1300 Henley Court
Pullman, WA 99163
509.334.6306
www.digilentinc.com

Zmod AWG 1411 Controller IP Core User Guide

Revised October 12, 2021; Author Tudor Gherman

1 Introduction

This user guide describes the Digilent **Zmod AWG 1411 Controller** Intellectual Property (IP). This IP interfaces directly with the Zmod AWG 1411 initializing the hardware and encoding the samples received over the Input Data Stream interface (AXI Stream) on a single double data rate (DDR) channel as required by the AD9717 digital to analog converter (DAC) featured by the Zmod AWG 1411. The Zmod AWG 1411 Low Level Controller is intended to be used as a stand-alone IP in projects that do not require processor interaction (standalone mode) or it can be used in conjunction with higher level IPs that may provide connectivity with the processing system.

IP quick facts	
Supported device families	Zynq®-7000, 7 series
Supported user interfaces	Custom, AXI Stream
Provided with core	
Design files	VHDL
Simulation model	**Yes** for AD9717 SPI interface; **No** for he IP itself (it can be simulated using the design fies)
Constraints file	XDC
Software driver	N/A
Tested design flows	
Design entry	Vivado™ Design Suite 2019.1
Synthesis	Vivado Synthesis 2019.1

2 Features

- Initializes the hardware on the Zmod AWG 1411.
- Formats the data received on the Input Data Stream interface (AXI Stream slave) according to the Zmod AWG 1411 requirements.
- Provides the possibility of overwriting the initial DAC configuration by providing an optional upper level interface that allows indirect access to the DAC's SPI interface.
- Performs offset and gain calibration based on coefficients specified by the user/upper level IPs.

3 Performance

This IP is designed to allow a variable sample rate for the Zmod AWG 1411, the maximum sample rate being limited by the capabilities of the AD9717 DAC (125MSPS) [1] and those of the FPGA [2]. The two clock signals required by the AD9717 need to be configured in the top level design and are only forwarded by this IP on the Zmod's external pins (one clock signal used to qualify the data at the input of the AD9717 device and one used as sample clock).

The data path of this IP is responsible with decoding the Input Data Stream interface, applying the calibration coefficients to the input samples and encoding the outputs of the calibration blocks in the format required by the AD9717 DAC. This process allows achieving the accuracy specified for the Zmod AWG 1411 (Zmod AWG Reference Manual).

4 Overview

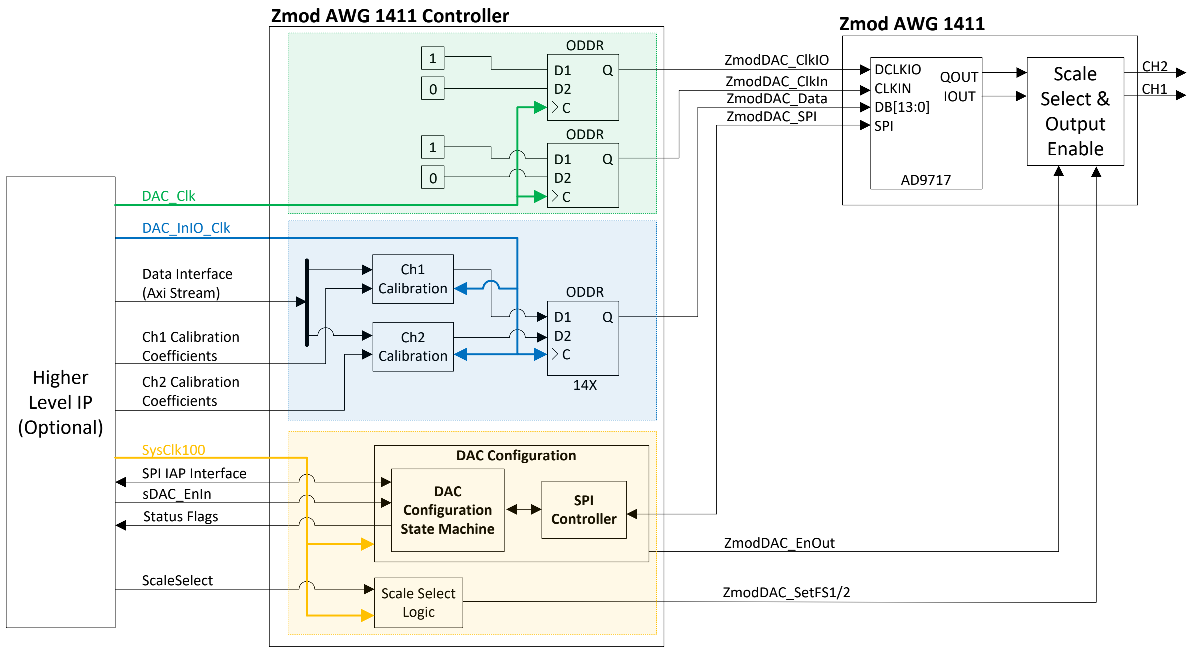


Figure 1. Zmod AWG 1411 Controller block diagram.

The structure of the IP is presented in Figure 1. The main functionalities are divided as DAC input clock generation, data formatting and calibration (data path) and Zmod AWG 1411 configuration (two items related with configuration functionalities will be detailed separately: the configuration state machine and the SPI controller).

