Bottling a Star Using the ARM® AMBA® AXI4 in an FPGA

By Billy Huang, PhD Researcher, Durham University / CCFE,
Dr. Roddy Vann, Assistant Professor, University of York,
Dr. Graham Naylor, Head of MAST Plasma Diagnostics and Control, Culham Centre for Fusion Energy (CCFE),
Dr. Vladimir Shevchenko, Senior Physicist, Culham Centre for Fusion Energy (CCFE),
Simon Freethy, PhD Researcher, University of York / CCFE

Fusion researchers in the U.K. demonstrate a data acquisition system for synthetic-aperture imaging using the latest ARM® AMBA™ AXI4 interface on Xilinx technology.

Fusion energy is the combining of hydrogen atoms into larger atoms at extremely high temperatures. It is how all the stars, including the sun, create energy. To generate fusion energy on Earth, we heat ionized hydrogen gas, known as "plasma" to over 100 million degrees kelvin in a magnetic bottle called a "tokamak" (see Figure 1).

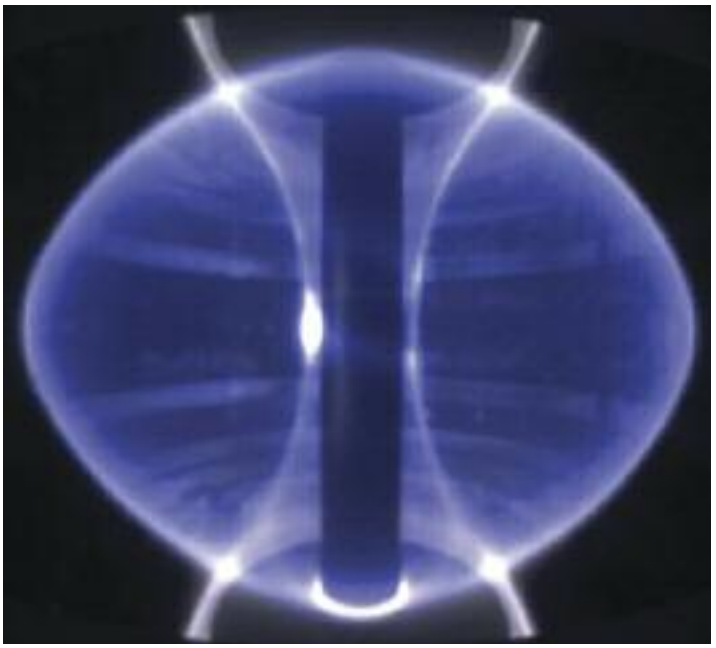


Figure 1: The Mega Amp Spherical Tokamak (MAST) at CCFE is uniquely shaped, more similar to a cored apple than the traditional doughnut. The clear view inside the device shows the "bottled star" within.

The end goal of fusion scientists like our team at the Culham Centre for Fusion Energy (CCFE) – a world-leading institution for the development of fusion energy near Oxford, England – is to create a fusion energy power station using hydrogen fuel that is readily available on Earth. In fact, there is enough fuel on Earth for fusion to supply our energy needs for more than a million years. The catch is that fusion is extremely difficult, just what you would expect when trying to bottle a star. The international ITER initiative, at $20 billion the world's largest terrestrial scientific project, will demonstrate fusion power on an industrial scale for the first time. Currently under construction in the south of France, ITER – the name means "the way" in Latin – expects to be in operation for two decades (see http://www.iter.org).

A key part of fusion research is the real-time measurement of the fusion plasma. Each diagnostic has its own requirements. At CCFE (http://www.ccfe.ac.uk), we have developed a diagnostic that images microwaves emitted from the plasma in order to measure the electrical current within it. For that purpose, we set out to design a synthetic-aperture imaging system.

