SmartFusion cSoC: ACE Sequencing Control Using Fabric and MSS

Table of Contents

Introduction

The mixed signal blocks found in the SmartFusion® customizable system-on-chip (cSoC) devices are controlled and connected to the rest of the system via a dedicated processor called the analog compute engine (ACE). The ACE is built to handle the sampling, sequencing, and post-processing of the ADCs, DACs, and signal conditioning blocks (SCBs). For a larger deterministic synchronization between the sampling and custom fabric logic, the sample sequencing in the ACE is controlled through the microprocessor subsystem (MSS) or FPGA fabric logic.

This application note provides the two design examples which allow deterministic synchronization between the sampling essential steps:

- "Design Example 1: ACE Sequencer Using Fabric Logic" on page 3
- "Design Example 2: Simultaneous Sampling Using the MSS" on page 5

This application note also assumes that you are familiar with the SmartFusion analog block. Refer to the *SmartFusion Programmable Analog User's Guide* for more information.

Sample Sequencing in ACE

The ACE in the SmartFusion cSoC is made up of two major blocks:

- Sample sequencing engine (SSE)
- Post processing engine (PPE)

The SSE offers a flexible configuration of the analog front end (AFE) resources (analog inputs and monitors, comparators, ADCs, and DACs), as well as a variety of simple and sophisticated sample sequencing. The PPE is a self sufficient block that allows the data processing such as linear transformation, filtering, and thresholds comparisions, etc. The ACE also has an interface with the AFE and the MSS/FPGA fabric.

It interfaces with the MSS and fabric is through the APB3 bus as depicted in Figure 1.

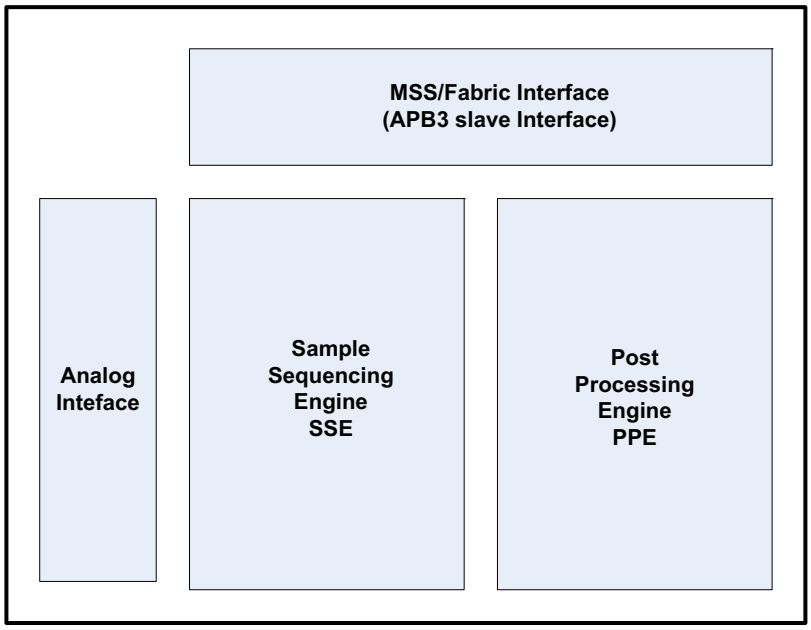

Figure 1 • **Overview of ACE Architecture and Interfaces**

Figure 2 provides an overview of the sample sequencing engine (SSE) block and its interfaces to PPE, APB3, ADCs, DACs, and ACB. The SSE block has the following:

- The sample sequencing instruction (SSI) unit multiplexes between APB3 and program instruction in SRAM, access to the analog fabric registers.
- Time division multiplexing finite state machine (TDM FSM): Creates the timeslots for each of the three ADCs, as well as a shared timeslot for APB3 or PPE accesses. The ADC timeslots are allocated during the ACE configuration in the SmartDesign.
- APB3/PPE interface: Arbitrates the SSE access between the APB3 and PPE.
- 24-bit phase accumulators: Front-end accumulators for the three SDDs.

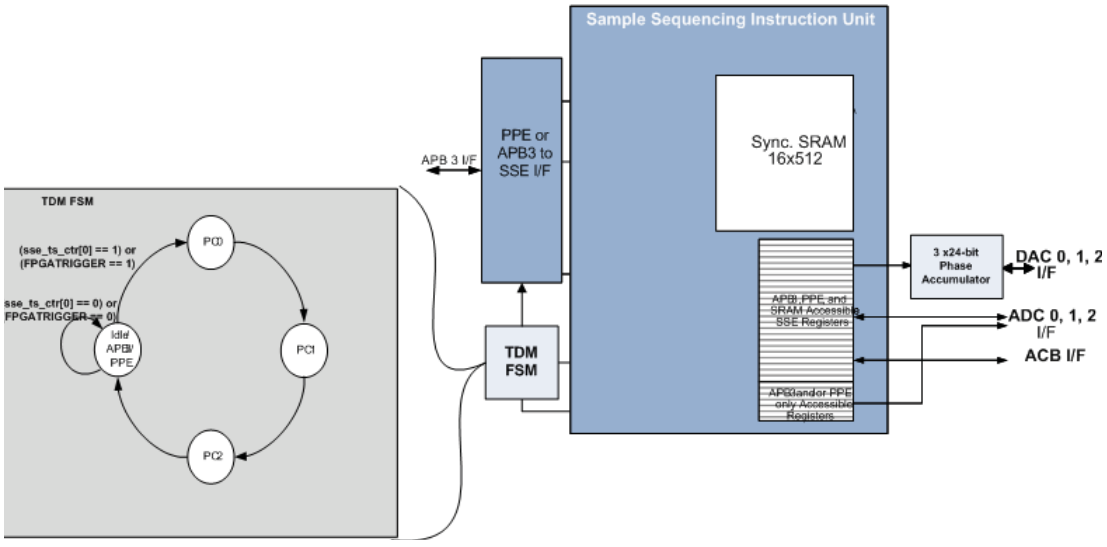

Figure 2 • **Overview of ACE Architecture and Interfaces**

The TDM FSM block (shown in Figure 2 on page 2) implements a mechanism to sequence through the ADCs. It allows the equal access by three separate program counters (PC0, PC1, and PC2) in the SSE and the APB3/PPE master. The SSE uses a separate program counters (PC) for sequencing and controlling each ADCs independently. When controlling the SSE from the APB3 master, the master needs to honor the wait-states generated by the *PREADY* signal during the normal operation of the SSE TDM timeslot counter.