4.1 Clocking

This IP requires 3 input clocks which divide it in 3 distinct clock domains as shown in Figure 1:

1. The system clock domain (*SysClk100*) clocks the DAC configuration module including the SPI controller. The frequency of this clock is expected to be 100MHz. All signals synchronous with *SysClk100* have the prefix "s".
2. The sample clock (*DAC_InIO_Clk*) clocks the data path related logic including the calibration module. It is important to note that, although this IP processes input and output samples in this clock domain, the DAC will be forwarded a phase shifted version of this clock so that timing on the DAC's data interface can be achieved. All signals synchronous with *DAC_InIO_Clk* have the prefix "c".
3. The DAC clock (*DAC_Clk*) should have the same frequency but a 90° phase shift with respect to *DAC_InIO_Clk*. *DAC_Clk* is used to generate the DCLKIO (labeled *ZmodDAC_ClkIO*) and DCLKIN (labeled *ZmodDAC_ClkIn*) signals for the AD9717 DAC. All signals synchronous with *DAC_Clk* have the prefix "d". Note that the external data port of the AD9717 data interface is named *dZmodDAC_Data*, even though *dZmodDAC_Data* is generated in the *DAC_InIO_Clk* domain. This choice of naming was made to emphasize that, at the DAC's input, the data signals are sampled on the *ZmodDAC_ClkIO* clock.

This IP does not constrain the clocks it requires as inputs. The exception is out-of-context synthesis, where the ConstrZmodDAC1411_ooc.xdc contains the necessary constraints. However, this file is not used in top-level synthesis. Therefore, clocks need to be constrained in the top-level design either manually or by relying on the auto-derived constraints, if using clock modifying blocks. For more information see [3].

4.2 Reset

This IP has a single asynchronous reset input with negative polarity (*aRst_n*) which resets the logic in all 3 clock domains. To assure that recovery/removal time of sequential logic is respected, the reset input is distributed to the different clock domains throughout the IP by ResetBridge modules. The ResetBridge modules are responsible with converting the asynchronous reset input in reset signals with synchronous de-assertion (RSD) for each clock domain.

The input reset (aRst_n) must be asserted for at least 2*$T_{slowest}$, where $T_{slowest}$ represents the period of the slowest clock input of the IP.

4.3 Data Path

The data path consists of 3 stages: Input Data Stream interface decode, calibration and output data encoding. Each of these stages is described separately below.

4.3.1 Input Data Stream Interface Decode

Data is sent to the Zmod AWG 1411 Low Level Controller over an AXI Stream interface (named *InputDataStram*), the Zmod AWG 1411 Low Level Controller being a slave on this interface. Since the samples received are processed in real time, the TREADY signal of the Input Data Stream interface is always asserted. The TDATA signal is encoded as follows:

31 18	17 16	15 2	1 0
Channel 1 DAC code (14 bit)	00	Channel 2 DAC code (14 bit)	00

Channel 1 and Channel 2 input codes are further passed to the appropriate calibration modules. If the TVALID signal is de-asserted, the inputs of the calibration modules hold the value of the last valid code.

4.3.2 Calibration (GainOffsetCalib.vhd)

To generate a particular voltage value at the output of the Zmod AWG 1411, the user application sends a signed 14 bit integer to the DAC. This value needs to consider the gain and offset errors in order to obtain the expected accuracy for the output, thus a calibration process needs to be applied. The DAC code is computed as:

$$N_{calib} = \frac{(V_{out} - CA)}{(1 + CG) * Range} * 2^{13} \tag{1}$$

were:

- N_{calib} = the 14 bit, 2's complement integer sent to the DAC that accounts for gain and offset errors.

- V_{out}= the desired value of the output voltage.
- *CA* = factory calibration raw additive constant (for the appropriate channel and gain).
- *CG* = factory calibration raw gain constant (for the appropriate channel and gain).
- *Range* = the theoretical range of the DAC's channel output stage
  - 1.33 (for low gain: ±1.25 V) or
  - 5.32 (for high gain: ±5 V)

The raw calibration additive and gain constants are computed and saved in the Zmod's calibration memory at manufacturing time, as described in the Zmod AWG Reference Manual. Using a fractional representation for the DAC code *N*, Eq (1) can be rearranged as:

$$V_{out} = N_{calib_{fract}} * (1 + CG) * Range + CA \tag{2}$$

The purpose of the calibration block is to provide an ideal representation of the DAC to the upper levels. By adding the calibration block, the user may send a code as for an ideal DAC having the output range equal to $Range_{ideal}$ (1.25 V for the low gain option and 5 V for the high gain option) and the calibration block will apply the offset and gain coefficients so that:

$$V_{out} = N_{raw_{fract}} * Range_{ideal} \tag{3}$$

From Eq (2) and Eq (3) one can obtain:

$$N_{calib_{fract}} * (1 + CG) * Range + CA = N_{raw_{fract}} * Range_{ideal} =>$$

$$N_{calib_{fract}} = N_{raw_{fract}} * CoefMult - CoefAdd \tag{4}$$

where:

$$CoefMult = \frac{Range_{ideal}}{(1 + CG) * Range} \tag{5}$$

$$CoefAdd = \frac{CA}{(1 + CG) * Range} \tag{6}$$

CoefMult and *CoefAdd* are fractional numbers computed for each channel and each gain option. *CoefMult* and *CoefAdd* are further scaled so that they can be passed as integer numbers to the IP:

$$CoefMult_{scaled} = CoefMult * 2^{16} \tag{7}$$

$$CoefAdd_{scaled} = CoefAdd * 2^{17} \tag{8}$$

The numerical representation of the operands of the calibration process is illustrated in Figure 2. The scaled version of *CoefMult* and *CoefAdd* are named "processed calibration coefficients" and these are the constants required by the GainOffsetCalib module. One can note (Figure 2) that the GainOffsetCalib converts the processed calibration coefficients back to their fractional representation.

The multiplicative and additive coefficients are passed by the upper layer IP Core if the **Zmod AWG 1411 Controller** is used in a processor system (the external calibration interface is enabled) or they can be passed as IP core parameters otherwise. The user can obtain the calibration coefficients by booting the Eclypse board with the Linux image provided at https://github.com/Digilent/Eclypse-Z7/releases and run the "decutil enum" command in the command line. The Zmod AWG 1411 has to be plugged in one of the Eclypse's board SYZYGY ports. The Digilent Eclypse Utility (decutil) provides a command line interface for discovering information about the features and configuration of an Eclypse platform board. Decutil reports the factory raw calibration parameters and processed calibration parameters as shown in Figure 3.

