FPGA-Based System Combines Two Video Streams to Provide 3D Video

By Witold Kaczurba

Introduction

Video systems, already ubiquitous in consumer applications, are increasingly prevalent in automotive, robotics, and industrial domains. This growth into nonconsumer applications resulted primarily from the introduction of an HDMI standard and faster, more efficient DSPs and FPGAs.

This article outlines the requirements for achieving stereoscopic vision (3D video) using analog or HDMI video cameras. It describes an FPGA-based system that combines two video streams into a single 3D video stream for transmission through an HDMI 1.4 transmitter, and a DSP-based system that saves DMA bandwidth compared to that normally required for receiving data from two cameras. Furthermore, it shows one method for achieving a side-by-side format for use with 3D cameras or systems requiring 3D video.

General Overview

Stereoscopic vision requires two video cameras separated by approximately 5.5 cm, the typical spacing between a person's eyes, as shown in Figure 1.

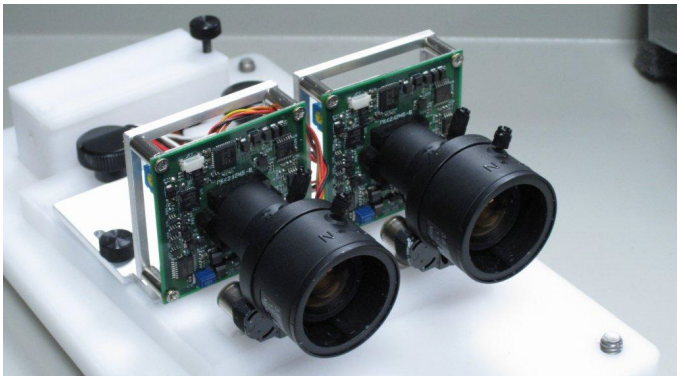

Figure 1. Two cameras on a stand aligned for stereoscopic vision.

The high-level block diagram shown in Figure 2 uses two synchronized video cameras that use the same video standard, two video decoders, and an FPGA. To ensure the exact same frame rate, the video cameras must be line-locked to a common timing reference. Without synchronization, it will not be possible to combine the outputs without using external memory to store complete video frames.

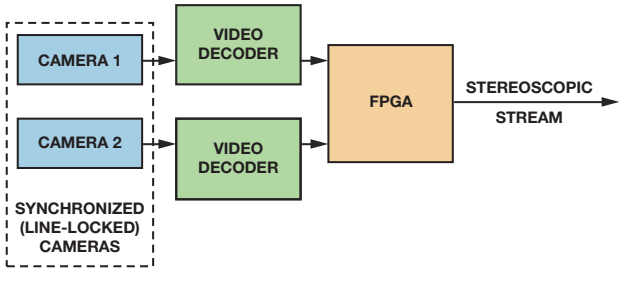

Figure 2. High-level block diagram.

Figure 3 shows two line-locked video streams being merged into a single stereoscopic image. Figure 4 shows how asynchronous video streams cannot be merged without saving the entire video frame in an external memory.

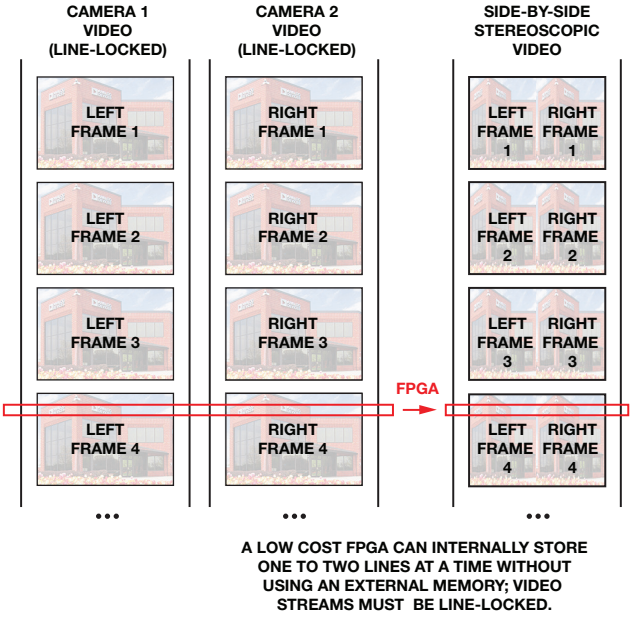

Figure 3. Merging two synchronized video streams.

