Software Controlled Modular FPGA White Paper

By: Geoff Tate, CEO Flex Logix Technologies, Inc.

Background:

Flex Logix has developed embedded FPGA IP (EFLX® embedded FPGA or eFPGA) that has been licensed for use in many commercial, aerospace and defense programs. It has also developed an edge inferencing accelerator, InferX® to efficiently process AI edge inferencing workloads requiring high throughput for the least power and area. This paper describes managing and dynamically programming eFPGA designs through software by a host processor combining patented, silicon proven techniques developed for Flex Logix eFPGA and InferX.

Integrating FPGA can allow for greater software control of FPGA

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

Why is FPGA so hard to program?

- Verilog code, used to program FPGAs, is a low-level language more like assembler than C++.
- Parallel programming is very hard for traditional software programmers to comprehend and learn.
- An FPGA is programmed as one giant "blob" of code. On the other hand, a processor has subroutines, main programs, linkers and loaders, paging, etc. that a programmer takes for granted. None of these capabilities are available for standard FPGAs.
- FPGAs are programmed in seconds, which is an eternity in hardware timeframes and usually the whole FPGA must be reprogrammed. There is some partial programming, but it's slow and all operations stop while it happens. Whereas with processor code, pages can be swapped in and out of cache while the main thread runs.

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

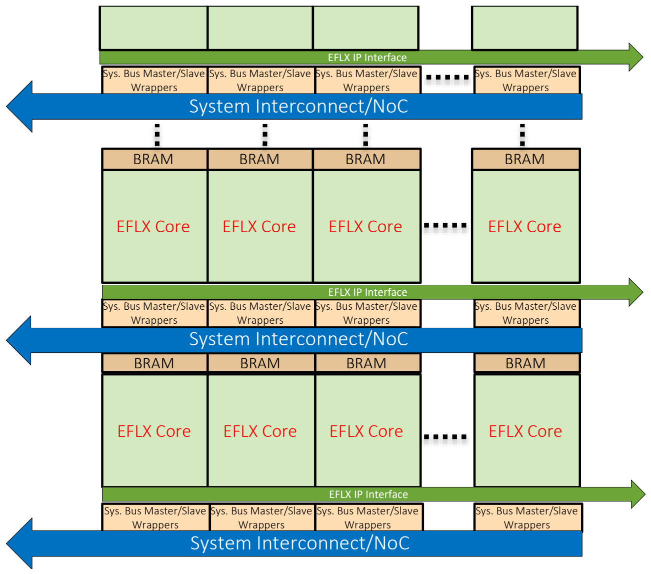
Software Controlled Reprogrammable eFPGA Basic Concept:

Step 1: Containerize/modularize code into "subroutines"

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

Flex Logix eFPGA IP was designed modularly using tiles
that can be “snapped” together with Block RAM (BRAM)
between the rows as needed to build up an eFPGA fabric of any size.

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



Now use the scarce Verilog coders to write compute intensive “subroutines” (aka FPGA code) that would be programmed into a container; provide it with input data or pointers to data in system memory; have the eFPGA execute; then deliver the results as output data or as a pointer to data in system memory.

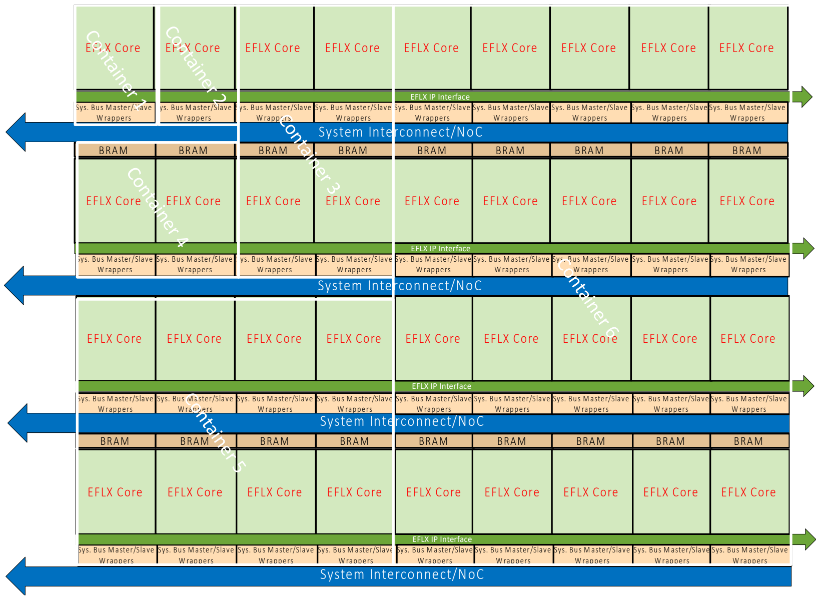
Store the “subroutines” in on or off -chip memory,” and have the C++ coders write code on the processor that calls the subroutines when needed using a function call or pragma in their code, just like they do today for hard-wired co-processors or custom instructions.