Note: When the TDM timeslot counter is enabled for normal operation of the SSE block, there are always four timeslots, regardless of whether there are one, two, or three ADC instances in a specific device.

The TDM FSM is controlled by *FABACETRIG* signal or *SSE_TS_CTRL* register when bit 0 of

- *SSE_TS_CTRL* register or *FABACETRIG* is set to 1, it enables all 4 timeslots
- *SSE_TS_CTRL* or *FABACETRIG* set to 0, it only enables access to the APB3 master – this fixes the *PREADY* signal high to allow the zero-wait state access to the APB3 master

The SmartFusion cSoC devices also have the capability of simultaneous sampling on different ADCs. Simultaneous synchronized ADC conversion control register (ADC_SYNC_CONV) allows for a single program to issue the synchronized start instructions for all ADCs.

Design Example

This section describes the design examples. This application note shows two design examples of controlling the ACE sequencer using the fabric logic and MSS.

The first design example shows a master in fabric, uses the *FABACETRIG* to access the APB3 interface in the ACE, and then control the SSE sampling. The second design example shows the simultaneous sampling on multiple ADCs using the MSS.

Note: The timeslots do not have any capability neither to block each other nor to signal each other. So, you need to use the special consideration during the simultaneous sampling. For more details, refer to "Special Consideration During the Simultaneous Sampling" on page 6.

Design Example 1: ACE Sequencer Using Fabric Logic

This design example shows a master in fabric controlling the SSE timeslot. It uses the *FABACETRIG* (refer to "Appendix A: ACE Configuration and Connectivity" on page 8) signal to access the APB3 interface in ACE and then control the SSE sampling. Figure 3 illustrates the concept of using the *FABACETRIG* to control the SSE sampling.

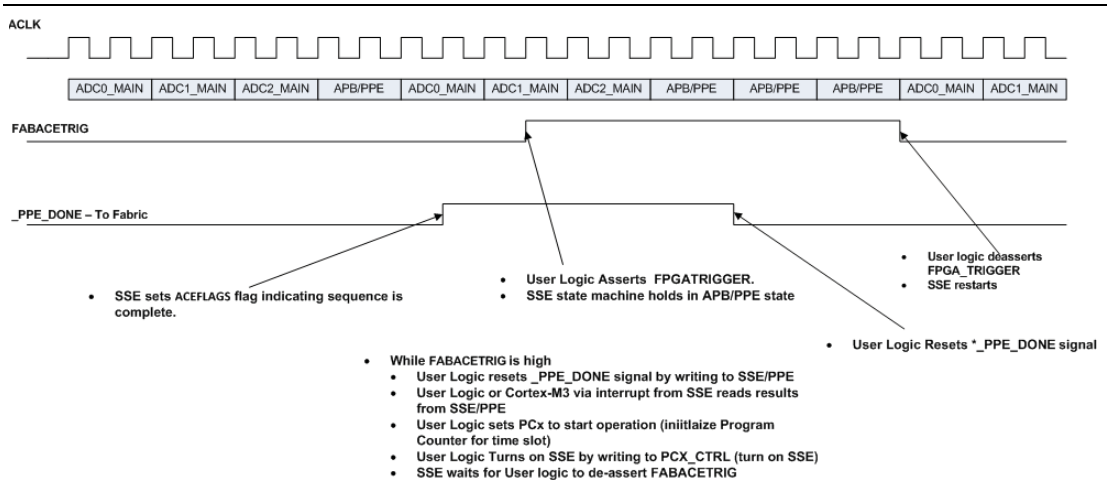

Figure 3 • **Timing Diagram showing FABACETRIG to Control SSE Sampling**

The fabric APB master is responsible for:

- Detecting that the ACE has reached the end of a sequence through the *ACEFLAGS* signal (ACEFLAGS are connected to *ACE_TRIGGER* in fabric).
- Assert the *FABACETRIG* signal to the ACE to halt the SSE time division multiplexing.
- Clear the **_PPE_DONE* notification flag by writing to the ACE PPE_FLAGS0_IRQ_CLR registers (refer to "Appendix B: Special ACE Register for Design Example" on page 12).
- Notifying the rest of the fabric logic to begin a transaction. This is simulated in this design with the *ENABLE_PWM* signal that enable a CorePWM IP.
- Re-enabling the ACE sequencer by writing to the program counter enables bit and writing the next program counter address (refer to "Appendix B: Special ACE Register for Design Example" on page 12). The program counter address is found in the *<project>\firmware\drivers_config\mss_ace\ace_config.c* file.
  The starting loop address for a given procedure is specified in the *ace_procedure_dest_t* table. For example, in the structure below, “2” is the starting loop address for this procedure.

```
{
      g_ace_sse_proc_0_name,                          /* const uint8_t * p_sz_proc_name */
        2,                                                  /* uint16_t sse_loop_pc */
        0,                                                /* uint16_t sse_load_offset */
         sizeof(g_ace_sse_proc_0_sequence) / sizeof(uint16_t), /* uint16_t
sse_ucode_length */
       g_ace_sse_proc_0_sequence,                      /* const uint16_t * sse_ucode */
        0                                                   /* uint8_t sse_pc_id */
},
```

The block diagram of the design example is shown in Figure 4.