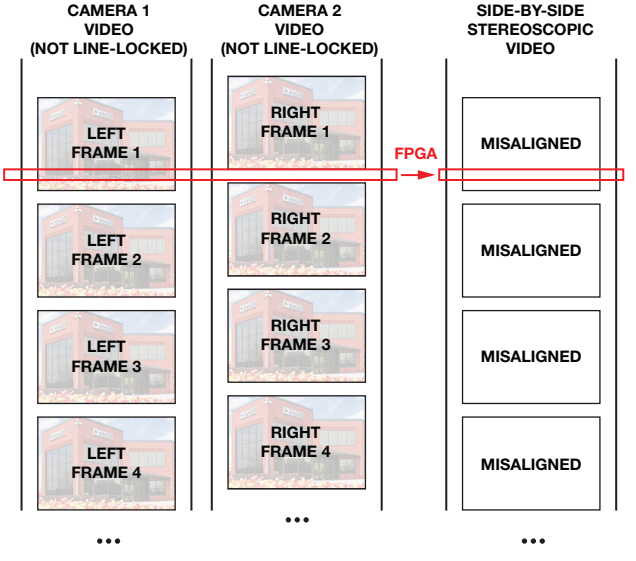

Figure 4. Asynchronous video streams cannot be merged without using an external memory.

The outputs of the two synchronized video cameras are then digitized by video decoders such as the ADV7181D, ADV7182, or ADV7186 for analog video cameras; or by HDMI receivers such as the ADV7610 or ADV7611 with digital video cameras.

Video decoders and HDMI receivers use internal phase-locked loops (PLLs) to produce clock and pixel data at their output buses. This means that two separate clock domains will be generated for the two cameras when digitizing the analog video or receiving the HDMI stream. Moreover, the two video streams can be misaligned. These timing differences and misalignments must be compensated in a back-end device such as an FPGA, bringing the data to a common clock domain before combining the two video pictures into a single stereoscopic video frame. The synchronized video stream is then sent through an HDMI 1.4 3D-capable HDMI transmitter such as the ADV7511 or ADV7513—or it can be presented to a DSP such as the ADSP-BF609 Blackfin® processor—for further processing.

Clocking Architectures

Video decoders have two distinct clocking sources depending upon whether they are locked or unlocked. When the video PLL is locked to the incoming synchronization signal—horizontal sync for video decoders or the TMDS clock for HDMI—it generates a clock that is locked to the incoming video source. When video lock is lost, or the PLL is in forced free-run mode, the video PLL is not locked to the incoming synchronization signal and it generates a clock output that is locked to the crystal clock. In addition, the clock may not be output after reset as the LLC clock driver is set to a high impedance mode after reset.

Thus, if the system has two or more video paths from the video decoder or HDMI receiver, it will have two different clock domains with different frequencies and phases, even when the same crystal clock is provided to two video decoders or HDMI receivers, as each device generates its own clock based on its own PLL.

Synchronous System with Locked Video Decoders

With typical stereoscopic video using two sources, each of the video decoders locks to the incoming video signal and generates its own clock based on incoming horizontal sync or TMDS clock. When two cameras are synchronized—or line-locked to the same timing reference—the frame lines will always be aligned. Because the two separate video decoders receive the same horizontal sync, the pixel clocks will have the same pixel clock frequency. This allows for bringing the two data paths into a common clock domain, as shown in Figure 5.

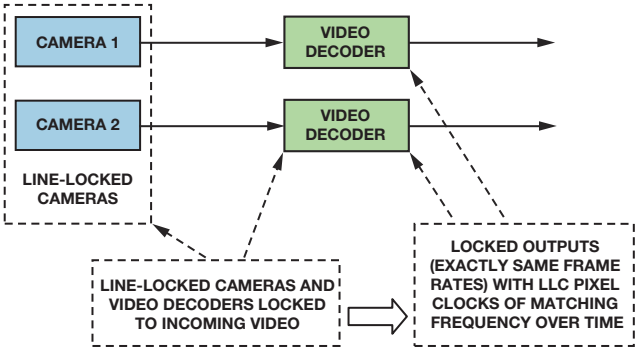

Figure 5. Two video cameras synchronized to a common reference. Both video decoders receive the same sync signal, so they are also locked.