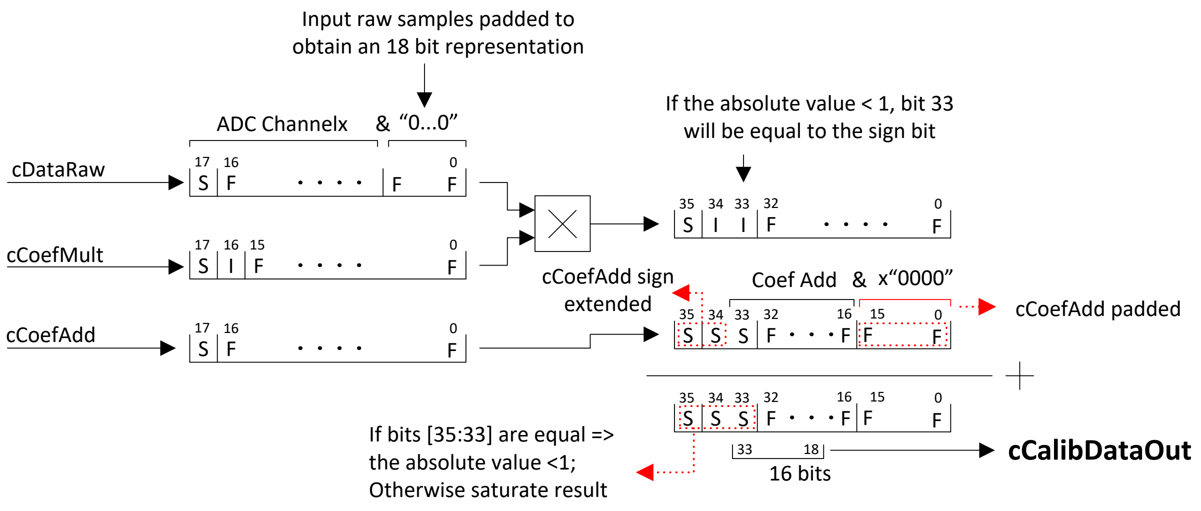
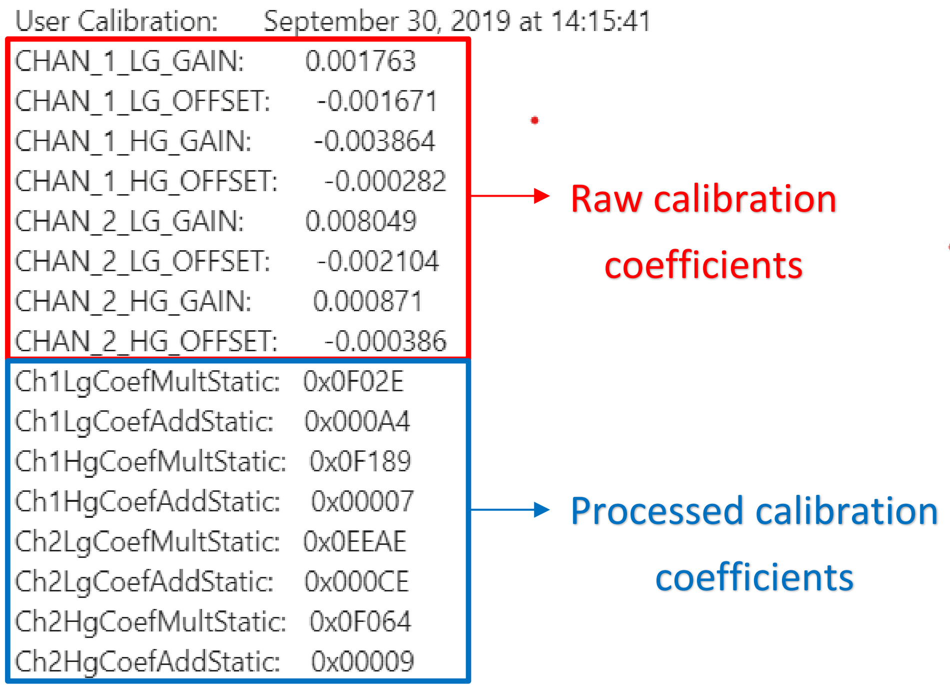

Input raw samples padded to obtain an 18 bit representation
ADC Channelx & "0...0"
cDataRaw
cCoefMult
cCoefAdd
If the absolute value < 1, bit 33 will be equal to the sign bit
cCoefAdd sign extended
Coef Add & x"0000"
cCoefAdd padded
If bits [35:33] are equal => the absolute value <1; Otherwise saturate result
16 bits
cCalibDataOut

Figure 2: Calibration process


Raw calibration coefficients
Processed calibration coefficients

```
User Calibration:      September 30, 2019 at 14:15:41
CHAN_1_LG_GAIN:        0.001763
CHAN_1_LG_OFFSET:      -0.001671
CHAN_1_HG_GAIN:        -0.003864
CHAN_1_HG_OFFSET:      -0.000282
CHAN_2_LG_GAIN:        0.008049
CHAN_2_LG_OFFSET:      -0.002104
CHAN_2_HG_GAIN:        0.000871
CHAN_2_HG_OFFSET:      -0.000386
Ch1LgCoefMultStatic:   0x0F02E
Ch1LgCoefAddStatic:    0x000A4
Ch1HgCoefMultStatic:   0x0F189
Ch1HgCoefAddStatic:    0x00007
Ch2LgCoefMultStatic:   0x0EEAE
Ch2LgCoefAddStatic:    0x000CE
Ch2HgCoefMultStatic:   0x0F064
Ch2HgCoefAddStatic:    0x00009
```

Figure 3: Obtaining calibration prameters from decutil

The parameters and ports of the calibration module are listed below:

Table 1. Calibration module parameter descriptions.

