Temperature Dependent Dynamic Voltage Scaling Implementation

XAPP1394 (v1.0) June 14, 2023

Summary

Dynamic voltage scaling (DVS) is a powerful technique that allows the core voltage of an SoC to be decreased or increased during runtime to account for junction temperature (T_J) changes, reduce power dissipation, or increase the performance of the SoC. AMD devices might require DVS to compensate for low temperatures or increase power savings. This application note describes the hardware and software changes that need to be implemented to support this feature for low-temperature compensation for -2LI stock-keeping units (SKU) of AMD Versal™ adaptive SoCs.

Download the reference design files for this application note from the Xilinx website. For detailed information about the design files, see Implementation Details.

Introduction

With the increased capabilities of high-performance computing and power complexity, current demands on power supplies also increase. It leads the industry to improve intelligent power as a means to reduce overall power, increase usable binned devices, and minimize cost. DVS is a powerful technique that more optimally powerful SoCs can leverage for the required use case.

Why is DVS Required?

To enable low core voltage in a -2 speed, industrial temperature grade, or -2LI SKU of a Versal device, a DVS scheme is required for reliable functionality at temperatures below 0°C. This enables -2LI performance over the entire –40°C to 100°C temperature range and broadens the number of applications that can be supported in the industrial temperature range at the low core voltage (-L). The DVS scheme scales the programmable logic core voltage (VCCINT) from 0.7 V to 0.725 V at lower temperatures to maintain the performance without incurring a power penalty at higher temperatures. The scheme uses the System Monitor (SYSMON) located within the platform management controller (PMC) to read the device junction temperature in a regular interval, sampling every 100 ms and changing the voltage as necessary.

When applying DVS to VCCINT, the voltage changes based on the junction temperature, which requires the VCCINT rail to have a dedicated voltage regulator that is not connected to any other power rails within the system. This might include an extra regulator that is required for a minimum rails consolidation but does not impact full power management implementations. Refer to the Power Design Manager (PDM) tool (download at www.xilinx.com/power) for more details

on rail consolidation. To change the voltage regulator output voltage, the PMC signals via PMBus or I2C to change the output voltage through voltage regulator module (VRM) register writes. It requires two PMC MIO pins in banks 500/501 for I2C/PMBus. It can also connect to a GPIO to indicate a voltage high or voltage low mode for use cases where an I2C/PMBus master already exists or is unavailable. If your device is designed never to go below 0°C, a -2LE SKU is recommended because the extended temperature range is from 0°C to 110°C.

Implementation Details

Overview of Operation

The default mode when the -2LI device is powered on is Voltage High mode (0.725 V) and the default VRM voltage for VCCINT should be set as this value. After the drivers are loaded and operating, the internal SYSMON of the AMD device continuously samples the junction temperature at 100 ms intervals. During this time, if the junction temperature (T_J) falls below or rises above the pre-determined thresholds, it triggers a voltage change.

This is executed by communication with the voltage regulator via I2C/PMBus to change to the appropriate voltage. The CDO file is configured to determine the I2C/PMBus operation, values, and configuration of the regulator.

Using these drivers, the AMD device operates as the PMBus/I2C master, and no other masters should be on the communication line because this might lead to contention and incorrect operation of the drivers. If there is already a master I2C/PMBus in your system, a configurable I/O can be used to indicate if the device should be in voltage high (0.725 V) or voltage low (0.7 V) mode by pulling the I/O to a High or Low state.

The thresholds for AMD DVS are seen in the following graph. There is an added window of hysteresis to prevent sudden changes to the voltage. It is important to note that other settings within your voltage regulator might need to change with the voltage. Overvoltage protection is one such example that needs to track with the output voltage. There are many regulators that can track over/undervoltage as a percentage of the output voltage to prevent additional register changes to the regulator during runtime. A regulator with a suitable ramp time between voltage Low/High must be used, and the voltage change must happen within 40 ms of the voltage indication.