Step 2: Allow containers/modules to be variable in size

Segment an eFPGA 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 any rectangular size up to the full size of the array. The array can be as little as 2 or 3 Flex Logix eFPGA cores or larger as illustrated below.



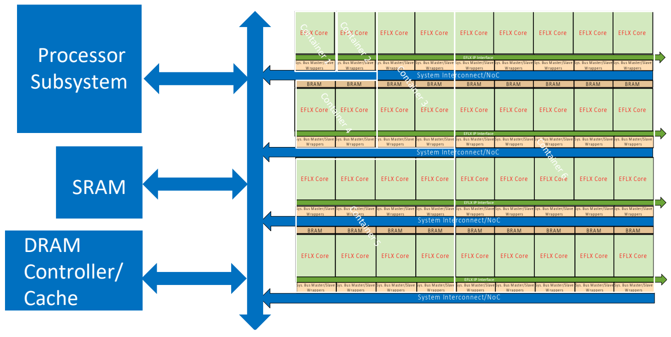
Step 3: Containers are pageable in microseconds (millionths of a second)

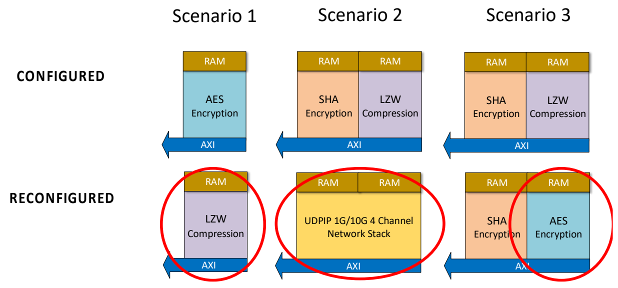
FPGAs have always been programmable in seconds from Flash memory – very slow and generally done infrequently: at boot time or when an upgrade is required; like updating your iPhone.

However, eFPGA can be reprogrammed in millionths of a second as demonstrated in Flex Logix's InferX X1 inference co-processor, taped out in TSMC 16FFC process. This was required because the inference accelerator processes a neural network layer utilizing a layer specific accelerator which executes billions of computations, then in <10 microseconds, reconfigures the accelerator with the next layer specific accelerator and restarts compute.

