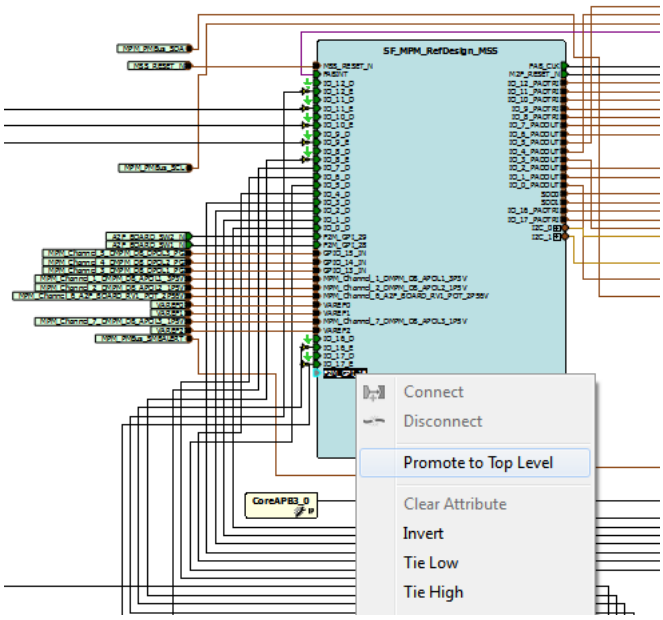


Figure 50 **• Promote to Top Level**

5. Right-click the F2M_GPI_16 pin of SF_MPM_RefDesign_MSS and select Modify port as shown in Figure 51. Enter port name as shown in Figure 52 on page 34.