Figure 1: **Temperature Threshold and Voltage Change**

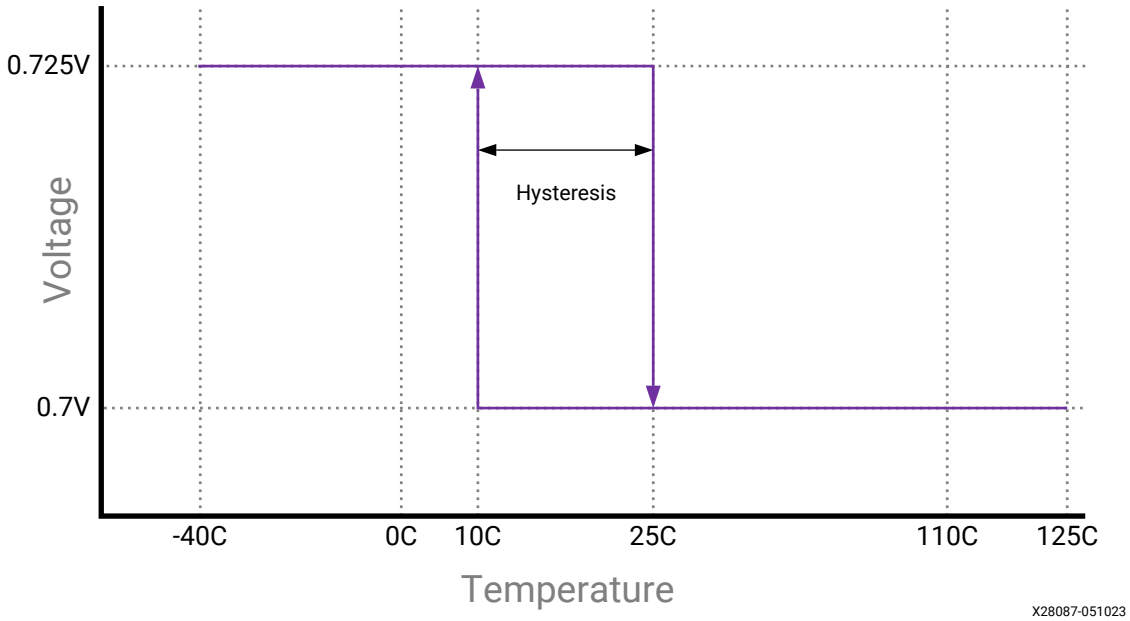
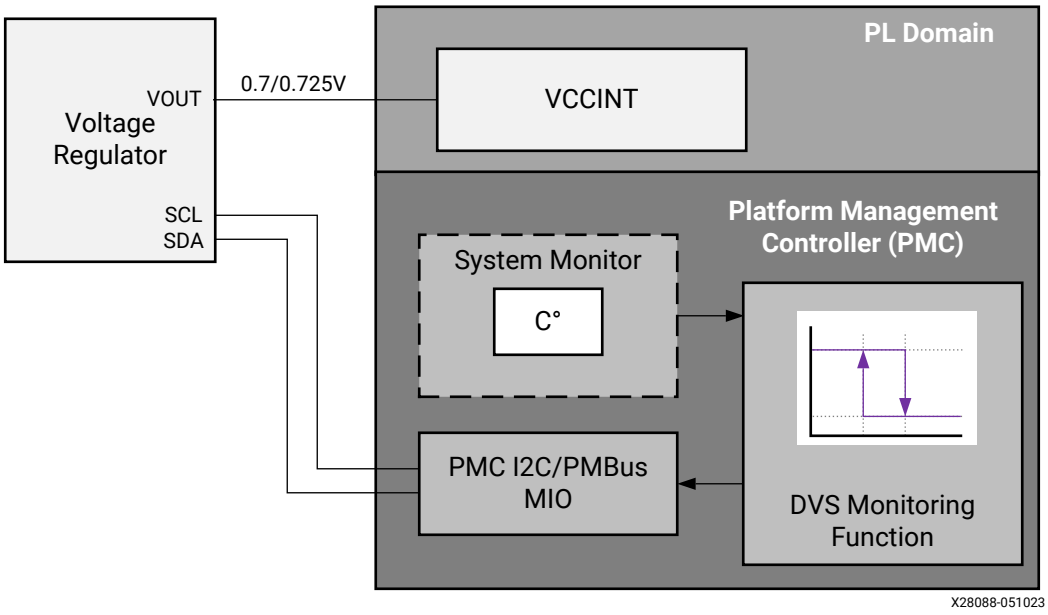


Figure 2: **System Implementation Diagram**



Hardware Implementation

- Voltage regulator requirements
  - PMBus or I2C compatible with address range 0x00–0x7F
  - Digitally configurable output voltage
- AMD device requirements
  - Two multiplexed I/Os configured for I2C/PMBus, PMC bank 500/501