Assessing Microwave Phases

Synthetic-aperture imaging uses phased arrays of antennas (see Figure 2) in a configuration that works similarly to your ears. If

Figure 2: The microwave-imaging phased antenna array utilizes novel PCB antennas.

you hear a noise to your right, the sound will reach your right ear sooner than your left. Another way of saying this is that the sound reaches your ears at a different phase. Your brain interprets the phase difference as a direction. In much the same way, by looking at the phase of microwaves an antenna array has detected, you can determine where they were emitted from. We recombine a picture of the plasma edge from a phased antenna array that uses this principle.

The radio frequency (RF) system (see Figure 3) downconverts

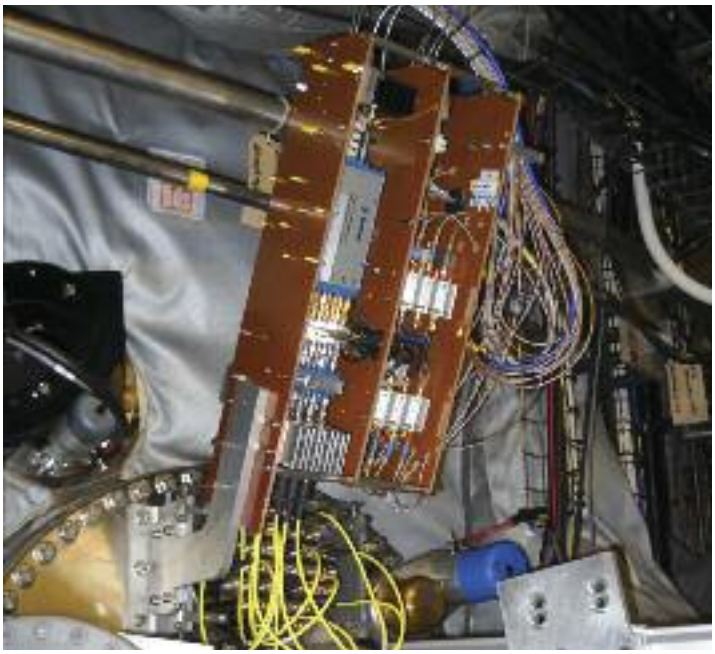


Figure 3: The RF electronics connected to the MAST tokamak downconvert the incoming 6 to 40 GHz signal to the 250 MHz bandwidth signal the FPGA data acquisition box will process.

the signal at each antenna in frequency from 6 to 40 GHz, to the 250 MHz bandwidth signal that the FPGA data acquisition box will process. This 250 MHz bandwidth defines the clock requirement for the analog-to-digital converters (ADCs). We use eight antennas, giving 16 channels that need to be digitized (the factor of two resulting from resolving real and "imaginary" components – mathematically, those for which the signal is phase-shifted by 90 degrees).

The system had to acquire data continuously for 0.5 seconds from 16 analog channels at 14 bits at 250 MHz. Bit packing the 14 bits to 2 bytes gives us a requirement of 32 bytes * 0.25 Gbytes/s = 8 Gbytes/s. We needed to acquire 4 Gbytes of data in half a second, and wanted FPGA boards with the FPGA Mezzanine Card (FMC) interface for flexibility in the choice of ADC manufacturers and portability in the future. We also wanted the option of using our in-house-developed FMC digital I/O board.

We decided during the summer of 2010 that an ideal solution would be to use two Xilinx® Virtex®-6 LX240T ML605 boards combined with two FMC108 (eight-channel) ADC boards from 4DSP. At that time, 8 Gbytes/s was a gigantic data rate; in fact, it still is. We could have taken the approach of divide and conquer by using more FPGA boards and having fewer channels on each. However, that would have increased the cost and size of the system.

In fact, the technology to make this aspect of our design happen arrived around January 2011, when Xilinx released a revision of its ISE® design software supporting the ARM AMBA AXI4 interface protocol. Before this the hardware existed, but not the means to exploit it to its full potential.