Asynchronous Video System

Unfortunately, one of the decoders may lose lock due to a poor quality video source signal, as shown in Figure 6; or the cameras may lose synchronization due to a broken video link, as shown in Figure 7. This will lead to different frequencies in the two data paths, which will then lead to asymmetry in the amount of data clocked into the back end.

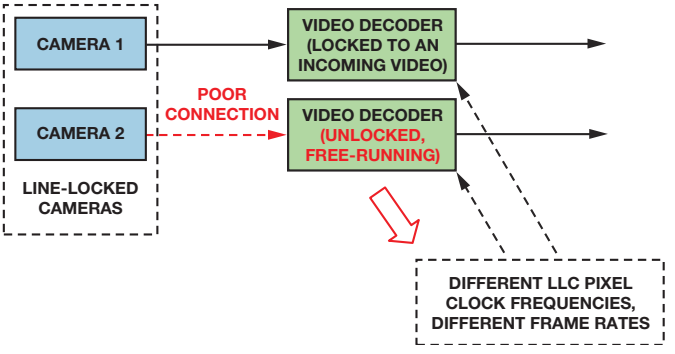

Figure 6. Line-locked cameras with unlocked video decoders.

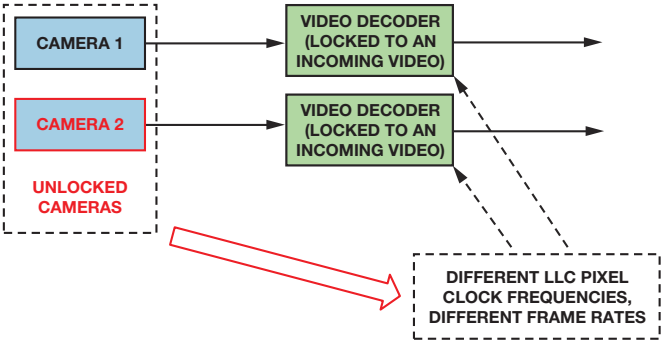

Figure 7. Unlocked cameras with locked video decoder.

Lost video lock can be detected by using an interrupt (SD_UNLOCK for SD video decoders, CP_UNLOCK for component video decoders, or TMDSPLL_LCK registers in HDMI receivers) that kicks in after a delay. Video decoders integrate mechanisms for smoothing unstable horizontal synchronization, so detection of lost video lock can take up to a couple of lines. This delay can be reduced by controlling lost lock within the FPGA.

Clock Tri-State Mode

When designing FPGA clocking resources, it is important to know that by default, many video decoders and HDMI products put the clock and data lines into tri-state mode after reset. Thus, the LLC pixel clock will not be suitable for synchronous resets.

Data Misalignment in Two Video Streams

To simplify the system and reduce the memory needed to combine the two pictures, data reaching the FPGA should be synchronized such that the N^{th} pixel of the M^{th} line from the first camera is received with the N^{th} pixel of the M^{th} line from the second camera.

This might be difficult to achieve at the input of the FPGA because the two video paths may have different latencies: line-locked cameras can output misaligned lines, different connection lengths can contribute to misalignment, and video decoders can introduce variable startup latencies. Because of these latencies it is expected that a system with line-locked cameras will have a number of pixels of misalignment.

Line-Locked Camera Misalignment

Even line-locked cameras can output misaligned video lines. Figure 8 shows the vertical sync signals from the CVBS output of two cameras. One camera, the sync master, provides a line-locking signal to a second camera, the sync slave. Misalignment of 380 ns is clearly visible. Figure 9 shows the data transmitted by the video decoders on the outputs of these cameras. An 11-pixel shift can be seen.