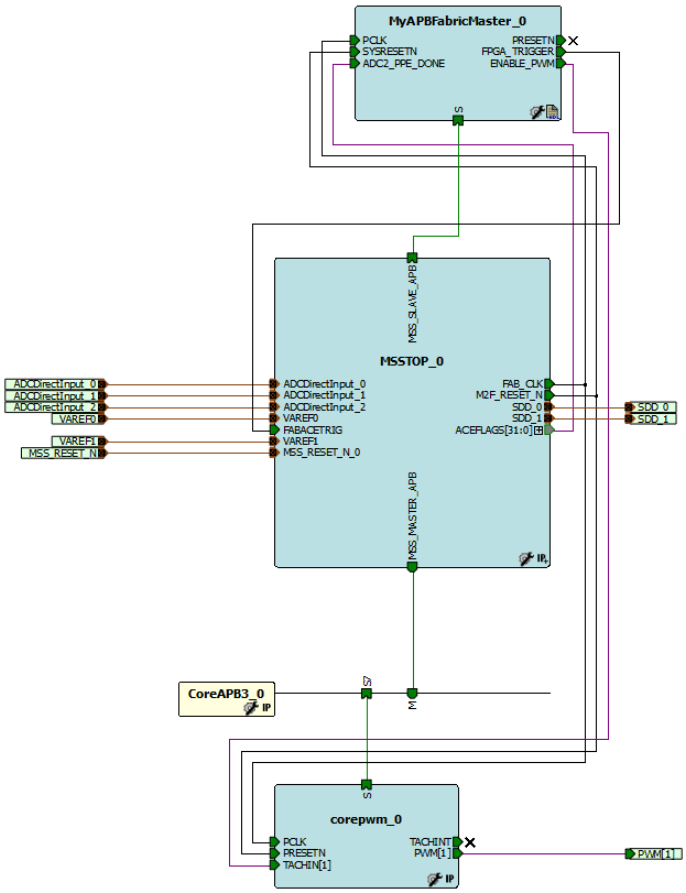

Figure 4 • **Top Level Block Diagram of ACE Sequencer Using Fabric Logic**

The design example is available for downloading at http://soc.microsemi.com/download/rsc/?f=A2F_AC366_DF .

The following waveform (Figure 5) shows the ACE and fabric APB3 master interacting.

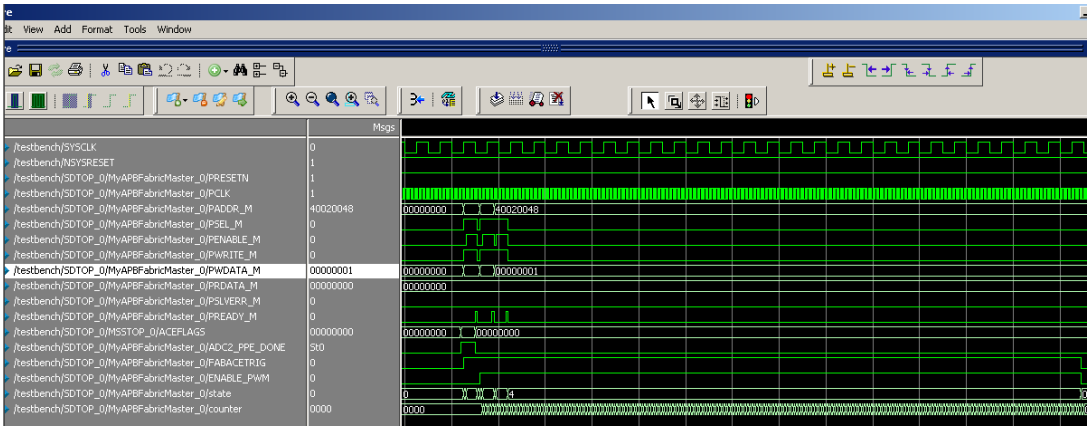

Figure 5 • **ACE and Fabric APB3 Master Simulation**

Design Example 2: Simultaneous Sampling Using the MSS

The second design example shows the simultaneous sampling on multiple ADCs using the MSS. Figure 6 shows the ACE configuration used in this design. "Appendix C: Configuration for Simultaneous Sample Procedure" on page 13 shows the option to enable simultaneous sampling in the ACE configurator.

