XAPP1218 (v1.0) August 29, 2014

AXI VDMA Reference Design for the Kintex KC705 Evaluation Board

Author: Dinesh Kumar

Summary

This application note demonstrates how to use the LogiCORE™ IP AXI Video DMA (VDMA) core in a typical video application. It describes hardware and software APIs. The hardware system uses two VDMAs with both MM2S and S2MM paths shorted.

You can refer to this design and use the API in video applications that use VDMA. This application note demonstrates VDMA in loopback mode, but the MM2S and S2MM can also each be connected to separate video IP.

You can download the Reference Design Files for this application note from the Xilinx® website. For detailed information about the design files, see Reference Design.

Included Systems

The reference design is created and built using Vivado® IP Integrator v2014.1, which is part of Vivado System Edition. The IP integrator is an interactive design and verification environment that you can use to create and verify a hierarchical system by graphically connecting Xilinx, third-party, or proprietary IP, using interface-level connections. It provides a device- and platform-aware, interactive environment that supports intelligent auto-connection of key IP interfaces, one-click IP subsystem generation, real-time DRCs, interface change propagation, and powerful debug capability.

The design also includes software built using the Xilinx Software Development Kit (SDK). The software runs on the MicroBlaze™ processor subsystem and implements control, status, and monitoring functions. Complete Vivado tools and SDK project files are provided with the reference design to allow you to examine and rebuild the design or to use as a template for a new design.

Introduction

The Xilinx AXI VDMA IP core implements a high-performance, video-optimized DMA engine with frame buffering and two-dimensional (2D) DMA features. Together, the AXI interconnect and AXI MIG implement a multiport memory controller (MPMC) for sharing data from multiple sources through a common memory device, typically DDR3 SDRAM. The AXI VDMA transfers the buffered video data streams to and from memory and operates under dynamic software control. A clock generator and processor system reset block supplies clocks and resets throughout the system.

Xilinx IP cores implement various functions for many video applications. AXI is a standardized IP interface protocol based on the Advanced Microcontroller Bus Architecture (AMBA®) specification. The AXI interfaces used in the reference design consist of AXI4, AXI4-Lite, and AXI4-Stream interfaces as described in the AMBA AXI4 specifications. These interfaces provide a common IP interface protocol framework for building the design.

The Kintex®-7 FPGA KC705 evaluation kit provides a comprehensive, high-performance development and demonstration platform using the Kintex-7 FPGA family for high-bandwidth and high-performance applications in multiple market segments. For more information about this evaluation board, see the *Kintex-7 FPGA KC705 Evaluation Kit Getting Started Guide* [Ref 2].

Setup Requirements

Following are the hardware and software requirements.

Software

Vivado Design Suite 2014.1

Hardware

Kintex-7 FPGA KC705 Evaluation Kit Base Board (Figure 1):