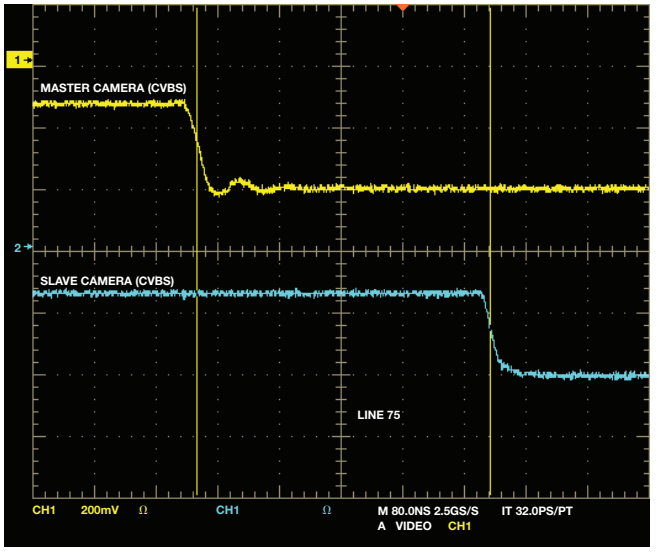

Figure 8. 380-ns video misalignment between line-locked video cameras.

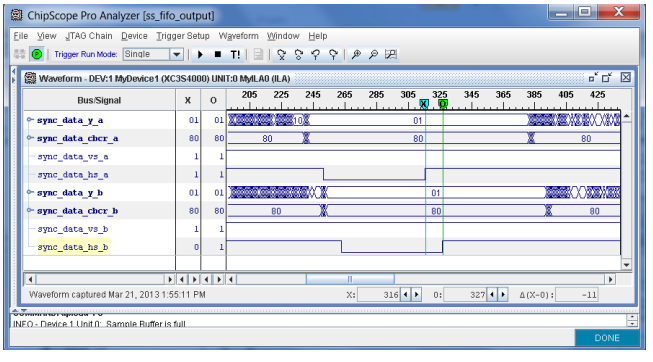

Figure 9. Uncompensated 11-pixel video misalignment in the digital domain.

Different Connection Lengths

All electrical connections introduce a propagation delay, so make sure that both video paths have the same track and cable lengths.

Video Decoder/HDMI Receiver Latencies

All video decoders introduce latency that can vary depending on the enabled features. Moreover, some video parts contain elements—such as a deep-color FIFO—that can add random startup latency. A typical stereoscopic system using video decoders may have a random startup delay of around 5 pixel clocks. A system containing HDMI transmitters and receivers, as shown in Figure 10, may have a random startup delay of around 40 pixel clocks.

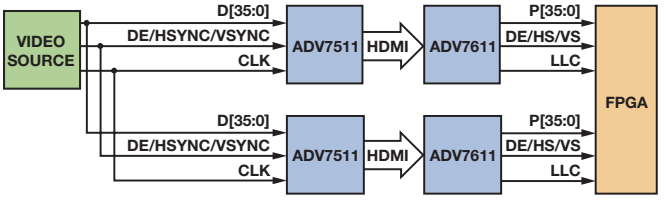

Figure 10. Pipeline delays measurement setup.

Misalignment Compensation

Figure 11 shows a system where an analog signal from each camera is digitized by a video decoder. The data and clock are separate for each video path. Both video paths are connected to FIFOs, which buffer the incoming data to compensate for data misalignment. When clocking out the data, the FIFOs use a common clock from one of the decoders. In a locked system, the two data paths should have exactly the same clock frequency, ensuring that no FIFO overflows or underflows as long as the cameras are line-locked and the video decoders are locked.

By enabling or disabling FIFOs outputs, the control block maintains FIFO levels to minimize pixel misalignment. If compensation is carried out properly, the output of the FPGA block should be two data paths aligned to the very first pixel. That data is then supplied to an FPGA back end for 3D format production.

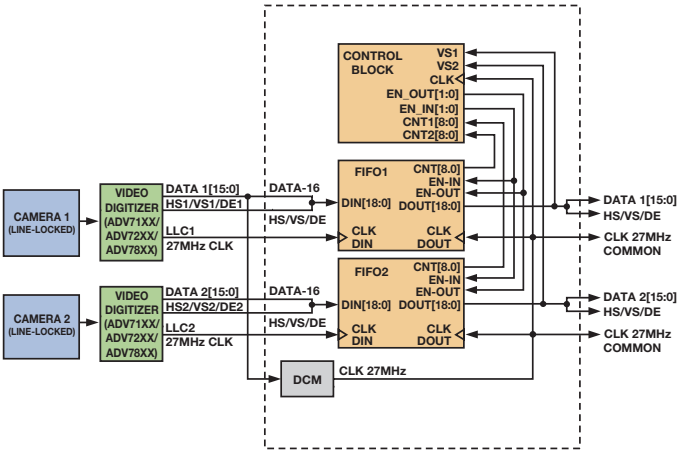

Figure 11. Using digital FIFOs to realign video pictures.