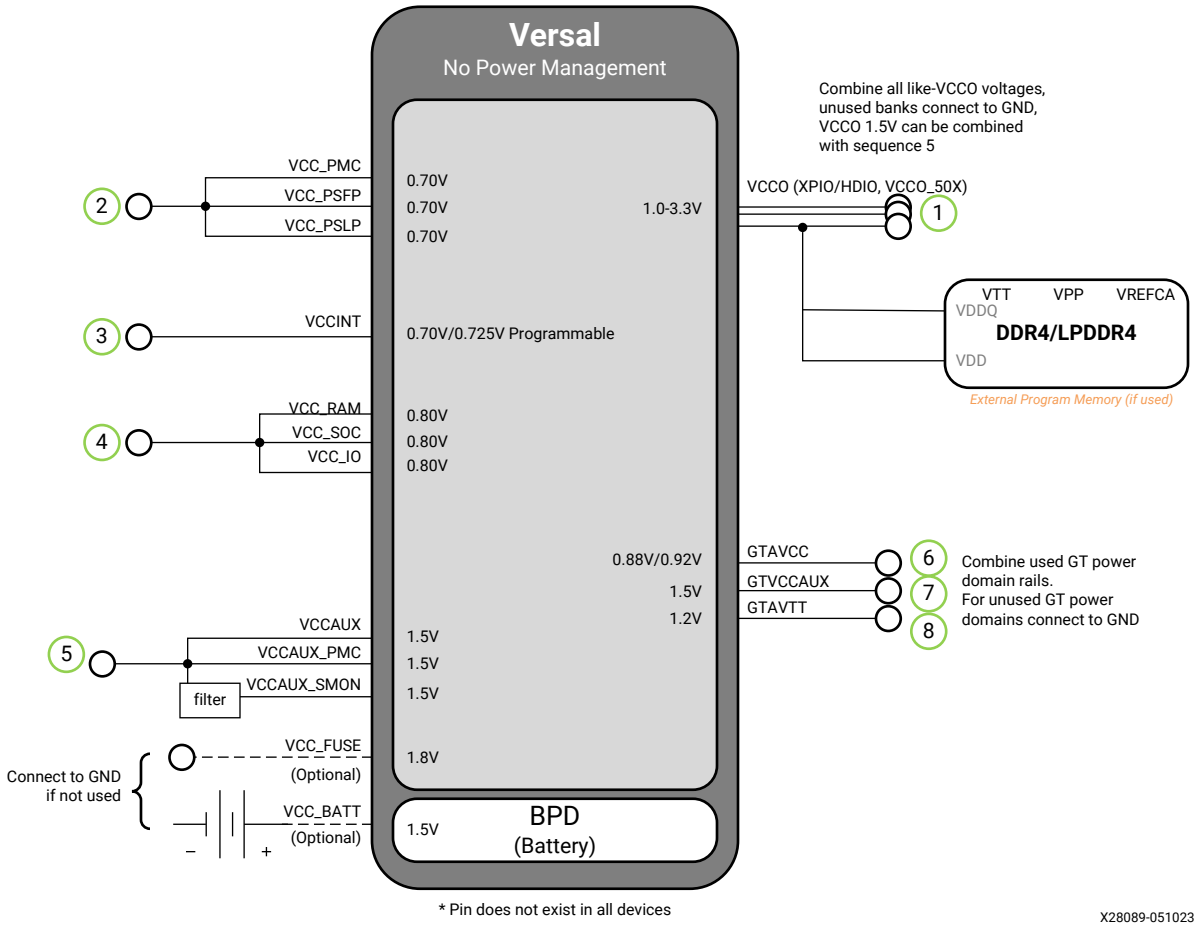
- A board controller that implements DVS functionality, not supported (optional)

Recommended power delivery solutions can be found at Power Efficiency on the AMD website.

Rail Consolidation Impacts

Due to the dynamic voltage changing on VCCINT, this rail cannot be consolidated with any other supply in the system as this rail requires a dedicated voltage regulator, a specific rail consolidation is used for minimum rails. See the PDM tool for the latest AMD Versal™ device power rail consolidations.

Figure 3: **DVS Minimum Rails Consolidation for -2LI**



Characteristics of Dynamic Voltage Scaling

Due to the small step in voltage required, there is rarely any reason to change the compensation network for your voltage regulator. However, this should be confirmed with the power vendor's solution to ensure that there are no other registers that require changes aside from the output voltage.

Software Implementation

XilPM (PLM) software monitors the Versal device's temperature periodically and adjusts the power regulator that provides power for the VCCINT power rail accordingly. PLM/XilPM needs two sets of information for this implementation: The policy as to which power rail and at what temperatures the voltage adjustment needs to be made. Both sets of information are provided to PLM through separate CDO files.

CDO Implementation

In this application note, the focus is on the CDO formats targeting the VCK190 evaluation board with DVS that covers in detail the explanation of the arguments passed to the `pm_add_node` command.

The software implementation is divided into the following steps:

- Policy CDO
- Board CDO
- Software Driver Implementation
- Build Instructions
- Test Steps

Policy CDO

The voltage adjustment policy based on temperature is provided to PLM/XilPM with the following CDO command entry:

```
#Lower Threshold: 5°C (0x280); Upper Threshold: 30°C (0xf00)
```

```
pm_add_node 0x4328030 0x3 0x280 0x3 0xf00 0x2
```

***Note*:** Do not modify the above values because they follow the supported upper and lower trip points.

Table 1: **Register Descriptions**

Node Id	0x4328030	PM_POWER_VCCINT_PL	Node Id of VCCINT Power Rail
Type: 0x1 (I2C/PMBus); 0x2 (Pgood); 0x3 (Temperature Voltage Adjust)	0x3	Temperature Voltage Adjust	Instructs XilPM how to handle the power rail.
Lower Temperature Threshold	0x280		5°C * 128 = 0x280 (Do not modify)