1. One KC705 board with power adapters
2. Two USB platform cables (1 mini and 1 micro)

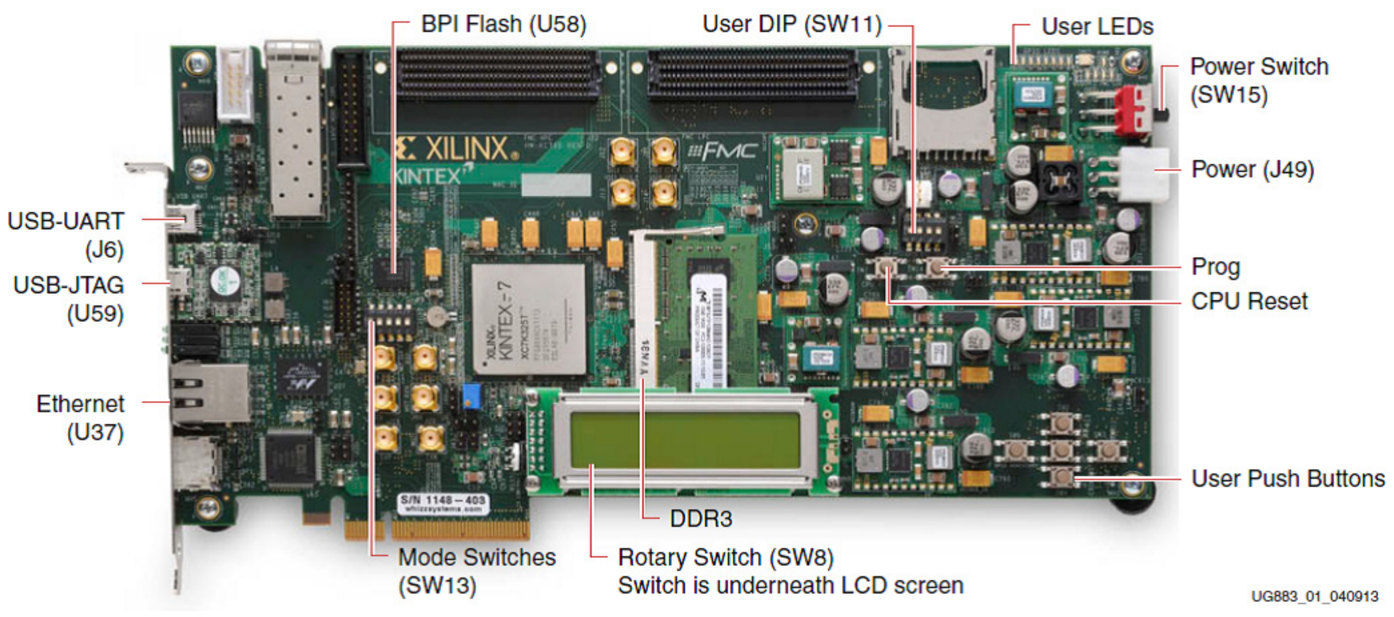


Figure 1: **KC705 Evaluation Kit**

Reference Design Specifics

Hardware

In addition to a MicroBlaze processor, the reference design include following cores:

- MDM
- LMB block RAM
- AXI_INTERCONNECT
- Clock Generator
- PROC_SYS_RESET
- AXI_UARTLITE
- AXI_INTC
- Memory Interface Generator (MIG)
- Video Direct Memory Access (VDMA)

Figure 2 shows an IP integrator block diagram of the reference system.

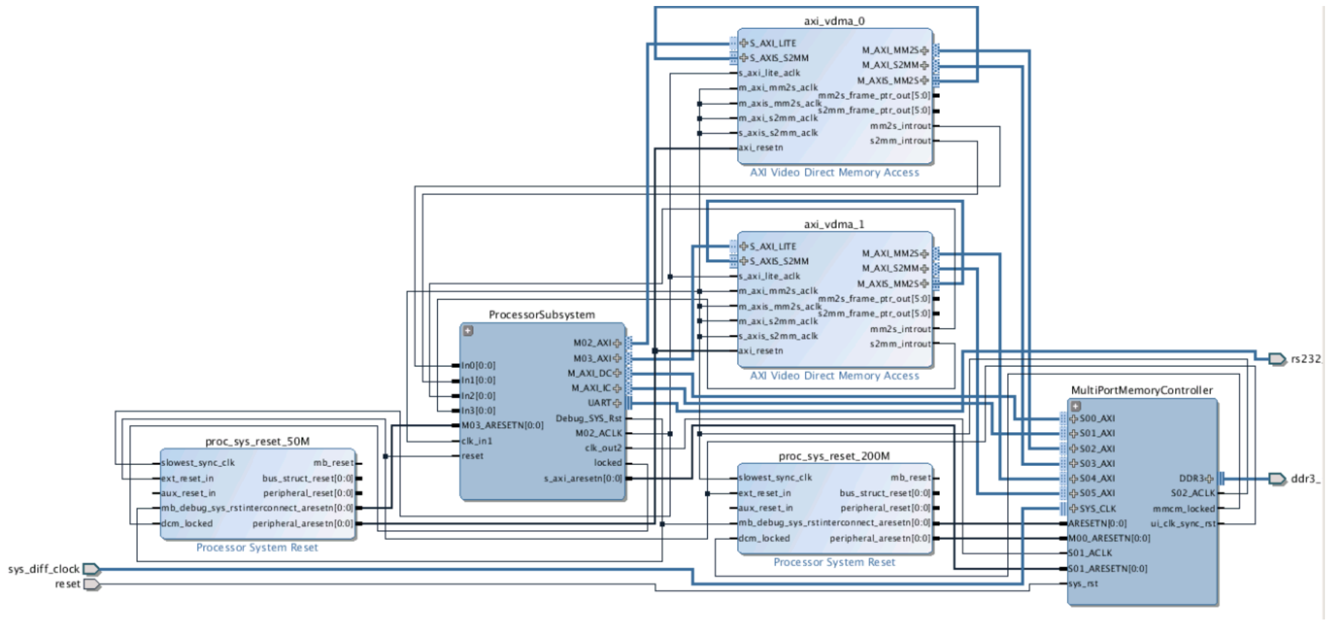


Figure 2: **IP Integrator Block Diagram**

The block diagram shows the system divided into the following modular blocks:

- Multiport memory controller
- Processor subsystem
- Two VDMAs
- Two processor system resets

Multiport Memory Controller

The multiport memory controller consist of an AXI interconnect and a MIG system (Figure 3). The frequency and data width of the MIG and AXI interconnect are set to meet the bandwidth requirement of a typical video application.

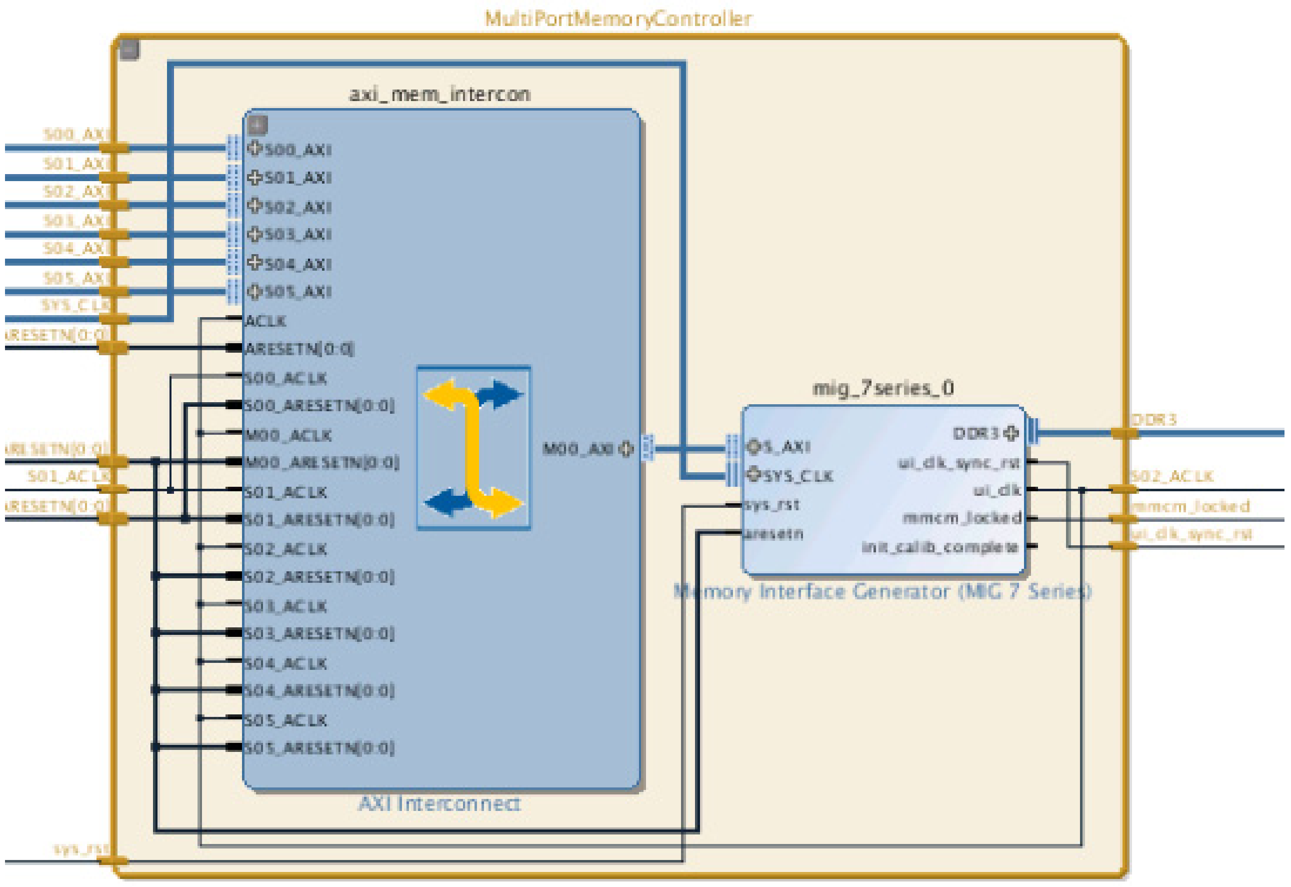


Figure 3: **Multiport Memory Controller**

Processor Subsystem

The processor subsystem consists of a MicroBlaze processor and the IP required for the system to function correctly (Figure 4). An AXI interconnect connects peripherals including local memory, UART Lite, and the interrupt controller to the processor. It also includes a clocking block and a microprocessor debug module (MDM) to enable downloading and debugging through the Xilinx microprocessor debugger (XMD). A reset block manages reset synchronization.

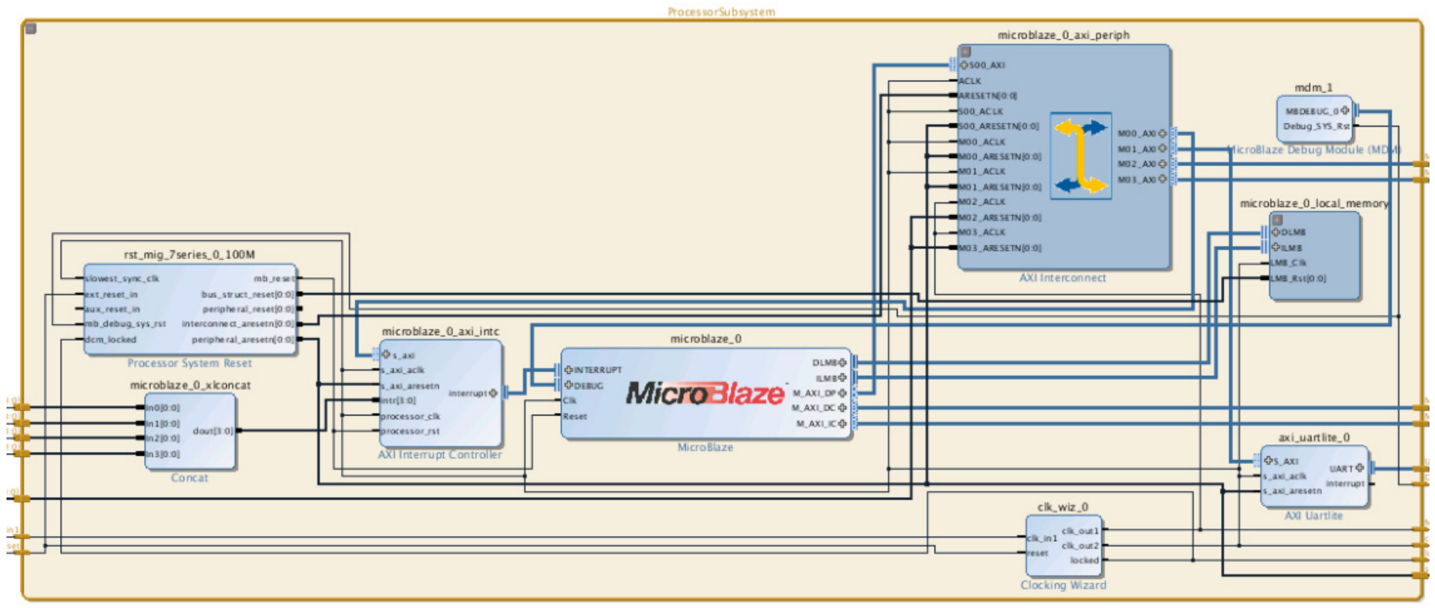


Figure 4: **Processor Subsystem**

VDMAs

The system uses two VDMAs. The MM2S path of one VDMA is connected to its S2MM path. The VDMAs are configured to allow programming of a frame counter and frame counter interrupt by setting C_ENABLE_DEBUG_INFO_7 and C_ENABLE_DEBUG_INFO_15. The VDMA is configured according to the following specifications:

- 200 MHz AXIS clocks
- 200 MHz AXI clocks
- 50 MHz AXI Lite clock
- 1920 by 1080 frame size
- Three frame buffers in circular mode
- S2MM in SOF on TUSER mode, and MM2S in free-run mode
- Dynamic genlock mode with the S2MM side as the master and MM2S as the slave
- Tdata widths of 24 bits
- MM data width of 32 bits

- DRE disabled; accesses must be properly aligned
- Error interrupts optionally enabled in the software application
- Frame count interrupt enabled to assert every frame in the software application

Using two VDMAs shows that the APIs used in the design are modular and that the same API can be called repeatedly to configure each VDMA.