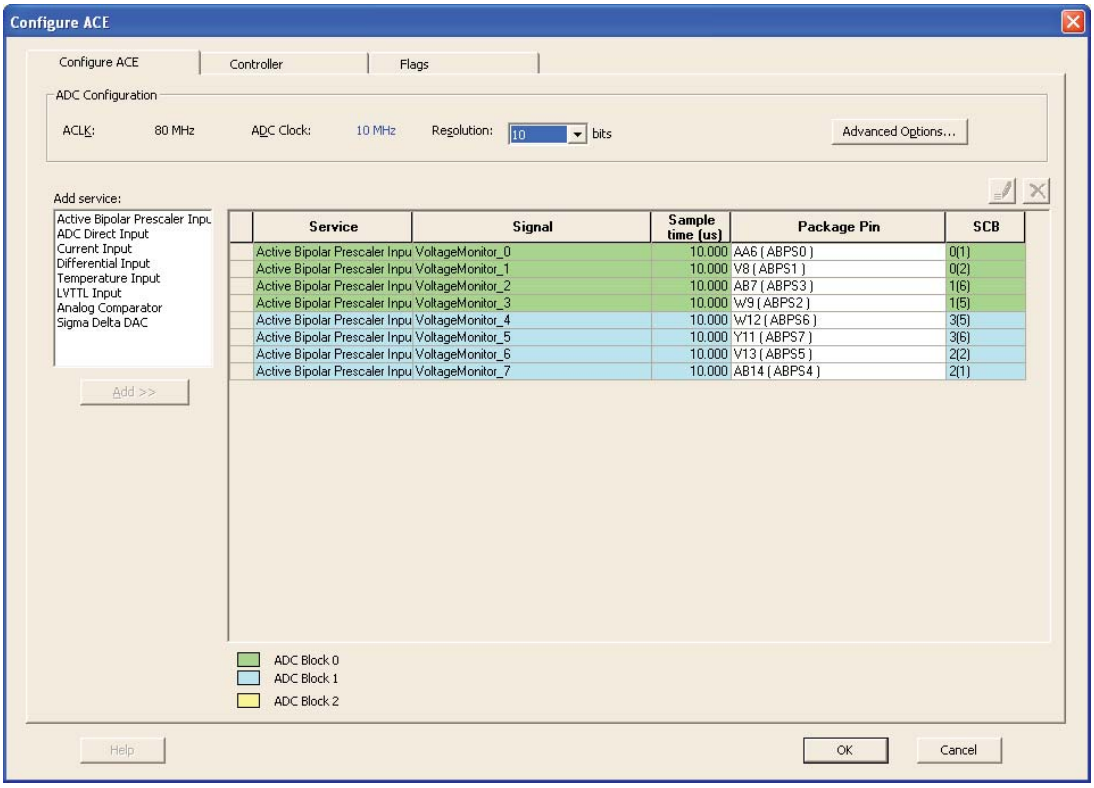

Figure 6 • **ACE Configuration for Simultaneous Sampling Design Example**

The block diagram of the design example is shown in Figure 7. The design example Libero® System-on-Chip (cSoC) software projects are available for downloading at http://soc.microsemi.com/download/rsc/?f=A2F_AC366_DF

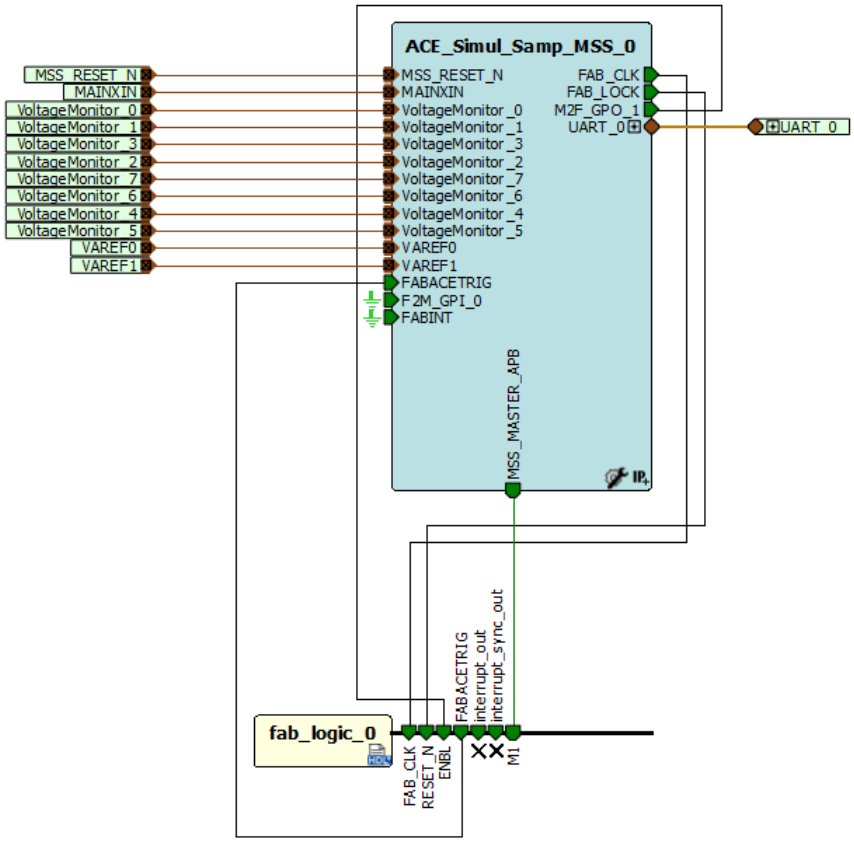

Figure 7 • **Top Level Block Diagram of Simultaneous Sampling Design Example**

Special Consideration During the Simultaneous Sampling

The ADC timeslots have the capability to access and control any of the ADCs. However, these timeslots do not have the capability to block or signal each other. Therefore, essentially they get two clock cycles each to execute the available microcode instruction, and then the SSE moves to the next timeslot. Due to this architectural advantage, you should take care of the following situation where the results might be wrong.

Assume that both the timeslot 0 and timeslot 1 are sampling the ADC0 channels. This means that the timeslot 0 executes its sample instructions and asserts the *ADCSTART* for the ADC0 on channel X. The ADC0 begins that process. Now the timeslot 1 is activated (after two clocks) and it also has an instruction that uses ADC0. There is no mechanism in the SSE to explicitly query whether the ADC is busy or not. Therefore, when that timeslot sees the sample microcode, it asserts another *ADCSTART* for the ADC0 on channel Y. Consequently, now you have the timeslot 0 waiting for a data valid from the ADC0 for channel X, and the timeslot 1 is waiting for a data valid from ADC0 for channel Y. When the ADC0 finally finishes and returns the data valid, both timeslots move onto their next respective microcode instructions. However, only one of them have the sample proper data. Only one piece of data is sent to the PPE, since it is the ADC data valid that is used to write into the *ADCFIFO* and only one ADC data valid signal allocated per ADC.

A technique is shown to solve this simultaneous sampling in "Design Example 2: Simultaneous Sampling Using the MSS" on page 5. Please refer to the technique below that solves the problem of simultaneous sampling on the same ADC.

Note: The two ADCs' procedure uses the simultaneous sample.