Life Before AMBA AXI4

For our system needs, the DDR3 SDRAM memory must be accessible to the MicroBlaze™ processor that resides on the Virtex-6, so that Linux can also have access to the real-time data we are capturing. This requirement constrains us to using a memory controller jointly accessible to the MicroBlaze bus and our real-time streaming IP. Initially we tried using the PLB bus, but found that limitations in the PLB-based memory controller meant we could not connect a 64-bit wide interface at our required frequency. Instead, only 32 bits were available. We learned this the hard way, after writing a core that communicated via the low-level NPI protocol directly to the memory controller, but could achieve only 2 Gbytes/s. Even though this was still an impressive rate and smashed any speed records we could find, it still was not enough.

Thankfully, ARM then pushed out the AMBA AXI4 interconnect and memory controller, giving full access to the whole 64 bits at 400 MHz double data rate (800 million transactions per second). That effectively gave a throughput of 6.4 Gbytes/s – a truly blistering speed that exceeded our requirement of 4 Gbytes/s on each board. This was exactly what we needed.

We actually found two ways to achieve this speed: one is a modification of the AXI_v6_ddrx memory controller (hidden under the AXI interconnect layer), and the other is an AMBA AXI Master PCore made in System Generator. The PCore can attach to the MicroBlaze system in Xilinx Platform Studio (XPS) as an AMBA AXI External Master.

Both solutions stream data into the DDR3 memory at 5 Gbytes/s. AMBA AXI is easy to program, and allows very high memory speeds with separate read and write channels. The XPS tool gives a lot of flexibility for AMBA AXI design. We used that flexibility to our advantage, such as choosing only a write channel if that was what we needed, thereby simplifying the logic design and freeing more resources.

A Soft-processor Interface

A unique capability of the Xilinx tool set is the soft processor known as the MicroBlaze. It is "soft" in that it is enabled using FPGA logic.

This capability has meant that we can have a PC-like interface to the FPGA system. This is enables, for example, Web and SSH servers on the FPGA.

Network Streaming

Given that we could acquire 2 Gbytes on each FPGA board in half a second, the question we found ourselves facing was how to get this data off the board over a standard interface in a reasonable amount of time? Typical network speeds using the MicroBlaze processor over Gigabit Ethernet under Linux and a simple protocol such as UDP proved too slow, achieving only around 0.5 Mbyte/s. At that rate we would have to wait over an hour to download data that had taken only half a second to acquire!

Clearly, we needed to go to a lower level in the design. Our solution took the form of a homegrown protocol we have dubbed FireStark, a UDP-based protocol that sits inside the AMBA AXI Ethernet DMA driver. By modifying the MicroBlaze Linux kernel drivers and having the FPGAs on a dedicated private network, we are now able to download the entire 2 Gbytes in under 60 seconds, a factor-of-70 speed-up. Testing with jumbo frames of sizes up to 6 kbytes has doubled this speed - that is, more than 70 Mbytes/s. Crucially, it shows that with DMA even the relatively slow MicroBlaze clock of 100 MHz is capable of high memory-to-network streaming throughput.

The latency measurement from the FPGA to the PC was 129 µs +/- 13 µs. (The real latency is even lower, since this measurement includes the latency overhead of the packet traversing a switch, through the PC kernel, up the network stack and into user space.) We also plan to measure the FPGA-to-FPGA latency, which we expect will be lower.

Clock Synchronization

Our tokamak has numerous diagnostics and systems, all of which need to be synchronized to a 10 MHz global experimental clock. We derive our 250 MHz acquisition clock from this signal; this

derived signal clocks the ADC boards. The onboard crystal clock drives the remaining FPGA logic.

Our system is unusual in that it does not send the experimental clock continuously, but only at trigger events for about ten seconds. Outside these periods we need to switch back to an internally generated clock. Thus, we have essentially two clocks that we need to switch between, an external clock and an internal clock.

The key requirement for both FPGA boards is that they must be precisely synchronous. Ideally, since we are sampling at 4 ns, we can expect a readable input sine wave on our ADC at our highest expected frequency to have a period of 8 ns, equivalent to 360 degrees. If we would like five degree phase accuracy we need a maximum skew requirement of 8 * (5/360) = 111 ps. This degree of accuracy is very challenging. Even light travels only 3.3 cm in this amount of time.

