eFPGA LUTs Will Outship FPGA LUTs Later This Decade and Will Enable Software Control of FPGA

by
Geoff Tate, CEO of Flex Logix

FPGA has become strategic technology. It used to be a "poor man's ASIC" and provided a customized IC for lower-volume applications. While it is still used this way in many systems, it has also become strategically important to two very big, high-growth applications:

- Cloud data centers: networking, storage and security
- Communications systems: base stations and 5G, etc.

In fact, FPGA is so strategic for the data center that it caused Intel to buy Altera and more recently AMD to buy Xilinx. This is because processor workloads in many cases are shifting to FPGA.

Data centers use FPGAs in volume to provide the parallel programmability a processor cannot achieve (one customer calls it "programmability at the speed of hardware"). These FPGAs are paired with dedicated function ICs such as NICs (network interface chips) and network switch chips. Each data center has different workloads and needs so a standard product for all doesn't work, and each data center has the volumes and capital to optimize for their needs.

Communications systems have long used FPGA to handle hundreds of national frequency bands and protocols. Now with 5G, FPGA is used to manage the complexity and evolving standards such as O-RAN.

While FPGA programmability is very valuable as described above, the power and size of FPGAs is another story. As a result, today data center and communication companies want to integrate their FPGA into SoCs to reduce size and power.

Integration of FPGA has already begun

Companies such as Achronix, Flex Logix and Menta have provided embedded FPGA (eFPGA) capabilities for almost a decade. eFPGA is now available on process nodes from 180nm to 7nm, and in capacity from 1K LUTs to >>100K LUTs with 1M LUTs coming. DSP and Block-RAM options are also available. Below is a brief history of some of the adoption that has taken place throughout the industry.

- The first customer to announce the successful use of eFPGA was Sandia National Labs working for their 180nm fab in New Mexico. They have made numerous ASICs with eFPGA since.

- Subsequently, Boeing, AFRL (the Air Force Research Lab), Microsoft, BAE, and many other government agencies and subcontractors also began using eFPGA extensively in US-manufactured defense systems. As a result, eFPGA is now a well-established technology for the DOD and DOE.

- Morning Core, a subsidiary of Datung Telecom, announced that it was using eFPGA in FinFet ASICs for car-to-car communications.

- Dialog announced the ForgeFPGA family of eFPGA starting at 1K LUTs at less than 50 cents in volume and sipping just milliwatts.

- Flex Logix uses eFPGA in its own 16nm AI Inference chips with performance over 500MHz which is being improved over time through software enhancements to 667MHz and then 800MHz.

- Socionext is working with a major systems company on a 7nm SoC using eFPGA with 100K LUTs of programmable logic running over 500MHz at worst case conditions. This is a 5G application and the first high volume communications chip to integrate large amounts of LUTs.

In addition to the above, there are many more customers using eFPGA that are not yet public. The pace of customer adoption of eFPGA is accelerating as awareness of the technology grows and leaders use eFPGA to gain a significant advantage in their market segments.

Integrating FPGA can allow for greater software control of FPGA

FPGAs have significant advantages, but they are not easy to program and there is a much smaller pool of qualified FPGA programmers than processor programmers.

Why is FPGA so hard to program?

- Verilog is a low level language more like assembler than C++.
- Parallel programming is very hard for traditional programmers to comprehend and learn.
- An FPGA is programmed as one giant “blob” of code. A processor has subroutines, main programs, linkers and loaders, paging, etc. that a programmer takes for granted. None of this is available for FPGA.
- FPGAs are programmed in seconds, which is an eternity in hardware and usually the whole FPGA must be reprogrammed. There is some partial programming, but it’s slow

and all operations stop while it happens. In a processor code, pages are swapped in and out of cache all the time while the main thread runs.

eFPGA provides an opportunity to re-think the programming strategy to leverage scarce Verilog coders and enable C++ coders to take software control over FPGA.

Here is the basic concept of one approach to do this:

1st. Containerize/modularize code into "subroutines"

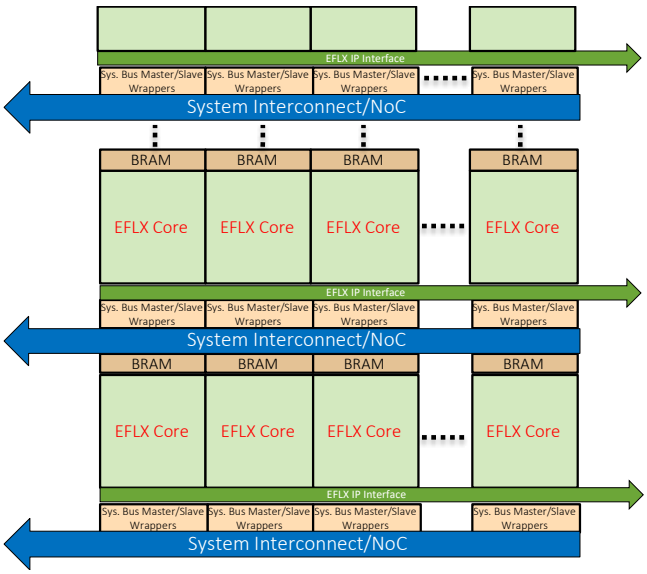
Segment an FPGA into modules or containers of smaller size and provide each of them with direct access to DRAM memory and the processor.

eFPGA is built up modularly already using tiles that can be "snapped" together with Block RAM (BRAM) between the rows as needed.