1. After power-on reset,
   - The MSS initializes the various peripherals.
   - The MSS loads the simultaneous sampling procedures into SSE to sample two channels per ADC (total four samples expected).
   - In the FPGA, *FABACETRIG* is set to 1.
   - The MSS sends an enable signal through a GPIO to FPGA (to convey that it can start the sampling sequence).
2. On seeing the GPIO set, FPGA starts 10 KHz counter.
3. On rising of 10 KHz, it brings *FABACETRIG* low after 1 µs.
4. The ACE starts simultaneous sampling and interrupts M3 when samples are available. The procedure disables the timeslot (*PCx_EN = 0*).
5. The M3 gets the data from the ACE into variables and loads both the sequential sampling procedures (*seq_adc00* and *seq_adc01*). Sequential sampling starts.
6. On completion, the M3 gets two interrupts for both the ADCs. The procedure disables both the timeslots (*PCx_EN = 0*). On serving both the interrupts (reading sample values), M3 does a write to fabric to signal end of sequence.
7. On detecting the write from M3, FPGA brings *FABACETRIG* high.
8. The M3 loads simultaneous sampling procedure (two ADCs).
9. Sequence repeats from step 3.

Conclusion

This application note provides the design examples of using the fabric logic or MSS to control the SSE sampling. This can be used to allow the deterministic synchronization.

Appendix A: ACE Configuration and Connectivity

Step1. MSS ACE Configuration for Design Example 1

1. Inside the ACE Configurator, click the **Advanced Options** dialog box. Select the **Expose FABACETRIG port** checkbox. This exposes the *FABACETRIG* port on the ACE block enabling the fabric logic to halt the ACE sequencing.

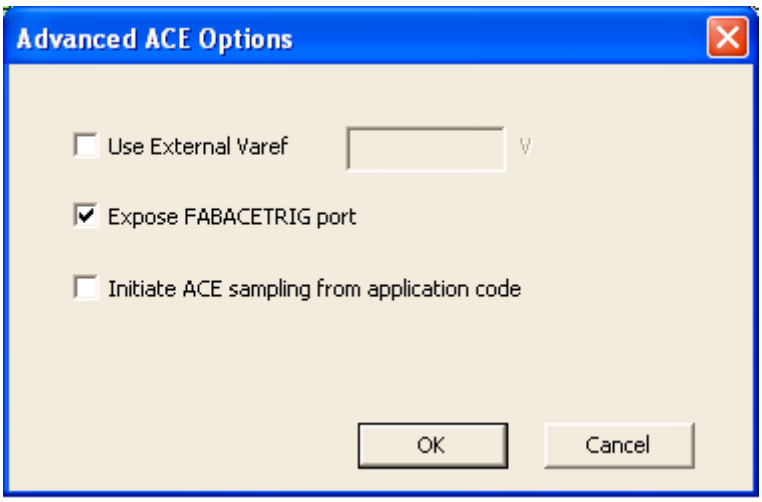

Figure 8 • **Expose FABACETRIG Port Checkbox**

2. Create the ACE design as normal. Configure the ACE services and threshold flags.

3. Sequence your analog services in the ACE configurator **Controller** tab.

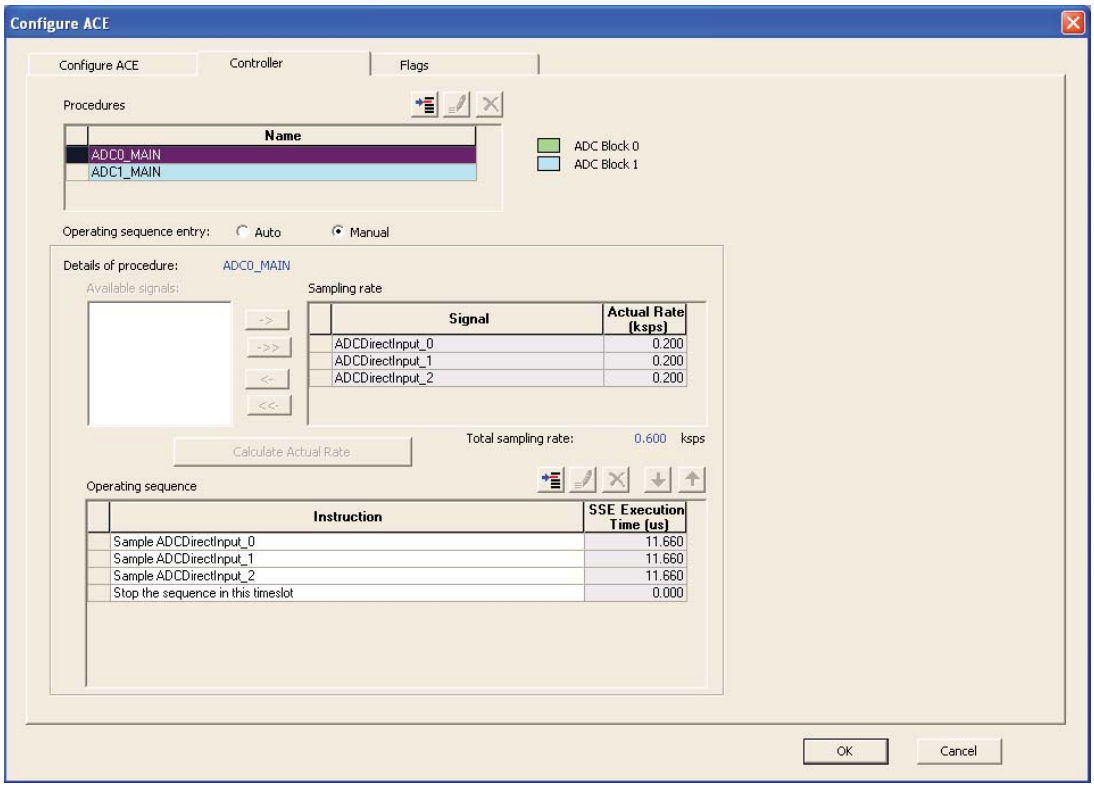

Figure 9 • **ACE Configurator Controller**

You can initiate a signal to the fabric when the end of this sequence is reached. You can accomplish this by reconfiguring the *ADCDirectInput_2* service to assert a signal on completing its post processing. Use this particular service, as it is the last one in this sequence.