Signal Name	Description
kWidth	ADC/DAC resolution.

kExtCalibEn	Enables the external calibration interface. Set to “true” when the IP core is expected to be interfaced with the processing system through a high level IP. Set to “false” when the core operates in stand alone mode.
kInvert	When asserted, *kInvert* determines the sign inversion of the data samples received. Used to compensate the physical inversion of some of the channels on the PCB at the ADC/DAC input/output on the Zmod.
kLgMultCoefStatic[17:0]	Low gain multiplicative calibration coefficient. If the *kExtCalibEn* parameter is set to “false”, the calibration block will expect the value of the multiplicative coefficient to be passed by this parameter. If the value of *kExtCalibEn* parameter is “true”, this parameter is ignored, and the high level IP is expected to update the corresponding external port.
kLgAddCoefStatic[17:0]	Low gain additive calibration coefficient. If the *kExtCalibEn* parameter is set to “false”, the calibration block will expect the value of the additive coefficient to be passed by this parameter. If the value of *kExtCalibEn* parameter is “true”, this parameter is ignored, and the high level IP is expected to update the corresponding external port.
kHgMultCoefStatic[17:0]	High gain multiplicative calibration coefficient. If the *kExtCalibEn* parameter is set to “false”, the calibration block will expect the value of the multiplicative coefficient to be passed by this parameter. If the value of *kExtCalibEn* parameter is “true”, this parameter is ignored, and the high level IP is expected to update the corresponding external port.
kHgAddCoefStatic[17:0]	High gain additive calibration coefficient. If the *kExtCalibEn* parameter is set to “false”, the calibration block will expect the value of the additive coefficient to be passed by this parameter. If the value of *kExtCalibEn* parameter is “true”, this parameter is ignored, and the high level IP is expected to update the corresponding external port.

Table 2. Calibration module port description

Signal Name	Interface	Signal Type	Init State	Description
SamplingClk	-	I	N/A	Calibration module clock input.
acRst_n	-	I	N/A	Active low reset (can be asynchronously asserted but synchronously de-asserted in the sampling clock domain).
cTestMode	-	I	N/A	*cTestMode* is used to bypass the calibration block. When asserted, raw samples are sent to the DAC.
cExtLgMultCoef[17:0]	-	I	N/A	Low gain multiplicative coefficient external port. This port is enabled by setting the *ExtCalibEn* parameter to “true”.
cExtLgAddCoef[17:0]	-	I	N/A	Low gain additive coefficient external port. This port is enabled by setting the *ExtCalibEn* parameter to “true”.
cExtHgMultCoef[17:0]	-	I	N/A	High gain multiplicative coefficient external port. This port is enabled by setting the *ExtCalibEn* parameter to “true”.

cExtHgAddCoef[17:0]	-	I	N/A	High gain additive coefficient external port. This port is enabled by setting the *ExtCalibEn* parameter to "true".
cGainState	-	I	N/A	Corresponding channel selected gain option. The ADC /DAC calibration module needs to be aware of the gain setting to apply the appropriate calibration coefficients, which are different between the low gain option and the high gain option. The relay gain state is provided by the Relay Configuration module. • 1 = High Gain. • 0 = Low Gain.
cDataRaw[Width-1 : 0]	-	I	N/A	Input raw samples provided by the Data Path module.
cDataInValid	-	I	N/A	Data valid indicator provided by the Data Path module.
cCalibDataOut[15 : 0]	-	O	N/A	Calibrated data output. Regardless of the ADC/DAC resolution, the output of the calibration process is represented on 16 bits.
cDataCalibValid	-	O	N/A	Delayed version of *cDataInValid*. The same latency obtained for the calibration process (3 sampling clock cycles) is added to the input valid indicator to obtain *cDataCalibValid.*

4.3.3 Output Data Encoding

Each of the output bits of the calibration stage is further connected to an ODDR primitive which formats the output data on a DDR interface according to the AD9717 requirements. The output of the Channel 1's calibration module is connected to the D1 input of the ODDR primitives (*ODDR_D1*), while the output of Channel 2's calibration module is connected to the D2 input (*ODDR_D2*) [4]. The output data encoding process is illustrated in Figure 4.

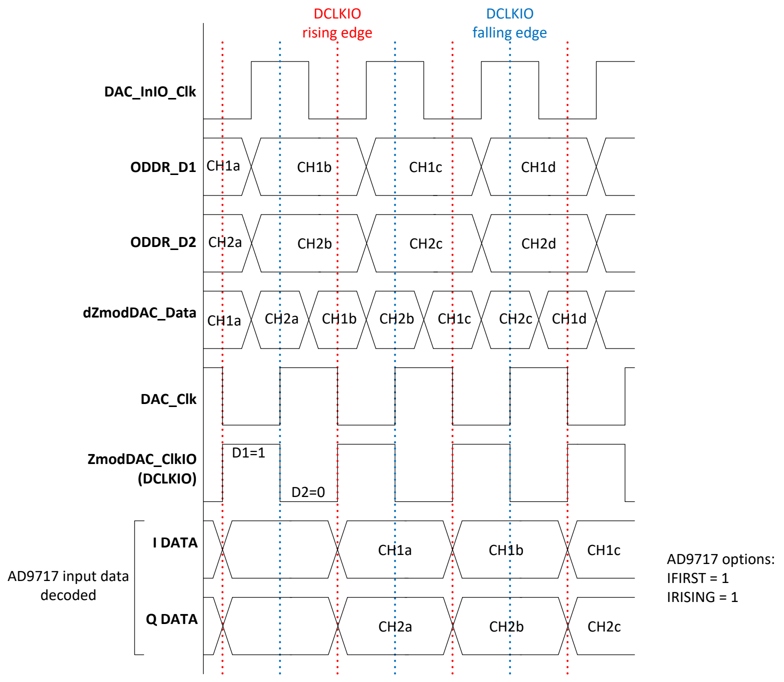


Figure 4: DAC data formatting

4.4 Gain Select

The IP core provides the choice of selecting the Zmod AWG 1411 gain options statically or dynamically. For static configuration, the *kExtScaleConfigEn* needs to be set as "*false*" and the gain configuration parameters (*kCh1ScaleStatic*, *kCh2ScaleStatic*) need to be configured in the IP core GUI. For dynamic gain control, the *kExtScaleConfigEn* needs to be set as "*true*" and the external gain control ports (*sExtCh1Scale*, *sExtCh2Scale*) will become available.

