Disruptive Compute Alternatives to COTS

Michael Parker
Raytheon, SAS
El Segundo, CA. USA

Abstract— **In recent decades, the defense community has largely adopted a COTS (commercial off-the-shelf) approach to high performance electronic and computerized weapons systems. The reasons were compelling, as the scale of investment of commercial industry was able to drive innovation in semiconductor process, architecture, IP and software eco-systems more quickly than custom defense programs, which has resulted in higher compute density and system capability using COTS devices. Furthermore, the volumes typical in defense programs do not generally provide the economies of scale for a custom chip development, with the required high engineering design and verification effort, as well as high manufacturing costs. Custom chip costs and transistor counts continue to increase at an exponential rate as semiconductor processes advance.**

By definition, COTS semiconductor products are used in commercial applications, and so can be easily obtained by adversarial nations. A major drawback of COTS is this technology is also in use in our adversary's weapons systems, which may even adapt the latest semiconductor products at a more rapid rate than US defense programs.

Another trend is the near ubiquitous "fabless" semiconductor model, where semiconductor companies design their chip products, but outsource manufacturing to a small number of foundries which provides the economy of scale for large investments necessary. This fabless business model has also supported the creation of many IP companies, which have created a wide variety of innovative architectures and tools which can be customized for specific applications. Many of these smaller companies are also more nimble, and can focus on optimizing a solution for specific problems, in contrast to large semiconductor vendors which generally are developing products for widest possible application.

An alternative to COTS semiconductors is to develop heterogeneous, programmable ASICs, which could be used for multiple defense applications in multiple programs. There are available a wide range of COTS IP and architecture, which, when customized, can provide a higher compute density for specific applications in demanding SWAP environment. Since these IP cores are programmable, this provides opportunity for firmware upgrades over the life of defense programs. Also, a custom chip composed of heterogeneous, programmable IP cores will be much more resistant to cyber attack compared to COTS semiconductor products.

To illustrate, a programmable ASIC example is presented which includes the following:

- **eFPGA, customized for defense applications of ML, radar, EW and to act as secure gateway for cyber protection**
- **Vector DSP, to provide balance of agility and parallel performance for dynamic applications such as networked comms or AESA and cognitive radar processing**
- **Vision engine, for 3D perceptual awareness, ML, and vision sensor pre-processing**
- **ARM CPU for overall system management, high level decision making**

In addition, it has been noted by DARPA that "Moore's law", which states that semiconductor density doubles approximately every 18 months, has ended. DARPA is therefore investing in programs emphasizing innovative architectural advancements in order to improve compute density for key applications.

I. Introduction

A notional architecture is shown in Fig 1, depicting a programmable ASIC intended digital signal processing applications in defense. The focus in on the core processing blocks. The I/O structures are not shown as it is assumed for any ASIC development that the I/O requirements are well defined. The proposal is to consider replacing application specific signal processing circuits with programmable cores.

Another important consideration is that a chip composed of a heterogeneous mix of programmable cores is unlikely to fall under ITAR restrictions, and therefore have greater flexibility on which fab to use for chip manufacture, including overseas locations which are generally not considered secure foundries. The ITAR content will be in the programming of the cores, which is done at the defense system integrator level.

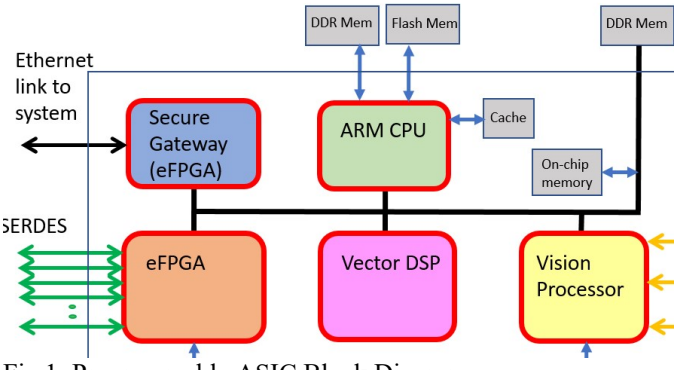

Fig 1: Programmable ASIC Block Diagram

II. EMBEDDED FPGA

Begin with FPGAs, which are general purpose devices dominated by large amounts of programmable logic. FPGA logic, while extremely flexible, is inherently inefficient compared to purpose built ASICs, ASSPs and some other programmable compute architectures. An industry rule of thumb is an FPGA is about 10% as efficient as an ASIC, due to the high amount of overhead in programmable routing and constructing logical functions using look-up-tables (LUTs). Also, the signal processing performed in FPGAs typically utilizes fixed-point numerical representation, which uses significant logic circuits to maintain the signal in the optimal range. With fixed point, the user must manage the bit growth naturally introduced by basic DSP operations of multiplication, addition an accumulation. To maintain the proper dynamic range and scaling of the signals at each step, various methods such as saturation, rounding, truncation, accounting for bit growth are used. This requires more design verification effort on part of user, particularly when the system design modeled in floating point.