Misalignment Measurement

Misalignment between two digitized data streams can be measured at the output of the video FIFOs by using a one-clock counter that is reset on the vertical sync (VS) pulse of one of the incoming signals. Figure 12 shows two video streams (vs_a_in and vs_b_in) misaligned by 4 pixels. Counters measure the misalignment using the method shown in Listing 1. Counting starts on the rising edge of VS1 and stops on the rising edge of VS2.

If the total pixel length of a frame is known, the negative skew (VS2 preceding VS1) can be calculated by subtracting the count value from the length of frame. This negative value should be calculated when the skew exceeds half of the pixel frame length. The result should be used to realign the data stored in the FIFOs.

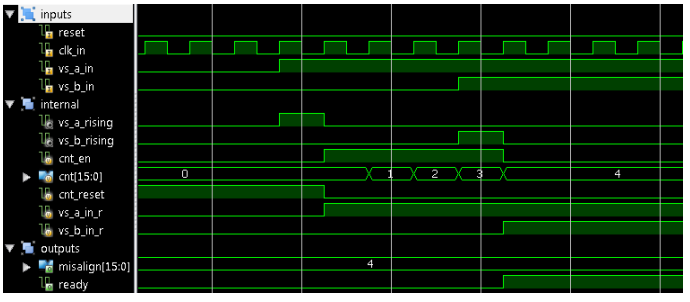

Figure 12. Misalignment measurement.

Listing 1. Simple misalignment measurement (Verilog®).

```
module misalign_measurement(
 input wire reset,
 input wire clk_in,
 input wire vs_a_in,
 input wire vs_b_in,
 output reg [15:0] misalign,
 output reg ready);

reg [15:0] cnt;
reg cnt_en, cnt_reset;
reg vs_a_in_r, vs_b_in_r;
assign vs_a_rising = vs_a_in > vs_a_in_r;
assign vs_b_rising = vs_b_in > vs_b_in_r;

always @(posedge clk_in)
begin
 vs_a_in_r <= vs_a_in;
 vs_b_in_r <= vs_b_in;
end

always @(posedge clk_in)
 if (reset)
 begin
 { ready, cnt_en } <= 2'b00;
 misalign <= 0;
 end else begin
 if ((vs_a_in == 1'b0) && (vs_b_in == 1'b0))
 { ready, cnt_reset } <= 2'b01;
 else
 cnt_reset <= 1'b0;
 /* beginning */
 if (vs_a_rising && vs_b_rising)
 begin
 misalign <= 0;
 { ready, cnt_en } <= 2'b10;
 end
 else if ((vs_a_rising > vs_b_in) || (vs_b_
 rising > vs_a_in))
 { ready, cnt_en } <= 2'b01;

 /* ending */
 if ((cnt_en == 1'b1) && (vs_a_rising || vs_b_
 rising))
 begin
 { ready, cnt_en } <= 2'b10;
  misalign <= vs_a_rising ? (-(cnt + 1)) : (cnt + 1);
 end
 end

always @(posedge clk_in) /* counter */
 if ((cnt_reset) || (reset))

 cnt <= 0;
 else if (cnt_en)
 cnt <= cnt + 1;

endmodule
```

Production of 3D Video from Two Aligned Video Streams

Once pixel, line, and frame data are truly synchronous, an FPGA can form the video data into a 3D video stream, as shown in Figure 13.

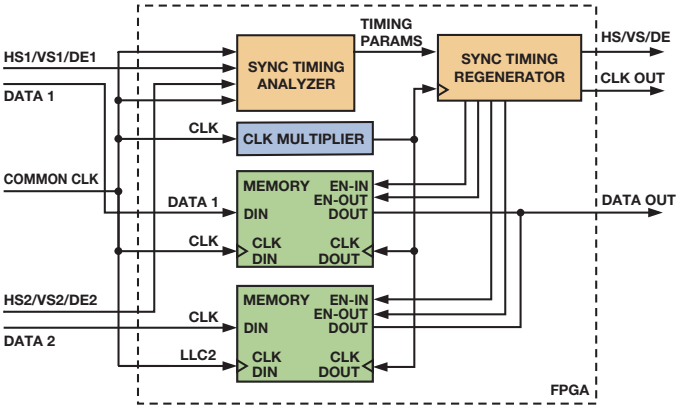

Figure 13. Simplified architecture that achieves 3D formats.