The gain configuration signals also need to be passed to the calibration module (GainOffsetCalib.vhd) so that the correct calibration coefficients can be applied. For this, the configuration signals need to cross clock domains, from the *SysClk100* domain to the *DAC_InIO_Clk* domain where the calibration module operates. Due to the latency associated with the clock domain crossing, the register stages in the data path and the register stages in the calibration module the output data (*dZmodDAC_Data*) may have incorrect values for *kSysClkPeriod* + 6**kDacClkPeriod*, where *kSysClkPeriod* represents the period of the SysClk100 clock signal and *kDacClkPeriod* represents the period of *DAC_InIO_Clk* clock signal.

4.5 DAC Configuration (ConfigDAC.vhd)

The DAC Configuration block sends a predefined list of SPI commands to the Zmods DAC 1411, performing the DAC initialization, and manages the SPI Indirect Access Port (IAP). The ports and parameters of the DAC configuration block, which is implemented as a distinct VHDL module are listed below:

Table 3. DAC Configuration parameter description

Parameter Name	Description
kDataWidth	The number of data bits for the data phase of the transaction: only 8 data bits currently supported, parameter provided for future development.
kCommandWidth	The number of bits of the command phase of the SPI transaction.

Table 4. DAC Configuration port description

Signal Name	Interface	Signal Type	Init State	Description
SysClk100	-	I	N/A	100MHz input clock signal.
asRst_n	-	I	N/A	Active low reset (can be asynchronously asserted but synchronously de-asserted).
sZmodDAC_SCLK	SPI	O	N/A	Output SPI clock. Signal managed by the SPI controller. More details in section 4.6.
sZmodDAC_SDIO	SPI	IO	N/A	2 Wire SPI interface SDIO signal. Signal managed by the SPI controller. More details in section 4.6.

sZmodDAC_CS	SPI	O	N/A	2 Wire SPI interface CS signal. Signal managed by the SPI controller. More details in section 4.6.
sInitDoneDAC	-	O	N/A	Flag indicating when the Zmod's DAC 1411 initialization is complete.
sConfigError	-	O	N/A	This flag is asserted if the DAC initialization fails. An IP reset is required to clear this flag.
sCmdTxAxisTdata[31:0]	SPI IAP	I	N/A	IAP command TX interface (AXI Stream) data port.
sCmdTxAxisTvalid	SPI IAP	I	N/A	IAP command TX interface (AXI Stream) valid signal.
sCmdTxAxisTready	SPI IAP	O	N/A	IAP command TX interface (AXI Stream) ready signal.
sCmdRxAxisTdata[31:0]	SPI IAP	O	N/A	IAP command RX interface (AXI Stream) data port.
sCmdRxAxisTvalid	SPI IAP	O	N/A	IAP command RX interface (AXI Stream) valid signal.
sCmdRxAxisTready	SPI IAP	I	N/A	IAP command RX interface (AXI Stream) ready signal.

Once the initialization is complete, the state machine of the DAC configuration block enters the idle state where it monitors if there is any valid data on the upper level SPI command interface named the SPI Indirect Access port (IAP). After executing the requested SPI transfers, the state machine passes the received SPI data (for read commands) and returns to the idle state. The SPI IAP is optional, and it can be enabled by setting the *kExtCmdInterfaceEn* to "true". It is designed to interface with 2 AXI StreamFIFOs, one that stores commands to be transmitted (command queue TX FIFO) and one to store the received data (command RX FIFO). Thus, the IAP consists of two AXI Stream interfaces: the IAP command TX interface and the IAP RX command interface. Each element of the command queue should be represented on 32 bits, having the following format:

31 24	23	22 21	20 13	12 8	7 0
-	Read/Write	Width		Address	Data

Bits	Field Name	Description
23	Read/Write	Write this bit to 1 for a read command and to 0 for a write command
22-21	Width	Only 1 byte SPI transfers are supported. This field should be always 0h.
12-8	Address	DAC SPI register address
7-0	Data	Data byte to be sent to the AD9717. Ignored for read operations.

The DAC Configuration state machine, when in the idle state, monitors the valid signal of the IAP command TX interface (*sCmdTxAxisTvalid*) and, if asserted and if the SPI controller can process a new command (not busy), the ready signal of the IAP command TX interface is asserted for one SysClk100 clock cycle. Write commands only consist of a single transaction on the IAP command TX interface. For read commands, after passing the register read request to the SPI controller and the request successfully completes, the DAC Configuration state machine asserts the valid signal of the IAP command RX interface (*sCmdRxAxisTvalid*) and places the register read data on the data port of the interface (*sCmdRxAxisTdata*). The DAC Configuration state machine waits for the upper layer IP to assert the ready signal o the IAP command interface (*sCmdRxAxisTready*) before it transitions to the idle state.

The initialization configuration command sequence is listed below. After configuring each register, the register data is read back and checked against the expected value in order to determine any SPI transaction error.