4. Configure the *ADCDirectInput_2* service and select the **Assert flag when post processing completed** checkbox.

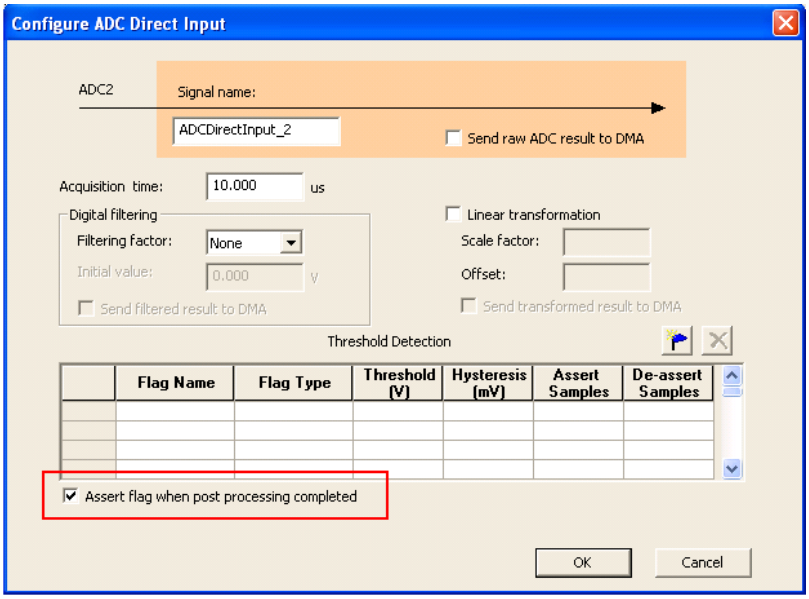

Figure 10 • **Configure the ADCDirectInput_2 Service**

Selecting this checkbox creates a flag signal named *<Signal name>:PPE_DONE* automatically.

5. Go into the **Flags** tab and assign it to the bit position of your choice. In the example, you have assigned the new *PPE_DONE* indication to bit 15.