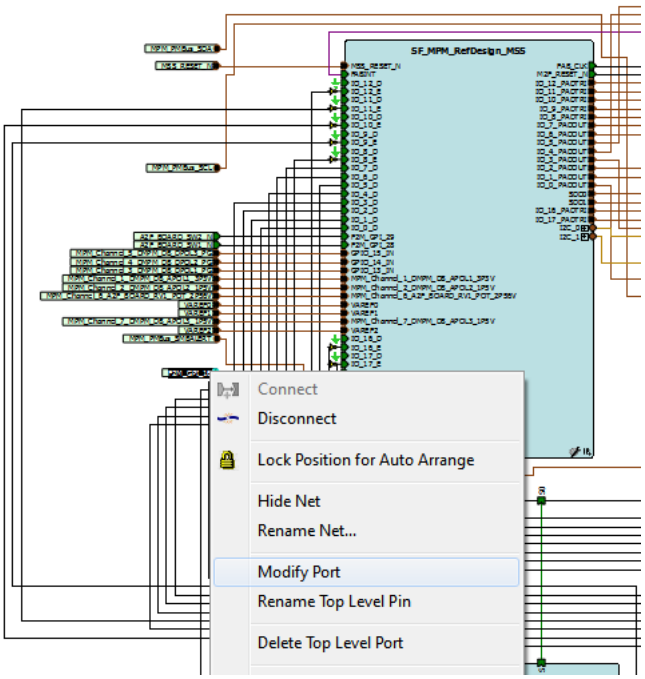


Figure 51 **• Selecting Modify Port**

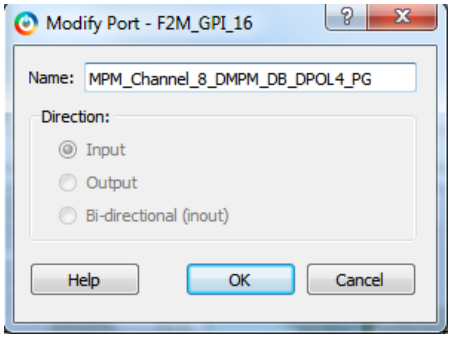
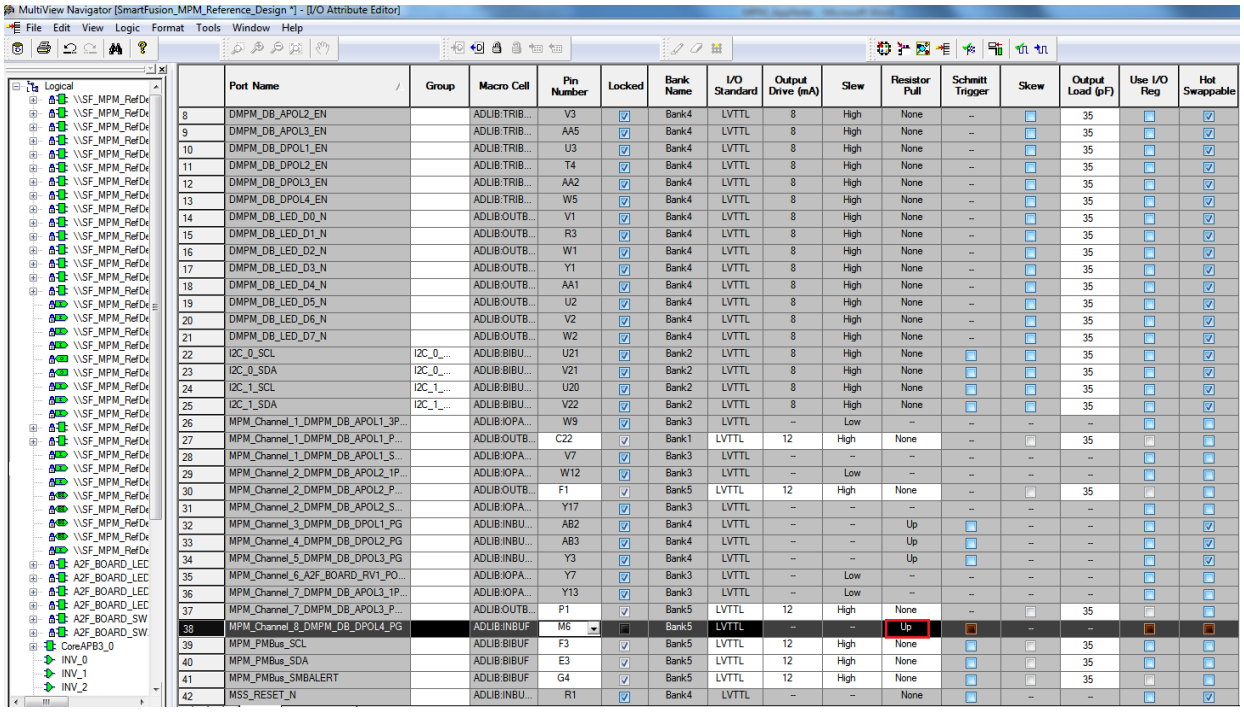


Figure 52 **• Modify Port Dialog Box**

6. Save the design, right-click the SmartDesign canvas, and generate the component as shown in Figure 44 on page 29.

7. In Libero project flow, after compiling, in I/O attribute editor, select corresponding pin number for the port MPM_Channel_8_DMPM_DB_DPOL4_PG and select Up for Resistor Pull as shown in Figure 53. Select Commit and Check from the File menu and close the window.