1. **SPI Control register**: Set software reset (Address: 00h; Data: 20h)
2. **SPI Control register**: Release software reset (Address: 00h; Data: 00h)
3. **Data Control Register**: Configuration selected: 2's complement, IDATA latched on DCLKIO rising edge, I first of pair on data input pads, data clock input enable, data clock output disable (Address: 02h; Data: **B4h**)
4. **CLKMODE Register**: Clear the reacquire bit (Address: 14h; Data: 00h)
5. **CLKMODE Register**: Set (toggle) the require bit (Address: 14h; Data: 08h)
6. **CLKMODE Register**: Clear the reacquire bit (Address: 14h; Data: 00h)
7. **Memory R/W Register**: Reset UNCALI and UNCALQ bits are reset (Address: 12h; Data: 00h)
8. **Cal Control Register**: Set up calibration clock to SysClk/64 (Address: 0Eh; Data: 02h)
9. **Cal Control Register**: Enable calibration clock (Address: 0Eh; Data: 0Ah)
10. **Cal Control Register**: Select the DAC to self-calibrate (Address: 0Eh; Data: 3Ah)
11. **Memory R/W Register**: Start self-calibration (Address: 12h; Data: 10h)
12. **Cal Control Register**: Check if self-calibration has completed by reading CALSTATI and CALSTATQ bits (Address: 0Fh; Data: -)
13. **Memory R/W Register**: Clear register (Address: 12h; Data: 00h)
14. **Cal Control Register**: Disable calibration clock (Address: 0Eh; Data: 00h)

4.6 SPI Controller (ADI_SPI.vhd)

The SPI controller is designed to carry out basic register access over the Analog Devices 2 wire SPI interface. The controller's parameters and ports are listed below:

Table 5. SPI controller parameter description

Parameter Name	Description
kSysClkDiv	The *sSPI_Clk* signal is obtained by dividing SysClk100 to $2^{kSysClkDiv}$.
kDataWidth	The number of data bits for the data phase of the transaction: only 8 data bits currently supported, parameter provided for future development.
kCommandWidth	The number of bits of the command phase of the SPI transaction.

Table 6. SPI controller port description

Signal Name	Interface	Signal Type	Init State	Description
SysClk100	-	I	N/A	100MHz input clock signal.
asRst_n	-	I	N/A	Active low reset (can be asynchronously asserted but synchronously de-asserted).
sSPI_Clk	SPI	O	N/A	Output SPI clock [1] divided from *SysClk100*. Connected directly to the corresponding Zmod port.
sSDIO	SPI	IO	N/A	2 Wire SPI interface SDIO signal [1]. Connected directly to the corresponding Zmod port.
sCS	SPI	O	N/A	2 Wire SPI interface CS signal [1]. Connected directly to the corresponding Zmod port.
sApStart	-	I	N/A	A pulse on this input initiates the transfers. When asserted, the inputs of the upper layer interface are also registered.
sRdData[kDataWidth-1 : 0]	-	O	N/A	SPI register read received data
sWrData[kDataWidth-1 : 0]	-	I	N/A	SPI register write data.
sAddr[kCommandWidth - 4:0]	-	I	N/A	SPI instruction phase address.
sWidth[1:0]	-	I	N/A	SPI instruction phase word length. The only value currently supported is 0 (1 data byte transferred).
sRdWr	-	I	N/A	Encodes the requested transaction type: ‘1’ - read transaction. ‘0’ – write transaction.
sDone	-	O	N/A	Indicates that the operation requested has completed and that the data placed on the *sRdData* port is valid.
sBusy	-	O	N/A	Indicates when a new command can be processed. *sApStart* is ignored while this signal is asserted.

A SPI transaction is triggered by generating a pulse on the *sApStart* input. When the pulse is detected, the inputs of the upper layer interface are also registered. The *sRdWr* signal encodes the type of the transaction (read/write). The SPI controller further constructs the command word and the data word based on the module’s inputs and formats them on the SPI bus as requested by the Analog Device’s specifications [1]. For read operations, the received data is outputted on the *sRdData* port and can be read by the upper layer module when the *sDone* signal pulses high. When the SPI controller returns to the idle state and is ready to process new commands, the *sBusy* output signal is de-asserted. The *sApStart* signal is ignored while *sBusy* is asserted. The width of the command word is configurable through the *kCommandWidth* generic. The frequency of the SPI output clock is obtained by dividing the controller’s input clock frequency with by a factor of $2^{kSysClkDiv}$ (*kSysClkDiv* is also a generic). Only single byte data transfers are currently supported. More details about the DAC’s SPI interface can be found in [1].

5 IP Top-Level Parameter Description

Table 7. IP core parameter descriptions.