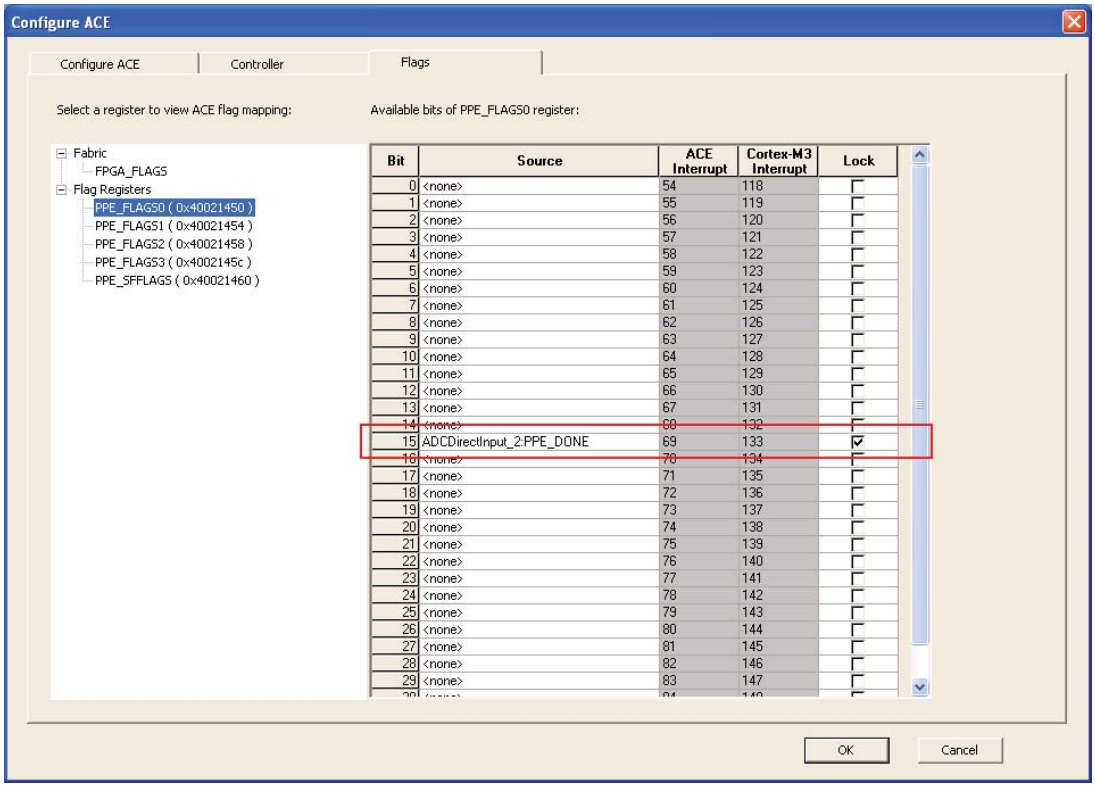

Figure 11 • **Assigning ADCDirectInput_2_PPE_DONE to Bit 15**

6. Click *FPGA_FLAGS* in the left-hand treeview and select the **Expose these signals as ACEFLAGs** checkbox.

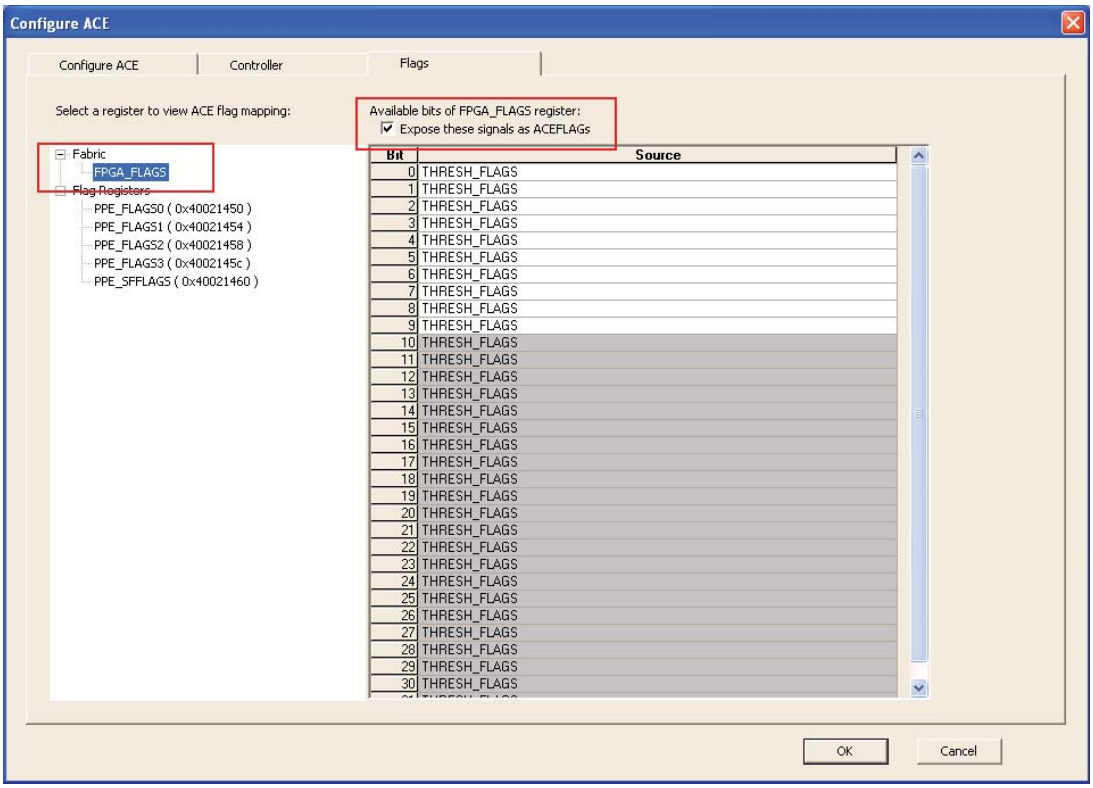

Figure 12 • **Exposing ACEFLAGs**

7. Click **OK**, the ACE instance in the MSS configurator window is displayed, as shown in Figure 13.

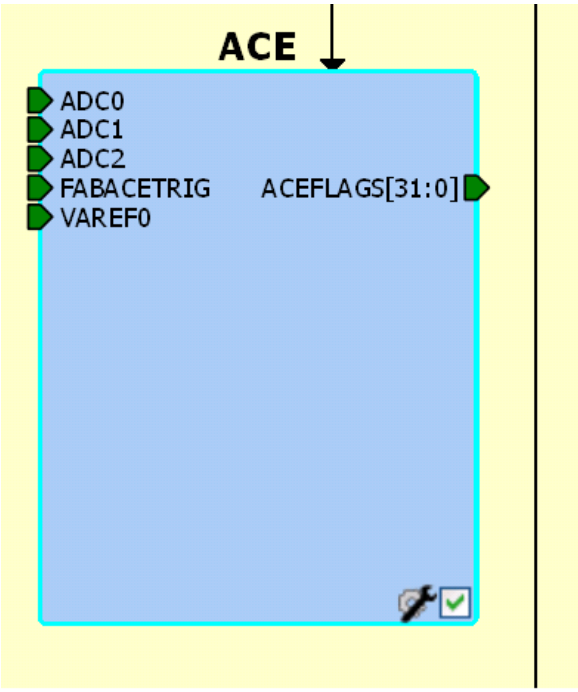

Figure 13 • **ACE Instance in the MSS Configurator**

Appendix B: Special ACE Register for Design Example

Table 1 • **Special ACE Register for Design Example**

ADDR	Register Name	Description
0x4002122c	PPE_FLAGS0_IRQ_CLR	These *write-only* bits are used to clear corresponding bits in the *PPE_FLAGS0_IRQ* register. Writing a 1 to any of the bits clears the corresponding bits in the *PPE_FLAGS0_IRQ* register, while writing a 0 to any of the bits does not have an effect. In the event that writing a 1 to these clear bits coincides with a set event in the *PPE_FLAGS0_IRQ* register, the set event shall have higher priority.
0x40020040	PC0_LO	Program Counter 0 points to address the next sample sequence instruction for ADC0. A write to this register causes the timeslot 0 to unconditionally jump to the SSE RAM address PC[7:0], which includes RAM addresses from 0 to 255 inclusive.
0x40020044	PC0_HI	Program Counter 0 points to address the next sample sequence instruction for ADC0. A write to this register causes timeslot 0 to unconditionally jump to the SSE RAM address 256+PC[7:0], which includes RAM addresses from 256 to 511 inclusive.
0x40020048	PC0_CTRL	Only bit 0 is used. When set to 1, Program Counter 0 is enabled and when set to 0, Program Counter 0 is disabled (stop).

Note: Table 1 shows the register description for the Program Counter 0. The registers are duplicated for Program Counter 1 and Program Counter 2.

Appendix C: Configuration for Simultaneous Sample Procedure

1. In the Configurator ACE window, click the **Controller** tab.
2. From the **Insert Operating Sequence Slot** drop-down list, select SIMULTANEOUS SAMPLE. The Configure 'SIMULTANEOUS SAMPLE' window is displayed.