	Port Name	Group	Macro Cell	Pin Number	Locked	Bank Name	I/O Standard	Output Drive (mA)	Slew	Resistor Pull	Schmitt Trigger	Skew	Output Load (pF)	Use I/O Reg	Hot Swappable
8	DMPM_DB_APOL2_EN		ADLIB:TRIB...	V3	☑	Bank4	LVTTL	8	High	None	--	☐	35	☐	☑
9	DMPM_DB_APOL3_EN		ADLIB:TRIB...	AA5	☑	Bank4	LVTTL	8	High	None	--	☐	35	☐	☑
10	DMPM_DB_DPOL1_EN		ADLIB:TRIB...	U3	☑	Bank4	LVTTL	8	High	None	--	☐	35	☐	☑
11	DMPM_DB_DPOL2_EN		ADLIB:TRIB...	T4	☑	Bank4	LVTTL	8	High	None	--	☐	35	☐	☑
12	DMPM_DB_DPOL3_EN		ADLIB:TRIB...	AA2	☑	Bank4	LVTTL	8	High	None	--	☐	35	☐	☑
13	DMPM_DB_DPOL4_EN		ADLIB:TRIB...	W5	☑	Bank4	LVTTL	8	High	None	--	☐	35	☐	☑
14	DMPM_DB_LED_D0_N		ADLIB:OUTB...	V1	☑	Bank4	LVTTL	8	High	None	--	☐	35	☐	☑
15	DMPM_DB_LED_D1_N		ADLIB:OUTB...	R3	☑	Bank4	LVTTL	8	High	None	--	☐	35	☐	☑
16	DMPM_DB_LED_D2_N		ADLIB:OUTB...	W1	☑	Bank4	LVTTL	8	High	None	--	☐	35	☐	☑
17	DMPM_DB_LED_D3_N		ADLIB:OUTB...	Y1	☑	Bank4	LVTTL	8	High	None	--	☐	35	☐	☑
18	DMPM_DB_LED_D4_N		ADLIB:OUTB...	AA1	☑	Bank4	LVTTL	8	High	None	--	☐	35	☐	☑
19	DMPM_DB_LED_D5_N		ADLIB:OUTB...	U2	☑	Bank4	LVTTL	8	High	None	--	☐	35	☐	☑
20	DMPM_DB_LED_D6_N		ADLIB:OUTB...	V2	☑	Bank4	LVTTL	8	High	None	--	☐	35	☐	☑
21	DMPM_DB_LED_D7_N		ADLIB:OUTB...	W2	☑	Bank4	LVTTL	8	High	None	--	☐	35	☐	☑
22	I2C_0_SCL	I2C_0_...	ADLIB:BIBU...	U21	☑	Bank2	LVTTL	8	High	None	☐	☐	35	☐	☑
23	I2C_0_SDA	I2C_0_...	ADLIB:BIBU...	V21	☑	Bank2	LVTTL	8	High	None	☐	☐	35	☐	☑
24	I2C_1_SCL	I2C_1_...	ADLIB:BIBU...	U20	☑	Bank2	LVTTL	8	High	None	☐	☐	35	☐	☑
25	I2C_1_SDA	I2C_1_...	ADLIB:BIBU...	V22	☑	Bank2	LVTTL	8	High	None	☐	☐	35	☐	☑
26	MPM_Channel_1_DMPM_DB_APOL1_3P...		ADLIB:IOPA...	W9	☑	Bank3	LVTTL	--	Low	--	--	--	--	☐	☐
27	MPM_Channel_1_DMPM_DB_APOL1_P...		ADLIB:OUTB...	C22	☑	Bank1	LVTTL	12	High	None	--	☐	35	☐	☐
28	MPM_Channel_1_DMPM_DB_APOL1_S...		ADLIB:IOPA...	V7	☑	Bank3	LVTTL	--	--	--	--	--	--	☐	☐
29	MPM_Channel_2_DMPM_DB_APOL2_1P...		ADLIB:IOPA...	W12	☑	Bank3	LVTTL	--	Low	--	--	--	--	☐	☐
30	MPM_Channel_2_DMPM_DB_APOL2_P...		ADLIB:OUTB...	F1	☑	Bank5	LVTTL	12	High	None	--	☐	35	☐	☐
31	MPM_Channel_2_DMPM_DB_APOL2_S...		ADLIB:IOPA...	Y17	☑	Bank3	LVTTL	--	--	--	--	--	--	☐	☐
32	MPM_Channel_3_DMPM_DB_DPOL1_PG		ADLIB:INBU...	AB2	☑	Bank4	LVTTL	--	--	Up	☐	--	--	☐	☑
33	MPM_Channel_4_DMPM_DB_DPOL2_PG		ADLIB:INBU...	AB3	☑	Bank4	LVTTL	--	--	Up	☐	--	--	☐	☑
34	MPM_Channel_5_DMPM_DB_DPOL3_PG		ADLIB:INBU...	Y3	☑	Bank4	LVTTL	--	--	Up	☐	--	--	☐	☑
35	MPM_Channel_6_A2F_BOARD_RV1_PO...		ADLIB:IOPA...	Y7	☑	Bank3	LVTTL	--	Low	--	--	--	--	☐	☐
36	MPM_Channel_7_DMPM_DB_APOL3_1P...		ADLIB:IOPA...	Y13	☑	Bank3	LVTTL	--	Low	--	--	--	--	☐	☐
37	MPM_Channel_7_DMPM_DB_APOL3_P...		ADLIB:OUTB...	P1	☑	Bank5	LVTTL	12	High	None	--	☐	35	☐	☐
38	MPM_Channel_8_DMPM_DB_DPOL4_PG		ADLIB:INBUF	M6	☐	Bank5	LVTTL	--	--	Up	☐	--	--	☐	☐
39	MPM_PMBus_SCL		ADLIB:BIBUF	F3	☑	Bank5	LVTTL	12	High	None	☐	☐	35	☐	☐
40	MPM_PMBus_SDA		ADLIB:BIBUF	E3	☑	Bank5	LVTTL	12	High	None	☐	☐	35	☐	☐
41	MPM_PMBus_SMBALERT		ADLIB:BIBUF	G4	☑	Bank5	LVTTL	12	High	None	☐	☐	35	☐	☐
42	MSS_RESET_N		ADLIB:INBU...	R1	☑	Bank4	LVTTL	--	--	None	☐	--	--	☐	☑