We have designed the firmware such that it is identical on both boards. We use a DIP switch to enable or disable different functions required of each board. This dramatically reduces the development time, as we only need to synthesize the firmware once.

The clock, which is generated on one of the boards, travels out over two closely placed SMA ports and then feeds back in (using cables of equal length) to the ADC board that is connected to each FPGA board's FMC port. This is to ensure that each board is running on precisely the same clock, with the only phase difference being that equal to the difference between the two SMA ports on leaving the FPGA board. Figure 4 more clearly illustrates this arrangement.

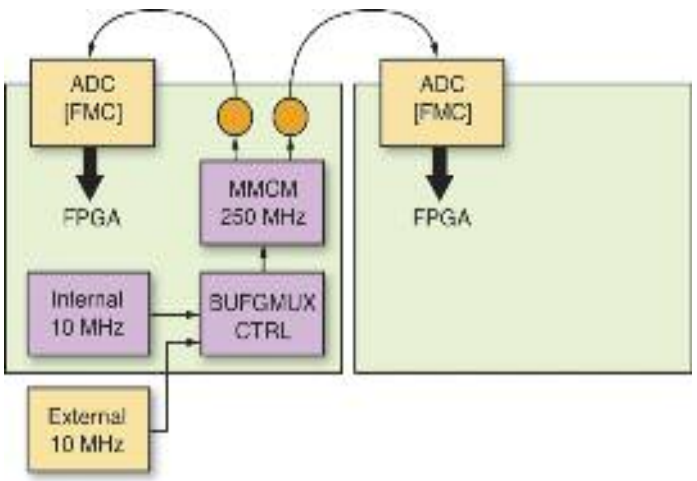


Figure 4: The two FPGA boards must be precisely synchronous. The clocking scheme shown here ensures that they are.

In a similar way to how the external 10 MHz arrives and gets sent out, coming back in on both ADCs, the external triggering uses the same method to ensure that both boards are triggered synchronously.

Benefiting from Unique Features

The Xilinx FPGA architecture offers a number of novel features that we have put to good use in our design. For example, we use the IODELAY primitive to fine-tune path delays at the pins. This allows us to compensate for differences in track length. It was vital to have this capability, since the data path lengths on the ADC attached to the FMC are not equal. Unless we compensated for the path delays, the data from the ADC would have been garbage. The data was coming off the ADC at double data rate with a 250 MHz clock, so the time between each valid piece of data was merely 2 ns. IODELAY allowed us to align the data paths very precisely, in steps of 125 ps.

Equally important are the Mixed Mode Clock Managers (MMCMs), which perform clock management tasks such as multiplication and phase shifting. In cascaded mode whereby one MMCM connects to another, we were able to generate a wide range of clocks from the original 10 MHz. This includes the 250 MHz ADC sampling clock, as well as additional clocks that we used for other purposes.

We likewise made good use of the BUFGMUX_CTRL and IDDR primitives. Since our system switched between internal and external 10 MHz clocks, it is crucial that switching between the two be glitch-free. The BUFGMUX_CTRL primitive allowed us to make sure it was. You can also use this primitive for standard logic such as triggers (not only for clocking); however, you need to ensure that the attributes IGNORE0, IGNORE1 are set to 1 to bypass the deglitching circuitry, which would otherwise not allow the logic to pass through.

The ADC, meanwhile, provides data in a DDR format; that is, the data is valid on both the rising and falling clock edges. To recover this data into single data rate (SDR), we use the IDDR primitive, which is hardwired on the I/O pads. This has a single data pin input, and two data pin outputs. We used the SAME_EDGE_PIPELINED attribute, which ensured the data was valid on both pins at the same time, thus reducing other logic. This does come at the cost of an additional cycle of latency, but for us the latency did not matter.