Table 1: **Register Descriptions** *(cont'd)*

Node Id	0x4328030	PM_POWER_VCCINT_PL	Node Id of VCCINT Power Rail
Upper Voltage Mode Id	0x3		Indicates which power mode of VCCINT the power rail transitions when the 'Lower Temperature Threshold' of 5°C is met. Different power modes for VCCINT are defined by "# Four power modes for VCCINT rail: 0 - OFF, 1 - ON, 2 - LOWER (0.70V), 3 - UPPER (0.725V)" CDO line. Because the power mode is 0x3, it indicates that when the temperature reaches 5°C, the power rail is transitioned to UPPER voltage of 0.725V.
Upper Temperature Threshold	0xf00		30°C * 128 = 0xf00 (Do not modify)
Lower Voltage Mode Id	0x2		Indicates which power mode of VCCINT the power rail transitions when the 'Upper Temperature Threshold' of 30°C is met. The VCCINT power rail is transitioned to power mode 2 (2–LOWER (0.70V)).

Board CDO

The second set of information is about the on-board power regulator that provides power to the internal power rail of the Versal device. This information is board-specific and varies based on the power topology of the board. On the VCK190 board, the power regulator that provides power for the VCCINT rail is described in CDO format with the following entries along with an explanation of each of its arguments:

```
# VCCINT Power Regulator on VCK190
```

```
pm_add_node 0x442c001 0x460101 0x1822402d 0x10174
```

```
# Four power modes for VCCINT_PL rail: 0 - OFF, 1 - ON, 2 - LOWER (0.70V),
3 - UPPER (0.725V)
```

```
pm_add_node 0x4328030 0x1 0x442c001 0x4 0x300 0x2010002 0x1021a02 0 0x301
0x2000002 0x1021a02 0x80
```

```
0x402 0x2000002 0x21030001 0x10200b3 0x80 0x403 0x2000002 0x21030001
0x10200ba 0x80
```

Table 2: **Register Descriptions**

Node Id	0x442c001	Node Id of VCCINT Power Rail	Node Id of VCCINT Power Regulator
[23:16]: I2C Address; [15:8]: No of Commands; [7:0]: Control Method, 1 (I2C/PMBus), 2 (GPIO)	0x460101	[23:16] = 0x46; [15:8] = 0x1; [7:0] = 0x1	The type of controlling the regulator is I2C/PMBus; the number of multiplexers to configure to get to the regulator is 1; the I2C address of the power regulator is 0x46.
Parent I2C Node Id	0x1822402d	PM_DEV_I2C_PMC	Node Id for I2C controller.
[7:0]: Mux I2C address; [15:8]: Length of Command; [23:16]: First Byte; [31:24]: Second Byte (if any)	0x10174	[23:16] = 0x1; [15:8] = 0x1; [7:0] = 0x74	The I2C address of the multiplexer is 0x74; the number of I2C bytes to configure the multiplexer channel is 1; the value of byte to write to the multiplexer is 0x1.

Table 3: **Register Descriptions**

Node Id	0x4328030	PM_POWER_VCCINT_PL	Node Id of VCCINT Power Rail	Upper/Lower Threshold
Type: 0x1 (I2C/ PMBus); 0x2 (Pgood); 0x3 (Temperature Voltage Adjust)	0x1	I2C/PMBus	Instructs XilPM how to handle the power rail.	
Parent Regulator Node Id	0x442c001		Node id of VCCINT Power Regulator.	
Number of modes (states) supported for this power rail	0x4		There are four power modes for this power rail: UPPER (3), LOWER (2), ON (1), OFF (0).	
[15:8]: Number of I2C commands; [7:0]: Mode Id	0x300	[15:8] = 0x3; [7:0] = 0x0	The power rail OFF mode is achieved by three I2C commands.	
I2C Commands - [7:0]: Number of Bytes; [15:8]: Byte 1; [23:16]: Byte 2; [31:24]: Byte 3 (if any)	0x200002	[31:24] = 0x2; [23:16] = 0x0; [15:8] = 0x0; [7:0] = 0x2	The first I2C command is two bytes. The first byte is 0x0 and the second byte is 0x0. This 2-byte command selects the page of the regulator that drives the power rail. The second I2C command is also two bytes ([31:24]).	
	0x1021a02	[31:24] = 0x1; [23:16] = 0x2; [15:8] = 0x1a; [7:0] = 0x2	The payload of the second I2C command is '0x2,0x1a' which is 'ON OFF CONFIG' PMBus command. The third I2C command is two bytes as well ([23:16]).	
	0x0	[7:0] = 0x0	The first byte of the third I2C command is from previous word ([31:24] = 0x1) and the second byte of third I2C command is 0x0. The payload '01, 00' is 'OPERATION OFF' PMBus command.	
[15:8]: Number of I2C commands; [7:0]: Mode Id	0x301	[15:8] = 0x3; [7:0] = 0x1	The first byte of the third I2C command is from the previous word ([31:24] = 0x1) and the second byte of the third I2C command is 0x0. The payload '01, 00' is 'OPERATION OFF' PMBus command.	
I2C Commands - [7:0]: Number of Bytes; [15:8]: Byte 1; [23:16]: Byte 2; [31:24]: Byte 3 (if any)	0x2000002	[31:24] = 0x2; [23:16] = 0x0; [15:8] = 0x0; [7:0] = 0x2	The first I2C command is two bytes. The first byte is 0x0 and the second byte is 0x0. This 2-byte command selects the page of the regulator that drives the power rail. The second I2C command is also two bytes ([31:24]).	
	0x1021a02	[31:24] = 0x1; [23:16] = 0x2; [15:8] = 0x1a; [7:0] = 0x2	The payload of the second I2C command is '0x2,0x1a,' which is the 'ON OFF CONFIG' PMBus command. The third I2C command is also two bytes([23:16]).	