***Figure 53* • Selecting Pin Numbers in I/O Attribute Editor**

8. Perform the place and route and program the FPGA.

Changes in Softconsole

If you are adding the new channel of APOL, delete mss_ace folder under drivers_config folder and add the same from firmware folder (C:\Microsemi\SF_MPM_RefDesign_v4.0\design_files\Libero_project\SF_MPM_RefDesign\firmware\drivers_config).

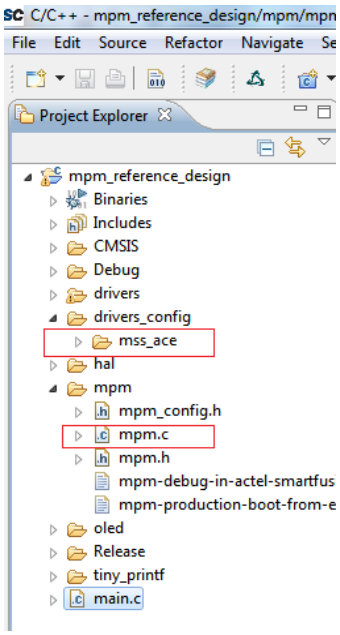


Figure 54 **• SoftConsole Project Explorer**

If you are adding the new channel of DPOL, do the following changes to mpm.c file.

1. To configure the port MPM_Channel_8_DMPM_DB_DPOL4_PG, add the configuration code to the function mpm_init_engine() as shown in Figure 55.

```
/* MPM Channel 3/4/5/8 = DPOL 1/2/3/4 with PGs on MSS GPIO 13/14/15/16 */
MSS_GPIO_config(MSS_GPIO_13, MSS_GPIO_INPUT_MODE | MSS_GPIO_IRQ_EDGE_BOTH);
MSS_GPIO_config(MSS_GPIO_14, MSS_GPIO_INPUT_MODE | MSS_GPIO_IRQ_EDGE_BOTH);
MSS_GPIO_config(MSS_GPIO_15, MSS_GPIO_INPUT_MODE | MSS_GPIO_IRQ_EDGE_BOTH);

MSS_GPIO_config(MSS_GPIO_16, MSS_GPIO_INPUT_MODE | MSS_GPIO_IRQ_EDGE_BOTH);

MSS_GPIO_enable_irq(MSS_GPIO_13);
MSS_GPIO_enable_irq(MSS_GPIO_14);
MSS_GPIO_enable_irq(MSS_GPIO_15);

MSS_GPIO_enable_irq(MSS_GPIO_16);
```

Figure 55 **• Configuring MSS_GPIO_16**

2. Add the following IRQHandler to mpm.c file

```
__attribute__((__interrupt__)) void GPIO16_IRQHandler(void)
{
   mpm_dpol_pg_changed(8, (MSS_GPIO_get_inputs() >> MSS_GPIO_16) & 1);
                MSS_GPIO_clear_irq(MSS_GPIO_16);
}
```

Now build the softconsole project in release mode. Use the *.HEX file from release folder (C:\Microsemi\SF_MPM_RefDesign_v4.0\design_files\SoftConsole_workspace\SF_MPM_RefDesign\mpm_reference_design\Release) in MSS eNVM client under MPM_Firmware as shown in Figure 56.