Another feature of the Xilinx architecture that helped in our design was the FPGA Mezzanine Connector (FMC). Strictly speaking this is not a unique feature of an FPGA, but of an FPGA board. Even so, it has proven very useful and has worked well with the Virtex-6. FMC connectors include high-frequency clock pins, which are wired to clock-capable pins on the Virtex-6 on the ML605 board. As such, it is possible to send a clock via the FMC and into the FPGA. This is advantageous since it means that we need only one entry point for the clock.

Using the Xilinx Tool Suite

Xilinx provides a number of tools to aid in the development of an FPGA system. We used a good number of them.

We used Project Navigator for manually coding VHDL and Verilog code. Additionally there is a graphical interface whereby you can make a "schematic" that allows the creation of logic visually. However, we found Project Navigator to be a low-level tool in that while we could operate easily on flip-flops (single bits), expanding to operations on larger bit numbers was more complicated. We found Project Navigator most useful for low-level clock design. It enabled us to have precise control over which clock was driving specific logic.

For high-level logic design, we turned to System Generator. It is particularly suited to designs where logic is driven by a single clock frequency (although isn't restricted to this case). System Generator is simple to use and has access to large range of IP, such as FFTs, divide generators and filters, to name a few. Additionally, you can tie logic easily into the MicroBlaze processor as read/write registers and shared memory. The tool automatically creates a peripheral core (PCore) and adds it into your XPS project.

We used CORE Generator™ for fine-tuning the parameters of the ADC FIFO. This FIFO had to be 256 bits wide with a write clock of 125 MHz and a read clock of 200 MHz. We imported the resulting generated NGC file into XPS as a PCore. We did this manually by creating the necessary .mpd, .pao and .bbd files.

The Impact tool helped us to program the FPGA, and also to generate the SystemACE™ file for permanently placing the firmware onto the CompactFlash. The CompactFlash worked very reliably, however it should be noted that this added an extra requirement (see under SDK, below) to our system.

Since we wanted our system to include the MicroBlaze processor, we needed the tool that creates the processor system: Xilinx Platform Studio. XPS is a comprehensive tool suite that allows you to build processor-centric systems. It allows you to set up the required links through a wizard. You can also include IP from CORE Generator by using the Create IP Wizard. It now includes the high-performance AMBA AXI4 on-chip interconnect.

Finally, we used the Xilinx Software Development Kit (SDK) to develop programs that run on the processor. In fact, we have only one program to run initially, and that is the SREC bootloader. Due to the CompactFlash having a FAT file system, the libraries required to access the SREC program (also on the flash) inflated the size of the resulting executable. We reduced its size by turning off debugging, turning on optimization and including "mb-strip–g <elf_file_name>" as the postcompilation command. Even after all these steps, the result was a large, 91 kbyte executable. Therefore, we had to increase the internal BRAM so that we could initialize the bitstream with this size of executable.

One problem we faced was the large compilation time with the Virtex-6. The Xilinx software PlanAhead™ can significantly help with this challenge. We intend to utilize PlanAhead to its full potential to reduce the compilation time.

We are excited by the possibilities that the new Zynq™-7000 extensible processing platform will provide (see cover story, Xcell Journal Issue 75). However, it remains to be seen whether Zynq will make the MicroBlaze obsolete, or if the MicroBlaze will hold its own thanks to its soft nature and the 10-plus years of development effort behind it. Could a future cache-coherent multiprocessor MicroBlaze system outperform the ARM® dual-core Cortex™-A9 MPCore™? Could the Physical Address Extension in the Zynq or MicroBlaze lead to more powerful systems that provide more than 32 bits of address space, thus allowing more than 4 Gbytes of RAM? It will be interesting to watch and see how time answers those questions.