Table 3: **Register Descriptions** *(cont'd)*

Node Id	0x4328030	PM_POWER_VCCINT_PL	Node Id of VCCINT Power Rail	Upper/Lower Threshold
	0x80	[7:0] = 0x80	The first byte of the third I2C command is from previous word ([31:24] = 0x1) and the second byte of the third I2C command is 0x80. The payload '0x1, 0x80' is the 'OPERATION ON' PMBus command.	
[15:8]: Number of I2C commands; [7:0]: Mode Id	0x402	[15:8] = 0x4; [7:0] = 0x2	The power rail LOWER mode is achieved by four I2C commands.	
I2C Commands - [7:0]: Number of Bytes; [15:8]: Byte 1; [23:16]: Byte 2; [31:24]: Byte 3 (if any)	0x2000002	[31:24] = 0x2; [23:16] = 0x0; [15:8] = 0x0; [7:0] = 0x2	The first I2C command is two bytes. The first byte is 0x0 and the second byte is 0x0. This 2-byte command selects the page of the regulator that drives the power rail. The second I2C command is also two bytes ([31:24]).	
	0x21030001	[31:24] = 0x21; [23:16] = 0x3; [15:8] = 0x0; [7:0] = 0x1	The payload of the second I2C command is '0x1,0x0' which is the OPERATION PMBus command to disable VOUT. The third I2C command is three bytes ([23:16]).	
	0x102YYXX	[31:24] = 0x1; [23:16] = 0x2 [15:8] = YY, [7:0] = XX	The first byte of the third I2C command is from the previous word, the second byte of the third I2C command is XX, and the third byte is YY. The payload '0x21, XX, YY' is the VOUT_COMMAND PMBus command, which sets the VOUT to XXYY. The fourth I2C command is two bytes. The first byte of the fourth I2C command is the OPERATION PMBus command.	0.70V: XX = 0xb3 YY = 0x00;
	0x80	[7:0] = 0x80	The first byte of the fourth I2C command is from previous word. The second byte is 0x80. This OPERATION PMBus command enables the VOUT.	
[15:8]: Number of I2C commands; [7:0]: Mode Id	0x403	[15:8] = 0x4; [7:0] = 0x3	The power rail UPPER mode is achieved by four I2C commands.	
I2C Commands - [7:0]: Number of Bytes; [15:8]: Byte 1; [23:16]: Byte 2; [31:24]: Byte 3 (if any)	0x2000002	[31:24] = 0x2; [23:16] = 0x0; [15:8] = 0x0; [7:0] = 0x2	The first I2C command is two bytes. The first byte is 0x0 and the second byte is 0x0. This 2-byte command selects the page of the regulator that drives the power rail. The second I2C command is also two bytes ([31:24]).	
	0x21030001	[31:24] = 0x21; [23:16] = 0x3; [15:8] = 0x0; [7:0] = 0x1	The payload of the second I2C command is '0x1,0x0' which is the OPERATION PMBus command to disable VOUT. The third I2C command is three bytes ([23:16]).	