Processor System Reset

Processor system reset blocks are used to synchronize resets across various clocks used in system.

Software

The software package consists of BSP, `system.xml` files, and the following application files.

vdma_api.c

The main file, consisting of APIs to initialize, configure, and start VDMA.

The API prototype is:

```
int run_triple_frame_buffer(XAxiVdma* InstancePtr, int DeviceId, int
hsize, int vsize, int buf_base_addr, int number_frame_count, int
enable_frm_cnt_intr);
```

- `instancePtr` - AXI VDMA driver instance pointer.
- `DeviceId` - Device ID for VDMA to allow the function to work for multiple VDMAs in a system.
- `hsize` - Horizontal size in pixels. The actual data transfer is (`hsize`) x (`tdatawidth`).
- `vsize` - Vertical size in lines.
- `number_frame_count` - The number of frames after which the application expects an interrupt from the VDMA.
- `enable_frm_cnt_intr` - Enable frame count interrupts.

The API returns `XST_SUCCESS` or `XST_FAILURE` based on the configuration of the VDMA.

vdma_api.h

The file containing the prototype of the API; can be added in the application.

vdma.c

Consists of the VDMA application to demonstrate how to use the VDMA triple buffer API.

vdma.h

Includes various files required for the application.

Executing the System on the Board

Creating the Bit File and ELF File

The package consists of `all.tcl` and the software. Complete the following steps to create the bit file and elf file:

1. Unzip the folder to the desired directory.
2. Launch Vivado Design Suite.
3. Source `all.tcl` in the Vivado IDE Tcl Console. This creates the IP integrator block diagram.
4. Generate the bitstream by clicking **Generate Bitstream** on the left side of the Vivado IDE. This creates the file `design_1_wrapper.bit` file at `./project_1/project_1.runs/impl_1`.
5. Export the design:
   a. Click **File > Export > Export Hardware for SDK**.
   b. Select **Export Hardware** and **Launch SDK**.
6. The SDK opens with the created project and compiles the software. If you receive an error or warning from the SDK, click **Yes** or **OK**.
7. In the SDK, select **Project > Clean** to clean the pre-generated file. The file `app.elf` is created in the `./sw/app/Debug` folder.

Running the System

1. Open the hyper-terminal (serial port) for baud rate of 9600.
2. Start the Xilinx Microprocessor Debugger (XMD) tool:

```
% xmd
```