Signal Name	Description
kDAC_Width	DAC resolution (number of bits). It is set to 14 and it is not visible in the customization GUI.
kExtScaleConfigEn	Enables the external gain configuration port. Set to “true” when dynamic configuration is required. Set to “false” when static configuration is sufficient.
kExtCalibEn	Enables the external calibration interface. Set to “true” when the IP core is expected to be interfaced with the processing system through a high level IP. Set to “false” when the core operates in stand alone mode.
kExtCmdInterfaceEn	Enables the upper layer IP SPI configuration interface. Set to “true” when the IP core is expected to be interfaced with the processing system through a high level IP. This will enable the processor to access the Zmod AWG 1411 SPI interface. Set to “false” when initial configuration described in Section 4.5 is sufficient.
kCh1ScaleStatic	Channel1 gain select static configuration parameter. If the *kExtRelayConfigEn* parameter is set to “false”, the Zmod’s gain will be configured according to this parameter’s value at initialization time. If the value of *kExtRelayConfigEn* parameter is “true”, this parameter is ignored. • 1 = High Gain. • 0 = Low Gain.
kCh2ScaleStatic	Channel2 gain select static configuration parameter. If the *kExtRelayConfigEn* parameter is set to “false”, the Zmod’s gain will be configured according to this parameter’s value at initialization time. If the value of *kExtRelayConfigEn* parameter is “true”, this parameter is ignored. • 1 = High Gain. • 0 = Low Gain.
kCh1LgMultCoefStatic[17:0]	Channel1 low gain multiplicative calibration coefficient. If the *kExtCalibEn* parameter is set to “false”, the calibration block will expect the value of the multiplicative coefficient to be passed by this parameter. If the value of *kExtCalibEn* parameter is “true”, this parameter is ignored, and the high level IP is expected to update the corresponding external port.
kCh1LgAddCoefStatic[17:0]	Channel1 low gain additive calibration coefficient. If the *kExtCalibEn* parameter is set to “false”, the calibration block will expect the value of the additive coefficient to be passed by this parameter. If the value of *kExtCalibEn* parameter is “true”, this parameter is ignored, and the high level IP is expected to update the corresponding external port.
kCh1HgMultCoefStatic[17:0]	Channel1 high gain multiplicative calibration coefficient. If the *kExtCalibEn* parameter is set to “false”, the calibration block will expect the value of the multiplicative coefficient to be passed by this parameter. If the value of *kExtCalibEn* parameter is “true”, this parameter is ignored, and the high level IP is expected to update the corresponding external port.
kCh1HgAddCoefStatic[17:0]	Channel1 high gain additive calibration coefficient. If the *kExtCalibEn* parameter is set to “false”, the calibration block will expect the value of the additive coefficient to be passed by this parameter. If the value of *kExtCalibEn* parameter is “true”, this parameter is ignored, and the high level IP is expected to update the corresponding external port.

kCh2LgMultCoefStatic[17:0]	Channel2 low gain multiplicative calibration coefficient. If the *kExtCalibEn* parameter is set to "false", the calibration block will expect the value of the multiplicative coefficient to be passed by this parameter. If the value of *kExtCalibEn* parameter is "true", this parameter is ignored, and the high level IP is expected to update the corresponding external port.
kCh2LgAddCoefStatic[17:0]	Channel2 low gain additive calibration coefficient. If the *kExtCalibEn* parameter is set to "false", the calibration block will expect the value of the additive coefficient to be passed by this parameter. If the value of *kExtCalibEn* parameter is "true", this parameter is ignored, and the high level IP is expected to update the corresponding external port.
kCh2HgMultCoefStatic[17:0]	Channel2 high gain multiplicative calibration coefficient. If the *kExtCalibEn* parameter is set to "false", the calibration block will expect the value of the multiplicative coefficient to be passed by this parameter. If the value of *kExtCalibEn* parameter is "true", this parameter is ignored, and the high level IP is expected to update the corresponding external port.
kCh2HgAddCoefStatic[17:0]	Channel2 high gain additive calibration coefficient. If the *kExtCalibEn* parameter is set to "false", the calibration block will expect the value of the additive coefficient to be passed by this parameter. If the value of *kExtCalibEn* parameter is "true", this parameter is ignored, and the high level IP is expected to update the corresponding external port.

6 IP Top-Level Port Description

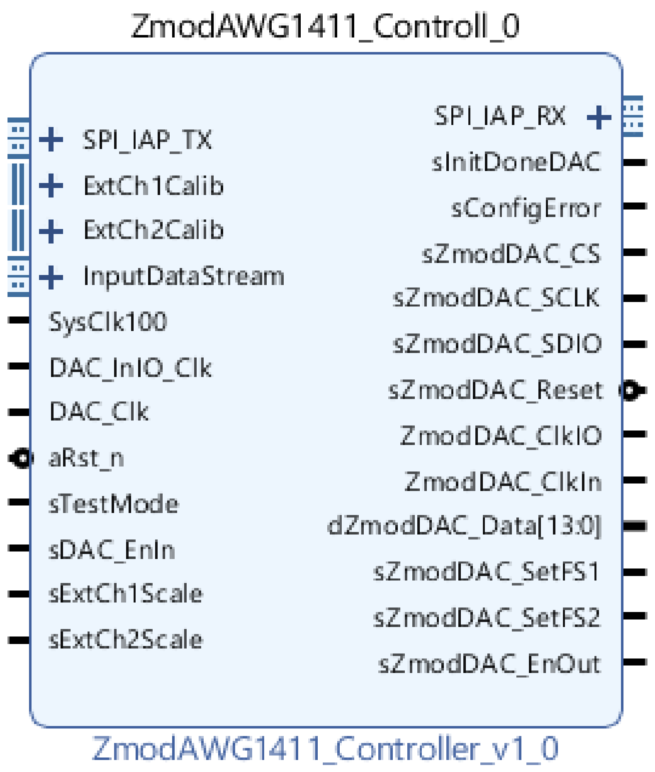


Figure 5: Zmod AWG 1411 Controller IP.