However, some FPGAs are now supporting single precision (FP32) and have half precision (FP16) floating point support coming soon. The use of floating point and the ability to optimize the FPGA solely for signal processing workloads allows for higher ratio of compute and memory blocks, relative to the amount of programmable logic in existing FPGA devices.

Two algorithms were benchmarked using Intel Arria 10 FPGA, the first FPGA with built in single precision floating point support. A matrix transformation algorithm, QR Decomposition, was used and is reasonably complex as well as computationally demanding. Next a very high-throughput "super-sample" FFT, where the FFT can process many samples simultaneously in parallel (super-sample meaning the input sample rate is far higher than the FPGA clock rate). This requires a high degree of compute resources, but also a high degree of programmable routing in the FPGA, using more logic. The ratios of the resources used in the Arria 10 GX1150 FPGA are given in Table 1, along with the GFLOPS. These are large designs, using a significant portion of FPGA resources, and realistic representations of processing in defense applications.

Percentage Resources Used	QR core: 128x128 607 GFLOPS	QR core: 64x64 237 GFLOPS	FFT core: 32k points 854 GFLOPS
Arria 10 10AX115	Logic: 6% DSP: 69%	Logic: 3% DSP: 35%	Logic: 19% DSP: 90%

Table 1 FPGA Processing Resource Usage

Notice that for floating point signal processing, the FPGA uses a small percentage of logic, even in designs where considerable routing and pipelining is needed. The unique value of FPGA architecture is the programmable routing. The routing allows the dataflow to match the natural algorithm flow.

The standard floating point precisions are aligned to 16 bit multiplies to match memory widths. However, this precision may not be the best choice. Just as FPGAs did not support the standard 16 bit integer capabilities of DSP processors, but found 18 bits was more optimal, the same is true for floating point. This flexibility is available in a new technology known as embedded FPGA, or eFPGA. The eFPGA is an IP core which can be licensed and customized in an ASIC design. An eFPGA can be optimized in at least two ways. First, the ratios of programmable logic, memory blocks and DSP blocks can be dictated by the signal processing applications. Since programmable logic is a disproportionate amount of the area, this can have a significant impact on compute density. Second, two new custom floating-point precisions can be supported, as FPGAs are not tied to the tradition of 16 bit multiples. These new proposed precisions are extended versions of the standard half and single precision floating point numerical representations. The new precisions offer better system performance at similar circuit power and area. The extended half precision can improve on the traditional FPGA 18 bit fixed point in regards to dynamic range, precision, programmable logic usage and user design effort.

Floating Point Precisions

Half Precision (FP16)	1 bit sign, 10 bit mantissa, 5 bit exponent
Extended Half Precision (FP20)	1 bit sign, 15 bit mantissa, 5 bit exponent
Single Precision (FP32)	1 bit sign, 23 bit mantissa, 8 bit exponent
Single Precision (FP40)	1 bit sign, 31 bit mantissa, 8 bit exponent

Precision	Description	Format S.Mant.Exp	Word Width	App SN
Extended Single Precision	FP40	1.31.8	40 bits	192
Single Precision	FP32	1.23.8	32 bits	144
Extended Half Precision	FP20	1.14.5	20 bits	90
Half Precision	FP16	1.10.5	16 bits	66

Table 2 SNR and Dynamic Range Comparisons

Using 20 bit wide internal dual port memory blocks, an eFPGA can support these extended precisions with very minor amounts of extra circuitry in the DSP blocks. It would be feasible for the DSP blocks to support a single MAC in FP40, or two MACs of FP20, with the same density and efficiency of FP20 as would be customary with 18 bit fixed point. Additionally, vector modes should be included to allow full use of all multipliers and adders to compute dot products, which is the processing kernel in many algorithms, including

matrix multiplies, convolutions (filtering), QR, Cholesky and SV decompositions.