The incoming data is read into memory by a common clock. The sync timing analyzer examines the incoming synchronization signals and extracts the video timing, including horizontal front and back porch lengths, vertical front and back porches, horizontal and vertical sync length, horizontal active line length, the number of vertical active lines, and polarization of sync signals. Passing this information to the sync timing regenerator along with the current horizontal and vertical pixel location allows it to generate timing that has been modified to accommodate the desired 3D video structure. The newly created timing should be delayed to ensure that the FIFOs contain the required amount of data.

Side-by-Side 3D Video

The least demanding architecture in terms of memory is the side-by-side format, which requires only a 2-line buffer (FIFOs) to store content of lines coming from both video sources. The side-by-side format should be twice as wide as the original incoming format. To achieve that, a doubled clock should be used for clocking the regenerated sync timing with doubled horizontal line length. The doubled clock used for clocking the back end will empty the first FIFO and then the second FIFO at a double rate, allowing it to put pictures side-by-side, as shown in Figure 14. The side-by-side picture is shown in Figure 15.

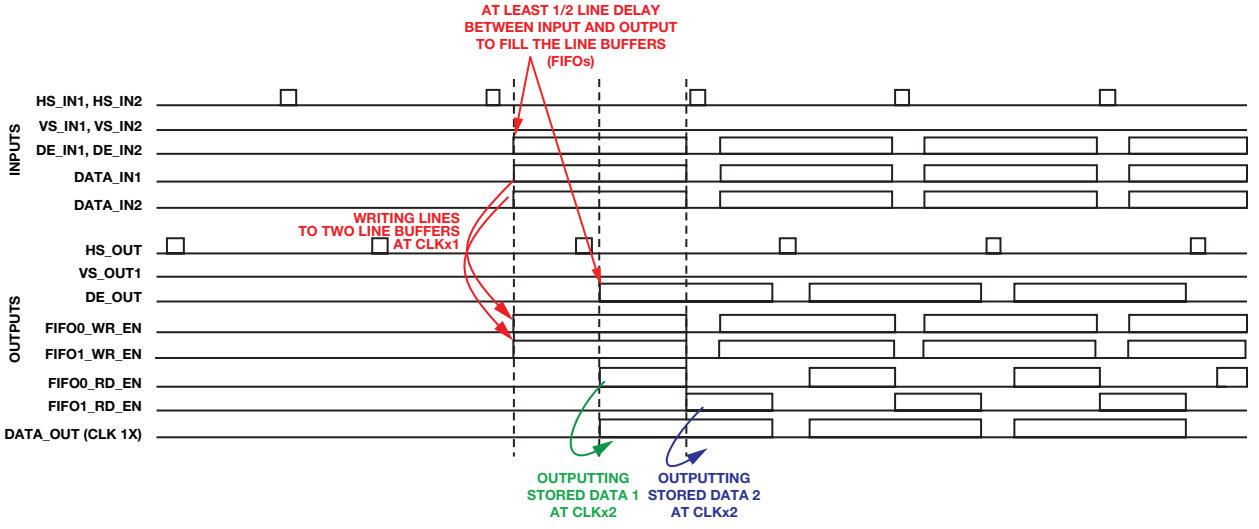

Figure 14. Stitching two pictures side-by-side using simple FPGA line buffers.

Figure 15. Side-by-side 576p picture with video timings

Conclusion

Analog Devices decoders and HDMI products along with simple postprocessing can create and enable the transmission of true stereoscopic 3D video. As shown, it is possible to achieve 3D video with simple digital blocks and without expensive memory. This system can be used in any type of system requiring 3D vision, from simple video recording cameras to specialized ADSP-BF609 DSP-based systems that can be used for tracking objects and their distances.

Author

Witold Kaczurba [witold.kaczurba@analog.com], a senior applications engineer in the Advanced TV group in Limerick, Ireland, supports decoders and HDMI products. He joined ADI in 2007 after graduating from the Technical University of Wroclaw, Poland, with an MSc in electrical engineering. As a student, he worked for small electronic and IT companies, then joined ADI in Ireland as a co-op student and subsequently as an applications engineer.