Table 8. IP core port description

Signal Name	Interface	Signal Type	Init State	Description
SysClk100	-	I	N/A	100MHz input clock signal.
DAC_InIO_Clk	-	I	N/A	Data path clock input.
DAC_Clk	-	I	N/A	Input clock used to generate the *ZmodDAC_ClkIO* and *ZmodDAC_ClkIn* output clock signals for the AD9717 DAC. *DAC_Clk* should have the same frequency but a 90° phase shift with respect to *DAC_InIO_Clk*.
aRst_n	-	I	N/A	Asynchronous reset of negative polarity which resets the logic in all four clock domains. Must must be asserted for at least 2*$T_{slowest}$.
sInitDoneDAC	-	O	N/A	Flag indicating when the Zmod's DAC 1411 initialization is complete.
sConfigError	-	O	N/A	This flag is asserted if the DAC initialization fails. An IP reset is required to clear this flag.
cDataAxisTdata[31:0]	Input Data Stream	I	N/A	(Slave) AXI Stream Data interface TDATA port. Channel1 and Channel2 data are concatenated as follows: Channel1 -> cDataAxisTdata[31:18]. Channel2 -> cDataAxisTdata[15:2].
cDataAxisTvalid	Input Data Stream	I	N/A	(Slave) AXI Stream Data interface TVALID signal.
cDataAxisTready	Input Data Stream	O	N/A	(Slave) AXI Stream Data interface TREADY signal.
cExtCh1LgMultCoef[17:0]	ExtCh1Calib	I	N/A	Channel1 low gain multiplicative coefficient external port. This port is enabled by setting the *kExtCalibEn* parameter to "true".
cExtCh1LgAddCoef[17:0]	ExtCh1Calib	I	N/A	Channel1 low gain additive coefficient external port. This port is enabled by setting the *kExtCalibEn* parameter to "true".
cExtCh1HgMultCoef[17:0]	ExtCh1Calib	I	N/A	Channel1 high gain multiplicative coefficient external port. This port is enabled by setting the *kExtCalibEn* parameter to "true".
cExtCh1HgAddCoef[17:0]	ExtCh1Calib	I	N/A	Channel1 high gain additive coefficient external port. This port is enabled by setting the *kExtCalibEn* parameter to "true".
cExtCh2LgMultCoef[17:0]	ExtCh2Calib	I	N/A	Channel2 low gain multiplicative coefficient external port. This port is enabled by setting the *kExtCalibEn* parameter to "true".
cExtCh2LgAddCoef[17:0]	ExtCh2Calib	I	N/A	Channel2 low gain additive coefficient external port. This port is enabled by setting the *kExtCalibEn* parameter to "true".

cExtCh2HgMultCoef[17:0]	ExtCh2Calib	I	N/A	Channel2 high gain multiplicative coefficient external port. This port is enabled by setting the *kExtCalibEn* parameter to "true".
cExtCh2HgAddCoef[17:0]	ExtCh2Calib	I	N/A	Channel2 high gain additive coefficient external port. This port is enabled by setting the *kExtCalibEn* parameter to "true".
sDAC_EnIn	-	I	N/A	Zmod AWG 1411 output relay control. When set to '1', the Zmod's analog outputs are enabled. When set to '0', the Zmod's analog outputs are in high impedance state.
sExtCh1Scale	-	I	N/A	Channel1 gain select external port. This port is enabled by setting the *kExtScaleConfigEn* parameter to "true". • 1 = High Gain. • 0 = **Low Gain**.
sExtCh2Scale	-	I	N/A	Channel2 gain select external port. This port is enabled by setting the *kExtScaleConfigEn* parameter to "true". • 1 = High Gain. • 0 = **Low Gain**.
sTestMode	-	I	N/A	*sTestMode* is used to bypass the calibration block. When asserted, raw samples are sent to the DAC.
sCmdTxAxisTdata[31:0]	SPI IAP	I	N/A	IAP command TX interface (Slave AXI Stream) data port. For more information see section 4.5.
sCmdTxAxisTvalid	SPI IAP	I	N/A	IAP command TX interface (Slave AXI Stream) valid signal. For more information see section 4.5.
sCmdTxAxisTready	SPI IAP	O	N/A	IAP command TX interface (Slave AXI Stream) ready signal. For more information see section 4.5.
sCmdRxAxisTdata[31:0]	SPI IAP	O	N/A	IAP command RX interface (Master AXI Stream) data port. For more information see section 4.5.
sCmdRxAxisTvalid	SPI IAP	O	N/A	IAP command RX interface (Master AXI Stream) valid signal. For more information see section 4.5.
sCmdRxAxisTready	SPI IAP	I	N/A	IAP command RX interface (Master AXI Stream) ready signal. For more information see section 4.5.
sZmodDAC_ClkIO	Zmod	O	N/A	Should be connected to the AD9717 DCLKIO input [1].
sZmodDAC_ClkIn	Zmod	O	N/A	Should be connected to the AD9717 CLKIN input [1].
sZmodDAC_Data[13:0]	Zmod	O	N/A	Should be connected to the AD9717 DB[13:0] input [1].
sZmodDAC_Reset	Zmod	O	N/A	Should be connected to the AD9717 RESET/PINMD input [1].
sZmodDAC_SDIO	Zmod	IO	N/A	SPI SDIO signal [1].
sZmodDAC_CS	Zmod	O	N/A	SPI CS signal [1].
sZmodDAC_Sclk	Zmod	O	N/A	SPI output clock.
sZmodDAC_SetFS1	Zmod	O	N/A	This signal controls the Zmod AWG 1411 gain select analog switch for channel1.

sZmodDAC_SetFS2	Zmod	O	N/A	This signal controls the Zmod AWG 1411 gain select analog switch for channel2.
sZmodDAC_EnOut	Zmod	O	N/A	This signal controls the Zmod AWG 1411 output enable relay.

7 IP Core customization

The IP through its customizable parameters (*kExtScaleConfigEn*, *kExtCalibEn*, *kExtCmdInterfaceEn*) enables the user to opt for a more basic design, with minimal external ports or a more configurable but also more complex design with several ports that enable dynamic configuration of several hardware features. When the Zmod AWG 1411 Low Level Controller is controlled by a higher level IP, the hardware configuration features are expected to be controlled by the upper layers (processing system), so all interfaces should be enabled and connected to the upper layer IP Core.

8 References

The following documents provide additional information on the subjects discussed:

1. Analog Devices, AD9717 Datasheet, Rev B.
2. Xilinx Inc., *UG472: 7 Series FPGAs Clocking Resources*, v1.6, October 2, 2012.
3. Xilinx Inc., UG903: Using Constraints, v2014.3, October 31, 2014.
4. Xilinx Inc., *UG471: 7 Series FPGAs SelectIO Resources*, v1.4, May 13, 2014.