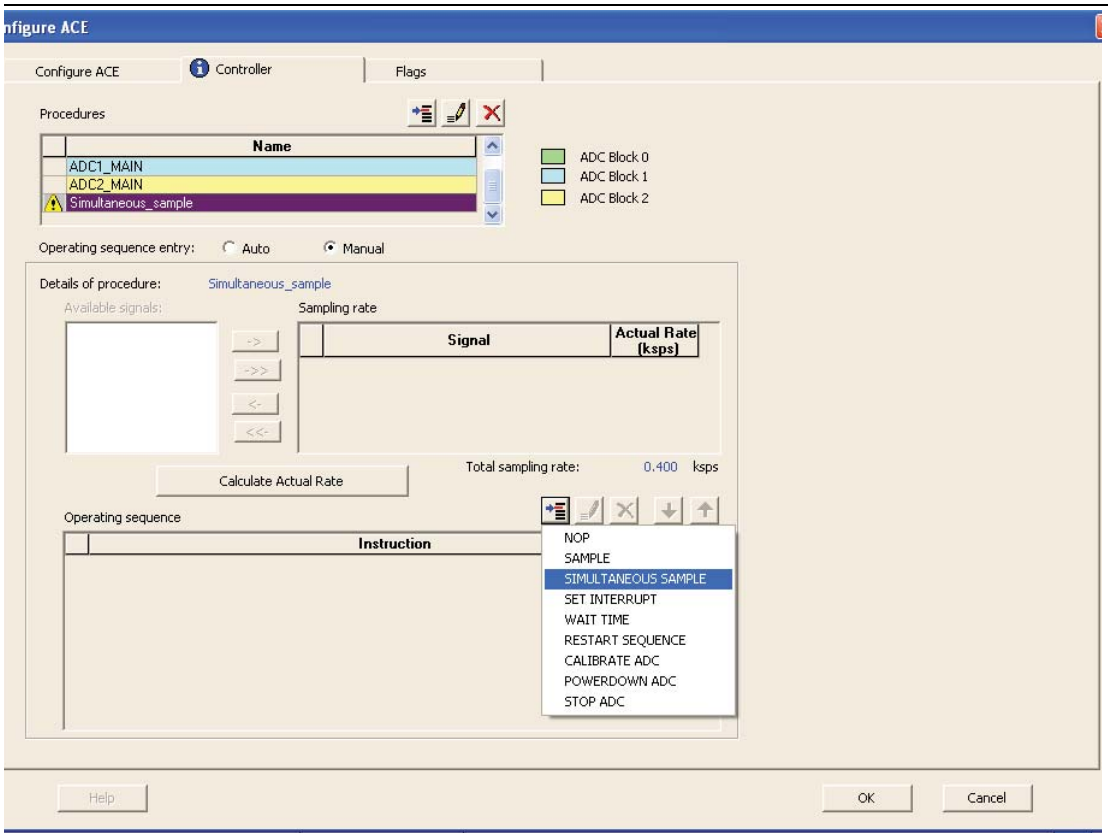

Figure 14 • **Adding Simultaneous Sample Procedure**

3. Select the services you want to sample simultaneously and click **OK**.

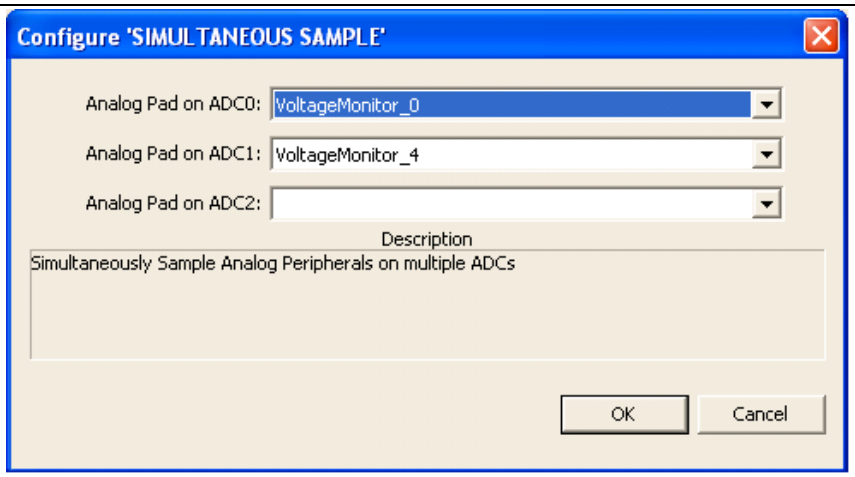

Figure 15 • **Selecting Services for Simultaneous Sample**

Appendix D: ACE Configuration and Procedure for Design Example 2

The ACE controller procedures are:

ADC0_MAIN:

- Stop the sequence in this time slot

ADC1_MAIN:

- Stop the sequence in this time slot

ADC2_MAIN:

- Stop the sequence in this time slot

seq_adc00:

- *Sample VoltageMonitor_2*
- *Sample VoltageMonitor_3*
- Assert Interrupt on GP1
- Stop the sequence in this time slot

seq_adc10:

- *Sample VoltageMonitor_6*
- *Sample VoltageMonitor_7*
- Assert Interrupt on GP0
- Stop the sequence in this time slot

two ADCs:

- *Simultaneously Sample VoltageMonitor_0 VoltageMonitor_4*
- Assert Interrupt on GP0
- *Simultaneously Sample VoltageMonitor_1 VoltageMonitor_5*
- Stop the sequence in this time slot

Appendix E: Design Files

You can download the design files from the Microsemi SoC Products Group website: http://soc.microsemi.com/download/rsc/?f=A2F_AC366_DF .

List of Changes

The following table lists critical changes that were made in each revision of the document.

Revision*	Changes	Page
Revision 1 (January 2012)	Updated Figure 4 (SAR 35795).	4
	Modified the section "Design Example" (SAR 35795).	3
	Updated Figure 7 (SAR 35795).	6
	Modified the section "Appendix E: Design Files" (SAR 35795).	14
Revision 2 (May 2019)	Modified the sections: Design Example 1: ACE Sequencer Using Fabric Logic Design Example 2: Simultaneous Sampling Using the MSS (SAR 51242).	5 and 6

*Note: *The revision number is located in the part number after the hyphen. The part number is displayed at the bottom of the last page of the document. The digits following the slash indicate the month and year of publication.*

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

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.

51900234-2/05.19