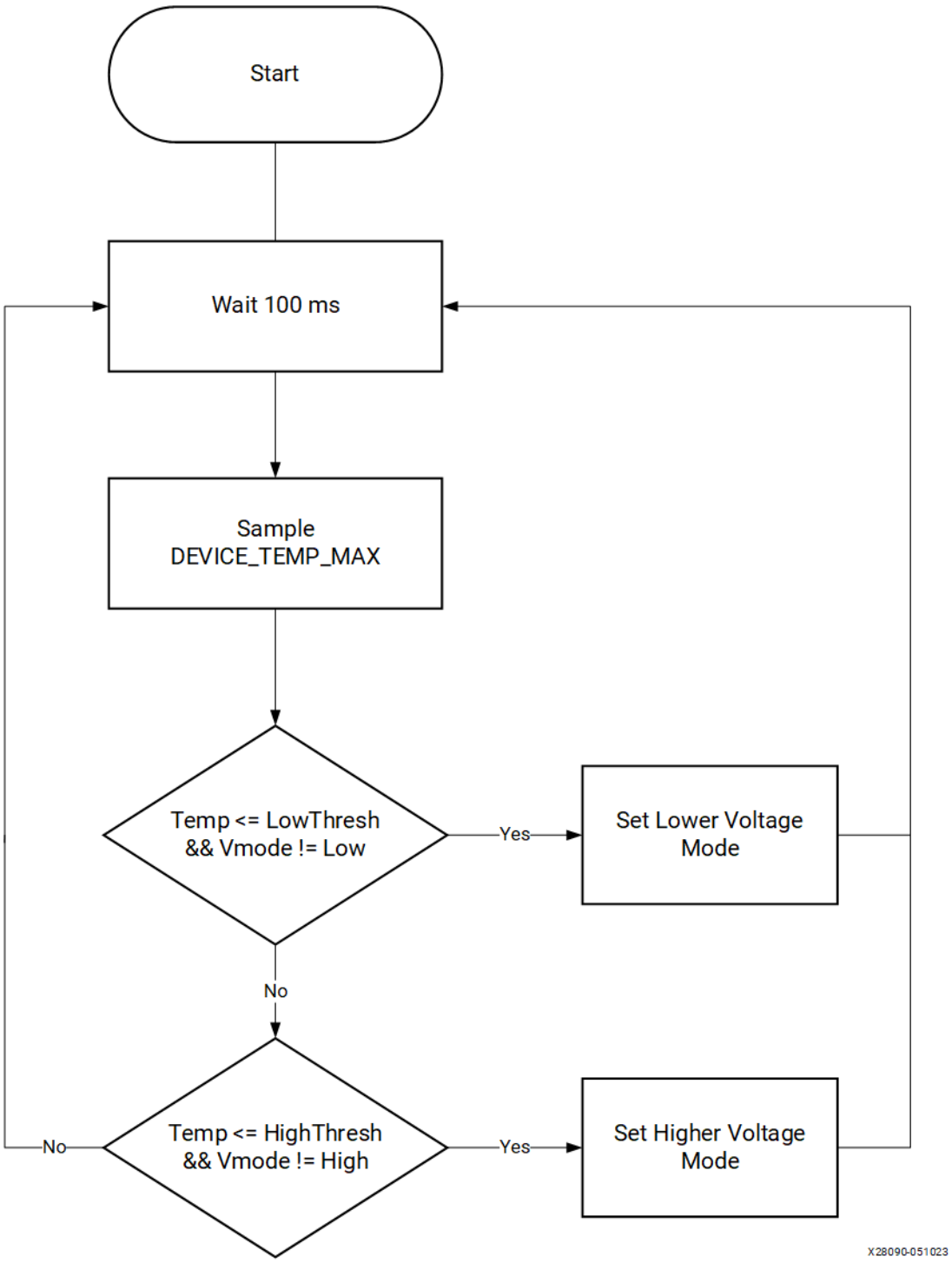
Table 3: **Register Descriptions** *(cont'd)*

Node Id	0x4328030	PM_POWER_VCCINT_PL	Node Id of VCCINT Power Rail	Upper/Lower Threshold
	0x102NNMM	[31:24] = 0x1; [23:16] = 0x2; [15:8] = NN; [7:0] = MM	The first byte of the third I2C command is from the previous word, the second byte of the third I2C command is MM, and the third byte is NN. The payload '0x21, MM, NN' is the VOUT_COMMAND PMBus command, which sets the VOUT to MMNN. The fourth I2C command is two bytes. The first byte of the fourth I2C command is the OPERATION PMBus command.	0.725V: NN = 0x00 MM = 0xba;
	0x80	[7:0] = 0x80	The first byte of the fourth I2C command is from the previous word. The second byte is 0x80. This OPERATION PMBus command enables the VOUT.	

Software Driver Implementation

The high-level overview of the software implementation is depicted in the following figure.

Figure 4: **DVS Temperature Monitoring Loop**



The implementation requires the PMC I2C/PMBus controller to be configured to drive the I2C0 bus on the VCK190 board. Upon parsing the policy CDO, as shown in the figure, a cyclic task is scheduled to be run every 100 ms. At each invocation, the content of the DEVICE_TEMP_MAX register is obtained through the SYSMON driver.