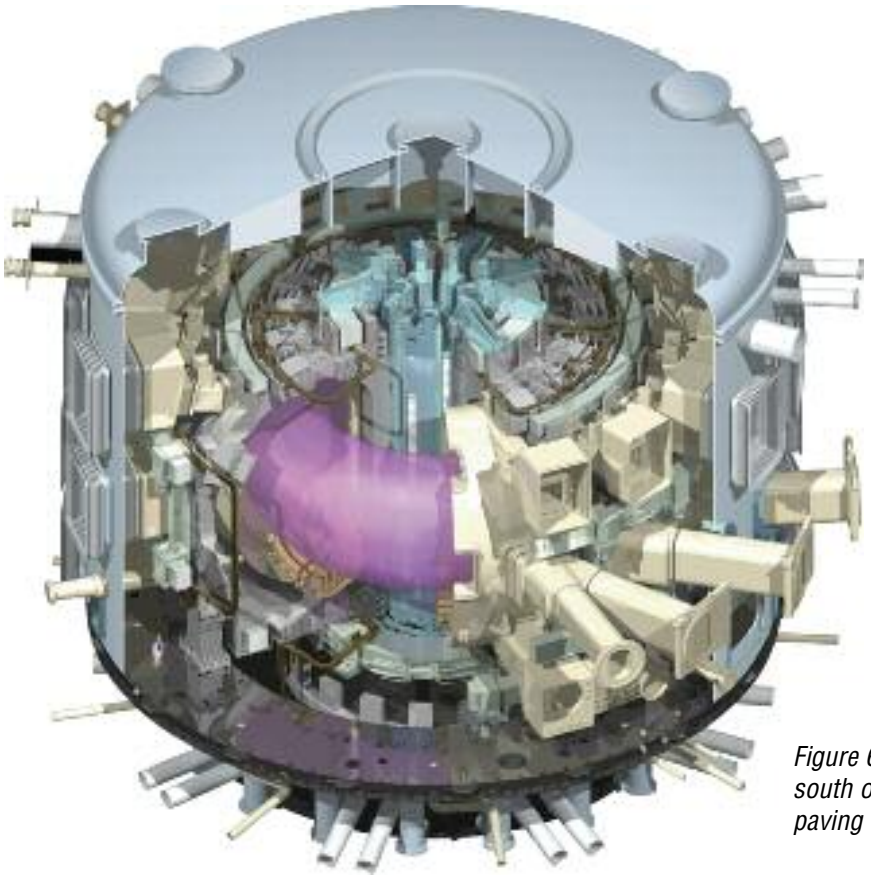
A Cutting-edge System

Ultimately, we developed a fully functional data acquisition system (see Figure 5) that is cutting edge in the world of FPGAs, making use of the latest Xilinx technology. It is capable of

Figure 5: The FPGA data acquisition box comprises Xilinx's ML605 evaluation board, 4DSP's FMC108 ADC board and our in-house FMC/PMOD header board. We wired the ADC SSMC connectors internally to front-panel SMA bulkheads to extend the life of the ADC analog connections.

real-time acquisition at 10 Gbytes/s (or 80 Gbits/s). The end cost was less than $15,000. We have demonstrated technology that we hope will find its way onto the largest fusion experiments in the world, such as the ITER project (see Figure 6 on page 17).

Fusion energy is one of the biggest technological challenges ever attempted. FPGAs are helping us to crack this tough nut in different ways by leveraging their unique advantages. Our fusion research device, which incorporates Virtex-6 FPGAs using the



latest AMBA AXI4 interconnect and the Xilinx tool flow, achieves extremely high data rates on a small, compact system.

A new website (http://fusion.phys.tue.nl/fpga/doku.php) promises to be a meeting place for people to communicate ideas and material for developing FPGA technology on fusion devices.

END

Acknowledgements

CCFE is associated with Durham University's Centre for Advanced Instrumentation and the University of York's Plasma Institute.

This work was funded partly by EPSRC under grant EP/H016732, by the University of York, by the RCUK Energy Programme under grant EP/I501045 and by the European Communities under a contract of association between EURATOM and CCFE.

The authors wish to thank John Linn, open-source Linux engineer at Xilinx, and the other Xilinx employees and Xilinx partners who have contributed to Linux support for the MicroBlaze processor.

Figure 6: The ITER tokamak, currently being built in the south of France, will produce 500 megawatts of fusion energy, paving the way for a demonstration fusion power plant.