3. Download the bitstream inside XMD:

```
XMD% fpga -f design_1_wrapper.bit
```

4. Connect to the processor inside XMD:

```
XMD% connect mb mdm
```

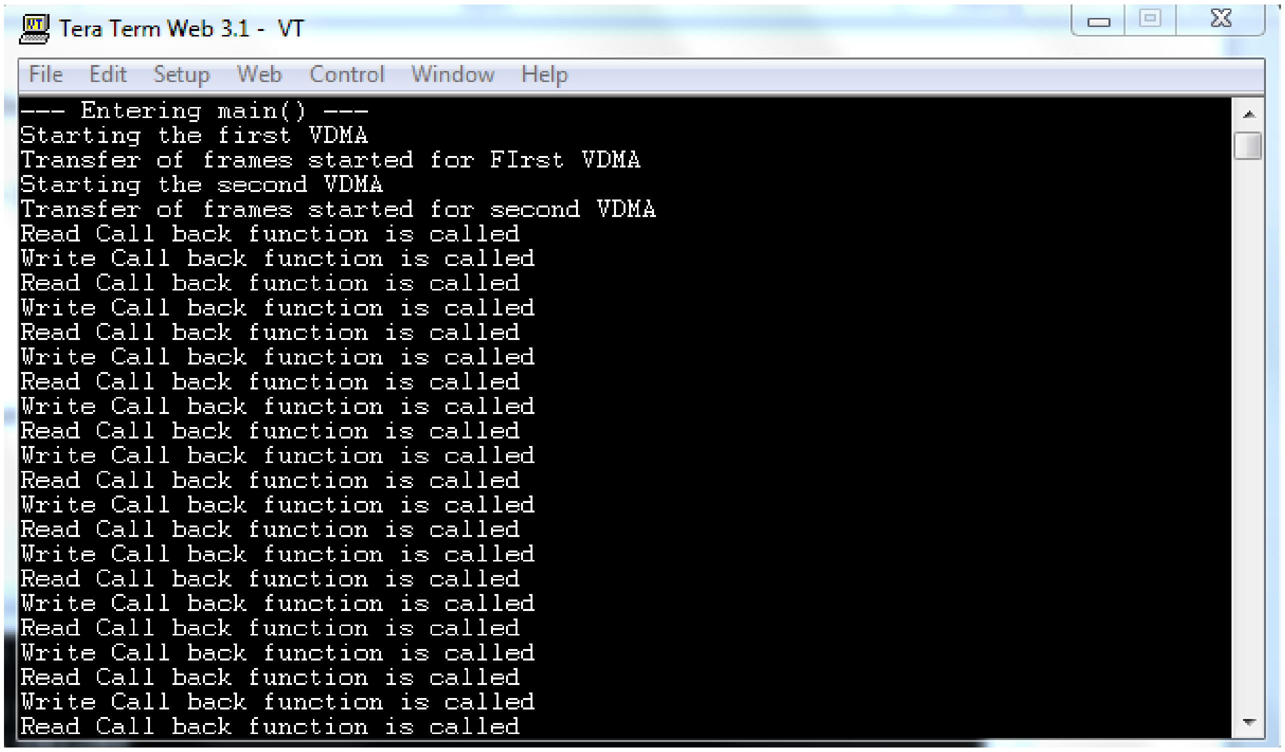
5. Download the processor code (ELF) file:

```
XMD% dow app.elf
```

6. Run the application:

```
XMD% run
```

The results are shown in Figure 5.



```
--- Entering main() ---
Starting the first VDMA
Transfer of frames started for FIrst VDMA
Starting the second VDMA
Transfer of frames started for second VDMA
Read Call back function is called
Write Call back function is called
Read Call back function is called
Write Call back function is called
Read Call back function is called
Write Call back function is called
Read Call back function is called
Write Call back function is called
Read Call back function is called
Write Call back function is called
Read Call back function is called
Write Call back function is called
Read Call back function is called
Write Call back function is called
Read Call back function is called
Write Call back function is called
Read Call back function is called
Write Call back function is called
Read Call back function is called
Write Call back function is called
Read Call back function is called
```

Figure 5: **Application Run Output**

The multiple callbacks indicate that frame counter interrupt occurs for read and write channels. Add code in these callbacks to create a complete system.

Reference Design

The reference design has been fully verified and tested on hardware. The design includes details on the various functions of the different modules and includes all functions required to implement a complete VDMA design. You can download the Reference Design Files for this application note from the Xilinx website.

Table 1 shows the reference design matrix.

Table 1: **Reference Design Matrix**

Parameter	Description
General	
Developer name	Dinesh Kumar
Target devices (stepping level, ES, production, speed grades)	Kintex7 (KC705) FPGA
Source code provided	Yes
Source code format	VHDL/Verilog (some sources encrypted)
Design uses code/IP from existing Xilinx application note/reference designs, CORE Generator software, or 3rd-party	Reference designs provided uses core generated from Vivado 2014.1
Implementation	
Synthesis software tools/version used	Vivado 2014.1
Implementation software tools/versions used	Vivado Design Suite 2014.1
Static timing analysis performed	Y (passing timing in)
Hardware Verification	
Hardware verified	Y
Hardware platform used for verification	KC705 board

References

1. AMBA AXI4 specifications
2. *Kintex-7 FPGA KC705 Evaluation Kit Getting Started Guide* (UG883)
3. *LogiCORE IP AXI Interconnect Product Guide* (PG059)
4. *LogiCORE IP AXI Video Direct Memory Access Product Guide* (PG020)
5. *7 Series FPGAs Memory Interface Solutions* (DS176)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
08/29/2014	1.0	Initial Xilinx release.

Please Read: Important Legal Notices