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

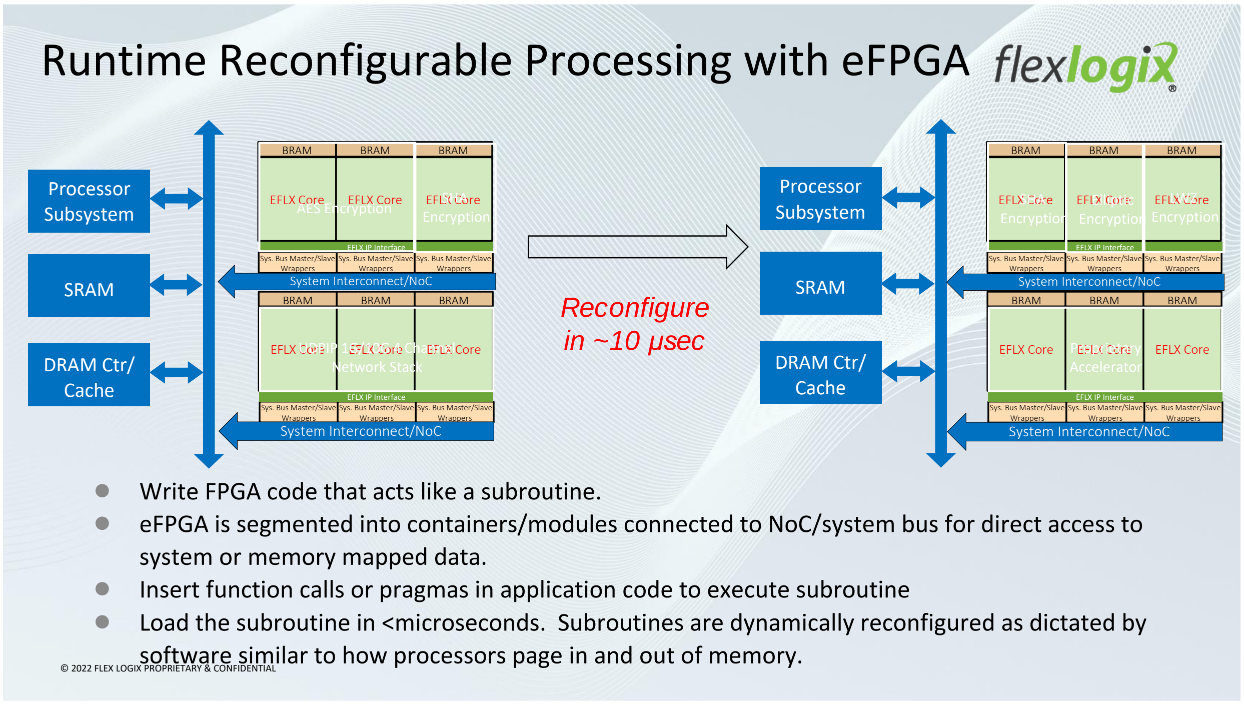
This allows eFPGA to page in the subroutines/FPGA code like processors.





Benefits of an eFPGA Based Solution:

With the flattening of Moore's law, dedicated accelerators are being highly leveraged to provide more compute power, whether they are implemented in ASIC gates or as FPGAs.   From a chip architecture point of view, they are co-processors residing on an internal system bus or accessed through a chip's highspeed interfaces. With an integrated eFPGA approach, power is reduced through the removal of redundant FPGA serdes, latency is greatly improved by eliminating chip to chip data transfers and costs are lowered through reduced system chip count.



Runtime Reconfigurable Processing with eFPGA

- Write FPGA code that acts like a subroutine.
- eFPGA is segmented into containers/modules connected to NoC/system bus for direct access to system or memory mapped data.
- Insert function calls or pragmas in application code to execute subroutine
- Load the subroutine in <microseconds. Subroutines are dynamically reconfigured as dictated by software similar to how processors page in and out of memory.

A Customizable Runtime Reconfigurable Solution Using a Scalable, Portable Technology.

Flex Logix eFPGA is proven and available in TSMC 40ULP, 16FFC, 12FFC, N7(in design), Globalfoundries GF12LP, GF12LP RHBD, GF22FDX (in design) and Sandia 180nm CMOS8. Porting to new process nodes takes 6 - 9 months. Any size array up to 2M LUTs can then be produced within a few weeks. The portability and scalability of container/module eFPGA based accelerators enables use on any process a customer chooses to use today and future chips on different process nodes.

The above approach is also processor agnostic, so companies can leverage their investment in existing software frameworks, tools, and applications.

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. Flex Logix through the development of InferX has created new technology that will enable software programmers to take advantage of eFPGA based hardware accelerators with extremely fast reconfigurability for their changing compute tasks.

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 board or rack through integration and allowing the benefits of eFPGA to be enjoyed by the much larger contingent of C++ programmers.

Geoff Tate is CEO &Cofounder Flex Logix Technologies, Inc. His background includes: BSc, Computer Science, University of Alberta. MBA Harvard. MSEE (coursework), Santa Clara University. 1979-1990 AMD, Senior VP, Microprocessors and Logic with >500 direct reports. 1990 joined 2 PhD founders as founding CEO to grow Rambus from 4 people to IPO to $2 Billion market cap, till 2005.

More eFPGA articles are available at https://flex-logix.com/news/ including:
Top 5 predictions for eFPGA in 2022

On-Chip FPGA: the "other" compute resource

eFPGA Saved Us Millions Of Dollars. It Can Do The Same For You.

Add Security And Supply Chain Trust To Your ASIC Or SoC With EFPGAs

Video:
Reconfigurable Computing with Analog and MCUs