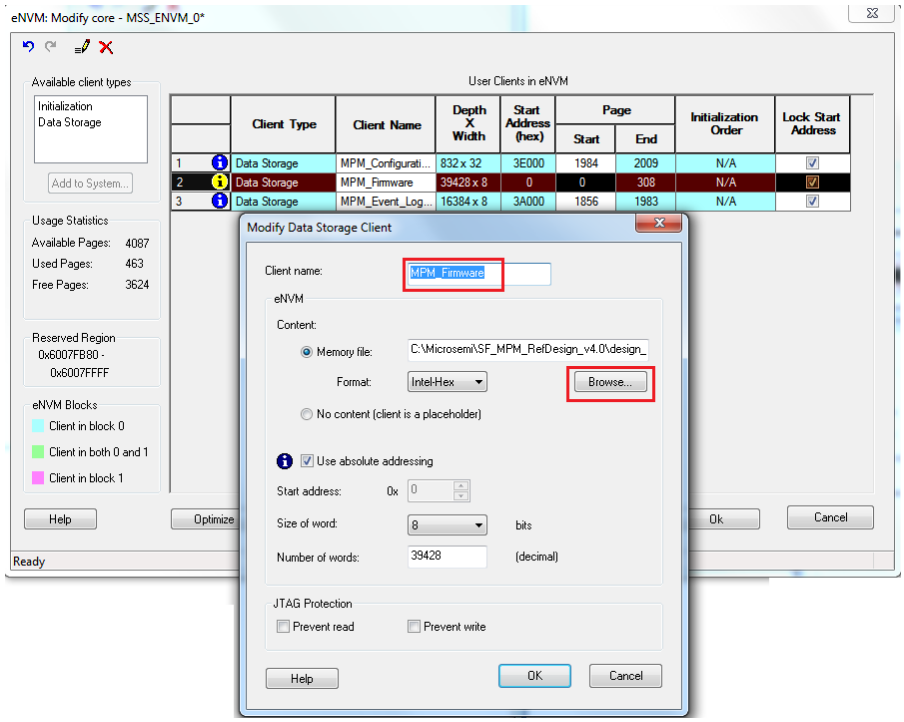


Figure 56 • **MSS eNVM MPM_Firmware Client**

After following above steps, generate the *.stp file from FlashPro and copy the same to C:\Microsemi\SF_MPM_RefDesign_v4.0\template.

Now from MPM GUI, choose the above generated *.stp file by selecting Choose STAPL Template option and load the default settings by selecting Load Values option from File submenu as shown in Figure 57 MPM GUI Menu.

Now you can set the all fields in GUI for your added channel(s) and click Write NVM & Fabric.

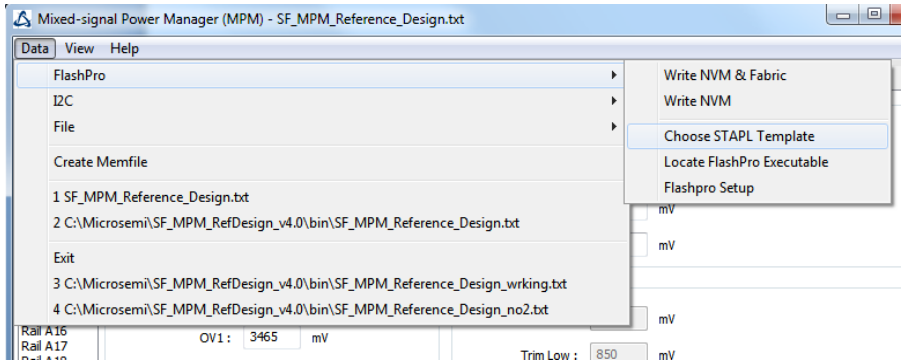


Figure 57 • **MPM GUI Menu**

Appendix A

Design Files

You can download the design files from the Microsemi SoC Products Group website: www.microsemi.com/soc/download/rsc/?f=A2F_AC385_DF.
The design file consists of Libero SoC projects and SoftConsole software projects. Refer to the ReadMe.txt file for directory structure, description, and software versions.
You can download the programming files (*.stp) in release mode from the Microsemi SoC Products Group website: www.microsemi.com/soc/download/rsc/?f=A2F_AC385_PF.

References

- MPM Product Brief: www.microsemi.com/soc/documents/MPM_SmartFusion_PB.pdf
- MPM Daughter Card: www.microsemi.com/soc/documents/MPM_DC_KIT_QS.pdf
- MPM Design User Guide: www.microsemi.com/soc/documents/SmartFusion_MPM_UG.pdf
- MPM White Paper: www.microsemi.com/soc/documents/MPM_WP.pdf

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo CA 92656 USA
Within the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996

Microsemi Corporation (NASDAQ: MSCC) offers a comprehensive portfolio of semiconductor solutions for: aerospace, defense and security; enterprise and communications; and industrial and alternative energy markets. Products include high-performance, high-reliability analog and RF devices, mixed signal and RF integrated circuits, customizable SoCs, FPGAs, and complete subsystems. Microsemi is headquartered in Aliso Viejo, Calif. Learn more at **www.microsemi.com**.

51900253 -0/08.12