It is easy to add a system interconnect/NOC/AXI bus and provide every FPGA module/container access to memory/processor.

Now write FPGA code that acts like a subroutine contained in a container; provide it with input data or pointers to data in system memory; have the FPGA execute; then deliver the results as output data or as a pointer to data in system memory.

Use the scarce Verilog coders to write the compute intensive "subroutines," and have the C++ coders write code on the processor that calls the subroutines when needed.

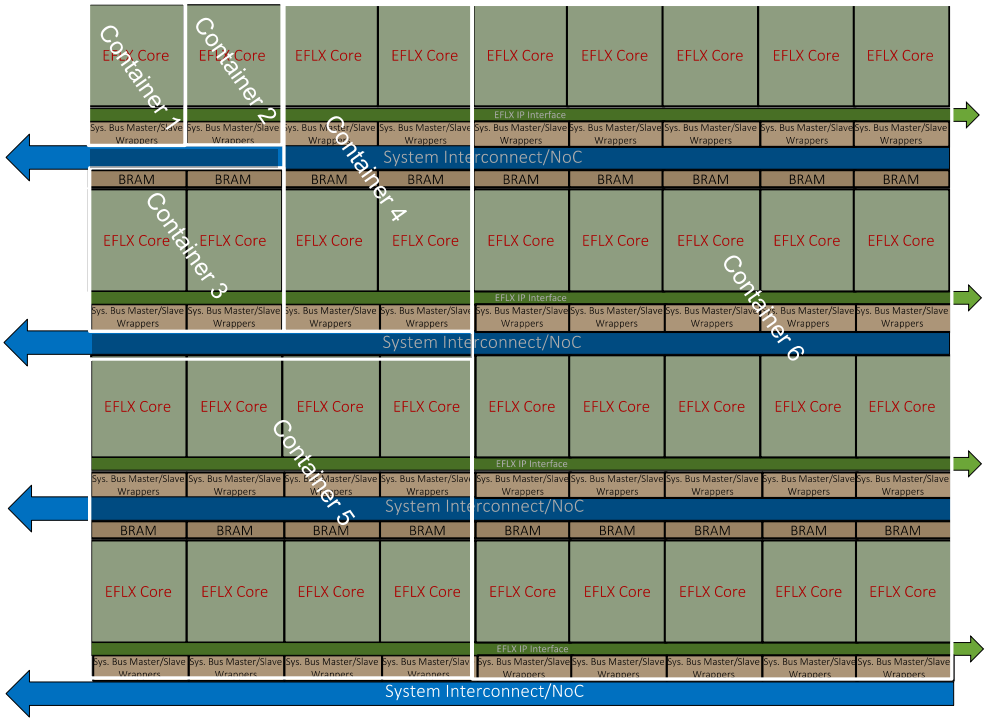


2nd. Allow containers/modules to be variable in size

Segment an FPGA into modules or containers of smaller size and provide each of them with direct access to DRAM memory and the processor.

Some algorithms are simpler and use fewer LUTs. Some use more LUTs.

As an example, using Flex Logix's flexible interconnect fabric, it is possible to enable containers to be of any rectangular size up to the full size of the array.



3rd. Containers are pageable in microseconds (millionths of a second)

FPGAs have always been programmable in seconds from Flash memory – very slow and generally done infrequently: at boot time or when an upgrade is required; like updating your iPhone.

However, eFPGA is already being reprogrammed in millionths of a second in a leading AI inference processor. This is required because the inference accelerator processes a neural network layer doing billions of computations then reconfigures for the next in <10 microseconds and restarts compute.

This microsecond reconfiguration can be applied to containers/modules in the array above. While a container is being reconfigured, the rest of the containers continue to run at full speed.

This allows eFPGA to do paging like processors.

Take control of your FPGA power, size and ease of programming

We've watched the industry transition from simple ALUs ⇒ processors ⇒ microprocessors ⇒ parallel processors ⇒ SOCs (that include cores and accelerators). And today, we have reconfigurable SOCs that better take advantage of the large number of cores in a system.

eFPGA will enable data center and communications customers to continue to benefit from the parallel programmability of FPGA while lowering power, shrinking size and taking software control of FPGA to improve productivity and time to market. For all of these reasons, eFPGA enables a new paradigm shift in computing architecture both improving the compute density per rack through integration and allowing the benefits of eFPGA to be enjoyed by the much larger contingent of C++ programmers. For these reasons, eFPGA integration will accelerate and more LUTs will be integrated in SoCs than sold in FPGAs later this decade.

Geoff Tate is Founder and CEO of Flex Logix. You can learn more about eFPGAs at Flex-Logix.com