This temperature reading is compared with the threshold values specified in the policy CDO and the current state of power mode. The hysteresis algorithm is used to determine when to increase the voltage when crossing the lower threshold and when to decrease the voltage when crossing the upper threshold.

```
UpperTempThresh = Rail->TempVoltAdj->UpperTempThresh;
    LowerTempThresh = Rail->TempVoltAdj->LowerTempThresh;
    UpperVoltMode = Rail->TempVoltAdj->UpperVoltMode;
    LowerVoltMode = Rail->TempVoltAdj->LowerVoltMode;
    CurrentVoltMode = &Rail->TempVoltAdj->CurrentVoltMode;

    /* Validate that the argument passed in is a power rail */
    if ((u32)XPM_NODETYPE_POWER_RAIL != NODETYPE(Rail->Power.Node.Id)) {
        Status = XST_INVALID_PARAM;
        goto done;
    }

    /*
     * If Root SysMon is not initialized yet, skip the cycle until
     * it is initialized.
     */
    if (0U == SysMonInstPtr->Config.BaseAddress) {
        Status = XST_SUCCESS;
        goto done;
    }

    /* Read DEVICE_TEMP_MAX register through SysMon driver */
    CurrentTemp = XSysMonPsv_ReadDeviceTemp(SysMonInstPtr, XSYSMONPSV_VAL);

    /*
     * If the current temperature is at or below lower threshold and
     * we are not in upper voltage mode, make an adjustment to higher
     * voltage.  Similarly, if the temperature is at or above upper
     * threshold and not in lower voltage mode, make an adjustment
     * to lower voltage.  If the temperature is between lower and upper
     * threshold, no adjustment is needed.
     */
    if ((XSysMonPsv_FixedToFloat(CurrentTemp) <=
         XSysMonPsv_FixedToFloat(LowerTempThresh)) &&
        (*CurrentVoltMode != UpperVoltMode)) {
        PmDbg("Current temperature is 0x%x, Set voltage to upper mode "
              "%d\n\r", CurrentTemp, UpperVoltMode);
        Status = XPmRail_Control(Rail, (u8)XPM_POWER_STATE_ON,
                     UpperVoltMode);
        if (XST_SUCCESS != Status) {
            DbgErr = XPM_INT_ERR_RAIL_UPPER_VOLT;
            goto done;
        }

        *CurrentVoltMode = UpperVoltMode;
    } else if ((XSysMonPsv_FixedToFloat(CurrentTemp) >=
            XSysMonPsv_FixedToFloat(UpperTempThresh)) &&
           (*CurrentVoltMode != LowerVoltMode)) {
        PmDbg("Current temperature is 0x%x, Set voltage to lower mode "
              "%d\n\r", CurrentTemp, LowerVoltMode);
        Status = XPmRail_Control(Rail, (u8)XPM_POWER_STATE_ON,
                     LowerVoltMode);
        if (XST_SUCCESS != Status) {
            DbgErr = XPM_INT_ERR_RAIL_LOWER_VOLT;
            goto done;
        }

        *CurrentVoltMode = LowerVoltMode;
    } else if ((XSysMonPsv_FixedToFloat(CurrentTemp) <
            XSysMonPsv_FixedToFloat(UpperTempThresh)) &&
           (XSysMonPsv_FixedToFloat(CurrentTemp) >
```

```
            XSysMonPsv_FixedToFloat(LowerTempThresh)) &&
           (0U == *CurrentVoltMode)) {
        PmDbg("Current temperature is 0x%x, Set voltage to lower mode "
              "%d\n\r", CurrentTemp, LowerVoltMode);
        Status = XPmRail_Control(Rail, (u8)XPM_POWER_STATE_ON,
                     LowerVoltMode);
        if (XST_SUCCESS != Status) {
            DbgErr = XPM_INT_ERR_RAIL_LOWER_VOLT;
            goto done;
        }

        *CurrentVoltMode = LowerVoltMode;
    } else {
        Status = XST_SUCCESS;
    }
```

When adjusting the voltage based on the selected power mode, a series of I2C commands are issued to the power regulator. The relevant source code can be found in the embeddedsw GitHub repository files (lib/sw_services/xilpm/src/versal/server/xpm_rail.c and lib/sw_services/xilpm/src/versal/server/xpm_rail.h).