Communication systems have traditionally used 16 to 18 fixed-point or integer processing has been used. However, the use of MIMO is driving up the dynamic range and precision requirements for communications algorithms. Matrix processing is now required to perform channel adaptation and equalization, and determine beamforming coefficients. For smaller matrices typical in many MIMO systems, the use of FP20 is sufficient, generally providing accuracy on the order of 10^{-5} (single precision is closer to 10^{-7}). Often 16-18 bit fixed-point processing is insufficient for various portions of MIMO processing. In contrast, FP20 is ideal, while also offering both logic and power reductions compared to 18-bit fixed-point. And of course, even higher precision is available if there are much larger antenna (64 to 256) arrays, necessitating use of larger matrices. For defense, the communications processing is also often integrated in radar and EW systems (which typically use array antennas), and having a wide array of floating point precisions to dynamically select at each different processing step is ideal for the most power and throughput efficient implementation.

Radar, both military and commercial, typically has varying precision requirements. The front-end processing such as pulse compression, Doppler processing, array beam forming, and pulse generation tends to be lower precision, often implemented in fixed point. This type of processing is ideal for FP20, as FP16 is insufficient. The benefit is both a 2X increase in compute density and efficiency over single precision, and is also more efficient than fixed point. And FP20 provides much higher and more consistent SNR than today's fixed-point processing. FP20 is ideal for FFTs, where bit growth requires higher dynamic range.

Radar back end processing examples include GMTI (STAP), image formation and back projection (SAR). Matrix multiplication and inversion, eigenvalue determination are typical processing steps in this type of processing, and often requires more dynamic range and precision than afforded by single precision floating-point, and then requires double precision. For higher bandwidth frequency domain processing, there is increased demand for the dynamic range, which can make single precision insufficient in increasing portions of the data path. However, double precision provides on the order of 150 dB of additional SNR compared to single, and this is much more than typically needed. Use of FP40, which provides a more modest 48 dB increase at the same compute density and efficiency as single precision, is ideal for this. No other architecture can offer FP20 or FP40 – the alternative must use higher power double precision floating point.

Electronic warfare (EW) encompasses a wide range of signal processing techniques, and typically requires very low latency between the receiver and the chosen transmit response. EW often processes very wide spectral bandwidths, which typically will require high numerical precision in the frequency domain processing. The optimized FPGA supports this mix of precisions, with very low latency afforded by FPGA "stream and compute" architecture, and high processing throughput and efficiency. Integer processing at 18 bits can be used for frontend FIR filtering. Alternately, FP20 can be used for time domain (FIR filters) processing and a mix of FP20 and FP40 can be used for frequency domain processing. As EW systems are designed to operate in the presence of jamming signals, a very high degree of dynamic range may be needed in the frequency domain processing to distinguish the low power desired signals in the presence of high power jamming signals. The use of FP40 is available for this, especially as double precision processing using GPU or CPUs may not be an option due to the latency and real-time processing requirements.

A key attribute of the FPGA architecture for signal processing is the ability to connect, with deterministic latency, the memory and DSP resources in a pattern that precisely matches the algorithm dataflow, using the FPGA programmable routing. Leveraging this capability generally allows near 100% duty cycle of the DSP blocks, which results in the FPGA to operating very close to the peak, theoretical GFLOPS rating of the device, sometimes called a "stream and compute" model. This can be achieved with relatively modest amounts of FPGA logic and routing. In addition, a minimal amount of programmable logic is needed to implement the control circuits needed locally throughout the algorithm data path. All other architectures require the programmer to map the data flow of the algorithm onto the pre-determined data buses built into the device, with varying degrees of efficiency.

The architectural changes suggested in this section are possible with eFPGA offerings from vendors such as Achronix, which can be used in the form of synthesizable core in a custom chip implementation. The Achronix eFPGA also offers a machine learning compute block with lower precision block floating point modes.

III. VECTOR DSP

Vector DSPs are available as IP cores, and can allow hundreds or thousands of MAC engine processing to be performed in parallel using VLIW architecture. Key reasons to use a vector DSP is for applications where the processing algorithm changes depending upon the data being processed. Such applications are difficult to implement in a hardware architecture efficiently, whereas the vector DSP has highly parallel and native DSP capabilities with advanced real-time controller attributes. Vector DSPs are also designed for extreme power efficiency, given that a key application is 4G and 5G processing in smart phones.

Vector DSPs can offer many features to boost throughput and enhance efficiency:

- 8, 16, and 32 bit integer support

- Single precision floating point support
- Custom floating point numerical formats
- Native support for complex math
- Specialized instructions for algorithms such as FFT, FEC, and PRBS
- Wide vector sizes and L1 memories
- Optimized math and linear algebra libraries

A vector DSP can offer multiple vector processing engines or units, to further enhance the degree of parallelism, where each vector unit can operate independently.