The following code snippet implements this functionality. This compares the current temperature with lower and upper temperature thresholds. If it is at or below a lower threshold and currently not at the upper voltage level, it adjusts the voltage to a higher value. If it is at or above an upper threshold and currently not at the lower voltage level, it adjusts the voltage to a lower value. If it is within lower and upper threshold boundaries, no voltage adjustment is made.

```
/*
     * This routine is invoked periodically by the task scheduler and its
      * task is to determine if a voltage adjustment is needed, based on
current
      * temperature and current voltage level.  The argument passed points
to power
      * rail that needs to be adjusted.
     */
    static int XPmRail_CyclicTempVoltAdj(void *Arg)
    {
        int Status = XST_FAILURE;
        u16 DbgErr = XPM_INT_ERR_UNDEFINED;
        XPm_Rail *Rail = (XPm_Rail *)Arg;
        u32 UpperTempThresh, LowerTempThresh;
        u8 UpperVoltMode, LowerVoltMode;
        u32 CurrentTemp;
        u8 *CurrentVoltMode;
        XSysMonPsv *SysMonInstPtr = XPlmi_GetSysmonInst();


        UpperTempThresh = Rail->TempVoltAdj->UpperTempThresh;
        LowerTempThresh = Rail->TempVoltAdj->LowerTempThresh;
        UpperVoltMode = Rail->TempVoltAdj->UpperVoltMode;
        LowerVoltMode = Rail->TempVoltAdj->LowerVoltMode;
        CurrentVoltMode = &Rail->TempVoltAdj->CurrentVoltMode;


        /* Validate that the argument passed in is a power rail */
        if ((u32)XPM_NODETYPE_POWER_RAIL != NODETYPE(Rail->Power.Node.Id)) {
            Status = XST_INVALID_PARAM;
            goto done;
        }
```

Build Instructions

For demonstration purposes, the following is a list of the VCK190 build steps to validate the temperature DVS functionality along with the CDO sources required and a sample PDI file with these CDOs.

1. Create a PetaLinux project with the 2022.2 release BSP.
2. Use the default BIF file from the PetaLinux project. This file can be referenced from the following location: `/hardware/xilinx-vck190-2022.2/xilinx-vck190-2022.2.runs/impl_1/project_1_wrapper.bif`.
3. Copy all the required CDO, RCDO, RNPI, and ELF files referenced in the above BIF file from different locations within the PetaLinux project folder.
4. Modify the BIF file to reference the following CDO files (A sample BIF file is provided in the reference design file accompanying this application note):

```
partition
{
id = 0x09
type = pmcdata, load = 0xf2000000
file = gen_files/pmc_data.cdo
//
// Add the following CDO files
//
file = gen_files/board_cdo.src
file = gen_files/temp_DVS_cdo.src
```

5. Generate the PDI file after including the above modified CDO files in the BIF file using the Bootgen utility. Board CDO and Temp_DVS CDO should appear after `pmc_data.cdo` in the BIF file mentioned below.

```
bootgen -arch versal -padimageheader=0 -log trace -w -o BOOT.BIN -image boot.bif
```

Test Steps

Ready to test BOOT.BIN and the required files to generate the PDI are found in the reference design file. The DVS functionality test steps are as follows:

1. Set up the VCK190 board with JTAG and serial cables connected.
2. Set the boot mode to JTAG.
3. Upon system controller boot, connect to the SYSCTL com2 port and run the following command before programming PDI: `sc_app -c getvoltage -t VCCINT`.
4. Program the PDI file with XSDB using the following steps:
   a. `xsdb% connect`
   b. `xsdb% device program BOOT.BIN`
5. Open the SYSCTL COM2 port. You can see the updated value in VCCINT voltage upon running the command: `sc_app -c getvoltage -t VCCINT`.

DVS Checklist

Download the reference design files for this application note from the Xilinx website.The following checklist summarizes the hardware and software changes required to support DVS.

Table 4: **DVS Checklist**

Parameter	Description
General	
Is DVS required?	DVS is required for AMD Versal™ -2LI SKUs.
Is the recommended PLM/AMD drivers solution used?	The AMD verified solution contained within the PLM should be used.
Is the recommended -2LI rail consolidation used?	This rail consolidation breaks out the VCCINT rail because its voltage is unique to that rail and cannot be consolidated with any other voltages.
Hardware	
I2C/PMBus compatible voltage regulator used?	I2C/PMBus is required to change the voltage on VCCINT.
Are there other primary I2C/PMBus devices on the same bus?	The DVS drivers require primary capabilities to drive the clocks on the bus and signal when the voltage change is required.
Default/start-up voltage set to 0.725V?	This must be the default voltage on power-up.
Software	
Synthesis software tools/versions used	AMD Vivado™ Design Suite 2022.2
Is the software driver implemented?	Functionality tested using baremetal driver
Any platform software changes needed?	CDO regulator changes are required before generating the PDI.

Conclusion

This application note shows how DVS is implemented on AMD Versal™ devices with examples in hardware and software that can be easily implemented by following the steps described. DVS is an important step to support the latest high-performing and competitive solutions in the market and helps further drive innovation.

References

These documents provide supplemental material useful with this application note:

1. *PetaLinux Tools Documentation: Reference Guide* (UG1144)
2. *Versal Adaptive SoC System Software Developers Guide* (UG1304)

Revision History

The following table shows the revision history for this document.

Section	Revision Summary
06/14/2023 Version 1.0	
Initial release.	N/A

Please Read: Important Legal Notices