Modern vector DSPs are design to support linear algebra common with MIMO algorithms, using floating point data representations. Convolution for FIR filters is usually implemented in fixed point, special instructions to support the data flow patterns, including optimizations for symmetric filters.

Vector DSPs typically interface to other blocks in the programmable ASIC using standard AXI bus protocols, including external memory and embedded CPUs. These AXI buses are quite wide, and support burst transactions. 2 and 3D DMA support is also available, to provide for seamless block data transfer in complex patterns as required by the algorithm processing flow.

Some vector DSP architectures can also support data caches with coherency. Non-blocking L1 caches are available for each core. Use of a shared L2 cache can allow multiple cores to use shared memory without software intervention.

A recent study conducted by the author showed that a vector DSP could best an ASIC hardwired implementation in terms of power efficiency, in a DSP intense application. This is indicative of how optimization level of a vector DSP.

The development environment for Vector DSP includes compiler, simulator (cycle accurate), hardware debugger with breakpoint support, profiler, as well as project management. The design entry is in C/C++, with vector intrinsics. A popular vector extension is the Vec-C extension. Automatic vectorization of the native C/C++ code is usually provided.

The leading vector DSP IP vendor is CEVA, which offers many variants of DSP IP cores all with common development environment. This allows for scalability in throughput and feature sets which can be tailored to the desired application space.

The vector DSP is likely the optimal architecture when a combination of hardware parallelism, high throughout, low latency and low power consumption are needed, for a complex mix of algorithms which may change dynamically (at run time) depending upon the operational situation. Some examples might be a cognitive radar system, where transmit waveforms are determined dynamically depending upon the environment, or for a complex, mesh communications systems needed on battlefields or in a crowded airspace, and for electronic warfare waveform generation and processing.

A Vector DSP and eFPGA may be very complimentary, with the eFPGA performing the highest throughput tasks which are well defined during operational modes, and the Vector DSP supporting the dynamic portions of algorithms in the processing chain. The Vector Processor will also be much easier to develop and more efficient to implement very complex or dynamically changing algorithms compared to eFPGA.

Depending on the size, an eFPGA could offer a TeraFLOP / TMACs performance, whereas vector DSP typically offers hundreds of GFLOPS / GMACS of performance.

In addition, machine learning is a wide application space, including CNNs, RNNs, LTST graphs and more. The FPGA has key advantages not reflected in raw TFLOPS figures. For each application, the necessary connections between memory and DSP blocks optimize the datapath to the algorithm can be built in the programmable logic/routing. This can include short and low latency feedback loops, very low precision where warranted, and ability for non-sequential mappings.

IV. Vision Processing Engine

Vision processors are optimized to interface directly with sensors and support functions such as deep learning, video analytics, and camera pre-processing. Some vision processors can also perform video compression/decompression (codec). Due to specialized architecture, the vision processor is more efficient at these tasks than DSPs, FPGAs, GPUs or CPUs. Note – some GPUs have added special cores to support deep learning algorithms. The Nvidia Tensor cores are essentially hardened matrix multiply blocks to perform 2D convolution, and do not have the programmable flexibility of the CUDA GPU cores.

Vision Processors are specifically built to act as a CPU accelerator for:

- Deep Learning
- Computer Vision or Video Analytics
- Image Processing
- ISP (Camera Preprocessing)
- Video Compression (optional)

Multiple vendors have deployed their latest versions of Vision Processors in 2018. This paper will reference the v-MP6000UDX from Videantis. Videantis, along with several their competitors, were previously analyzed across several computer vision benchmarks by the author, and consistently provided the greatest throughput per Watt, and superior silicon area efficiency.

The v-MP6000UDX is vision processor is available in fine grained, scalable IP form, allowing up to 256 cores. Each core is composed of many VLIW/SIMD media processors (called MPs by vendor) and a smaller number of stream processors for more serial functions such as necessary for compression/decompression.

For deep learning inferencing, the v-MP6000UDX can scale to 16 TMACs @ 1 GHz, and can process all the layers is a neural net (not just convolution) in real time with low latency.

Complementary to deep learning is video analytics. Video analytics function can be used to identify regions of interest, to determine motion, depth perception (3D) and provide contextual awareness.

Popular techniques supported include Haar/Adaboost, HOG/SVM and optical flow, used in tasks such as SLAM, stereo vision, SfM and other algorithms. Basic building blocks are standard functions such as pyramid generation, histograms, edge and line detection. Efficient support of these visual tasks requires specific high-bandwidth memory architectures, multi-channel DMA, vector register files and many other architecture features.

Camera or vision sensors have specific artifacts which must be corrected prior to performing higher level functions. The vision processor suited for this preprocessing, or ISP (image signal processing) functions. The ISP is typically specific for a given camera, and provides lens correction, de-bayering, high dynamic range enhancement, filtering, image transforms, color conversions, and scaling. Image stitching can also be performed across multiple cameras.

The v-MP6000UDX also comes with a codec library. Compression can be useful when video streams must be transmitted or stored to other nodes in the system (in addition the metadata resulting from computer vision processing). Standard compression algorithms such as H.264, MPEG, JPEG are supported, as well as ultra-low (few milliseconds) delay compression algorithms. The use of the SP processors, which accelerate algorithms such as CABAC or Huffman, is key to enabling codec support, a feature many vision processors lack.

Having one processor architecture for all vision tasks provide maximum flexibility, reduces data movement and memory requirement, and increases the application and algorithm space which can be addressed.

Standard libraries such as OpenCV and the more embedded OpenVX library are supported as function calls which operate seamlessly from the host CPU. The libraries take full advantage of multi-core architecture to execute in parallel. OpenVX also has the advantage of using graph manager which reduced the memory interaction across function calls.

A neural net kernel library CNNDesigner is available which to map popular neural networks to the v-MP6000UDX, and is compatible with these networks trained using popular frameworks such as TensorFlow, Caffe, or Pytorch.

For generating custom algorithms or functions, a LLVM-based C compiler is used, which utilized vector data types based on the OpenCL's kernel language model. The compiler performs scheduling, VLIW packing, register allocation and zero overhead looping, and DMA interrupts.

The development environment is an Eclipse-based IDE which provides a single, hierarchical and graphical view of the project. Simulator, profiler, and debugging functions are also integrated into the environment.

V. HOST CPU

A host CPU is required, primarily to allocate and manage workloads across the various accelerators, and to make decisions based on the resulting metadata from the processing results on these accelerators.

A wide variety of embedded CPUs are available, with ARM being the most popular vendor. Interfacing to accelerators will be primarily through AXI buses, although having shared access to external memory can also be useful.

The host CPU will also process external communications to the rest of the system, and perform system management functions. A secure gateway will often be required to ensure safe operation and protect the integrity of the communications channel to the external world. This can be implemented in a variety of methods, but an eFPGA is an interesting option. The eFPGA allows for hardware monitoring of the communications. Nearly all crypto attacks will involve a significant change in the statistics and timing of the communications messaging, which is difficult to detect in software but is easily monitored by hardware. And using programmable hardware allows for updates so that secure gateway can evolve over the ASIC product lifetime.

VI. CONCLUSIONS

An alternative to COTS technology for defense programs has been presented. Unlike traditional ASICs which are purpose build for a single narrow application, the programmable, heterogeneous ASIC supports the benefits of COTS technology with the efficiencies of custom ASICs. The summary of benefits is summarized as:

- Highest efficiency for specific types of workloads which require extreme amounts of processing. Ability to customize for defense applications and requirements.
- Unmatched architectural flexibility through heterogeneous nature of architecture, compared to COTS.

- Provides strategic processing advantage to the defense contractor and to the DoD, and will be unavailable to competitors or adversaries
- Highly resistant to cyber attacks through design and lack of available architectural information to adversaries.
- System updates supported throughout program lifetimes
- Applicability can be across multiple programs and weapons systems, reducing NRE for any one program, and increasing unit volume for the ASIC
- The un-programmed chip is unlike subject to ITAR restrictions, allowing manufacturing and supply chain flexibility. Custom firmware created by defense contractor is the sensitive part of system.

It is suggested that this new paradigm can provide an ideal solution compared to COTS or custom ASIC solutions in many of the fastest changing and very critical defense applications.

The heterogeneous ASIC concept also allows the defense community access to some of the most innovative processing architectures, which often come from small, nimble vendors who normally are not considered by the defense establishment. And in such a way that US DoD technology investments are not available to foreign rivals or adversaries.

VII. References

[1] Embedded Digital Signal Processing Innovations, GOMAC, March 2018 Michael Parker

[2] Microprocessor Report: Videantis IP Cores pack lots of MACs, The Linley Group, Feb 2018, Mike Demler

[3] Multi-GSPS FFTs using FPGAs, NAECON, July 2016 Michael Parker, Simon Finn, Hong Shan Neoh

[4] QR Decomposition using FPGAs, NAECON, July 2016 Michael Parker, Volker Mauer, Dan Pritsker