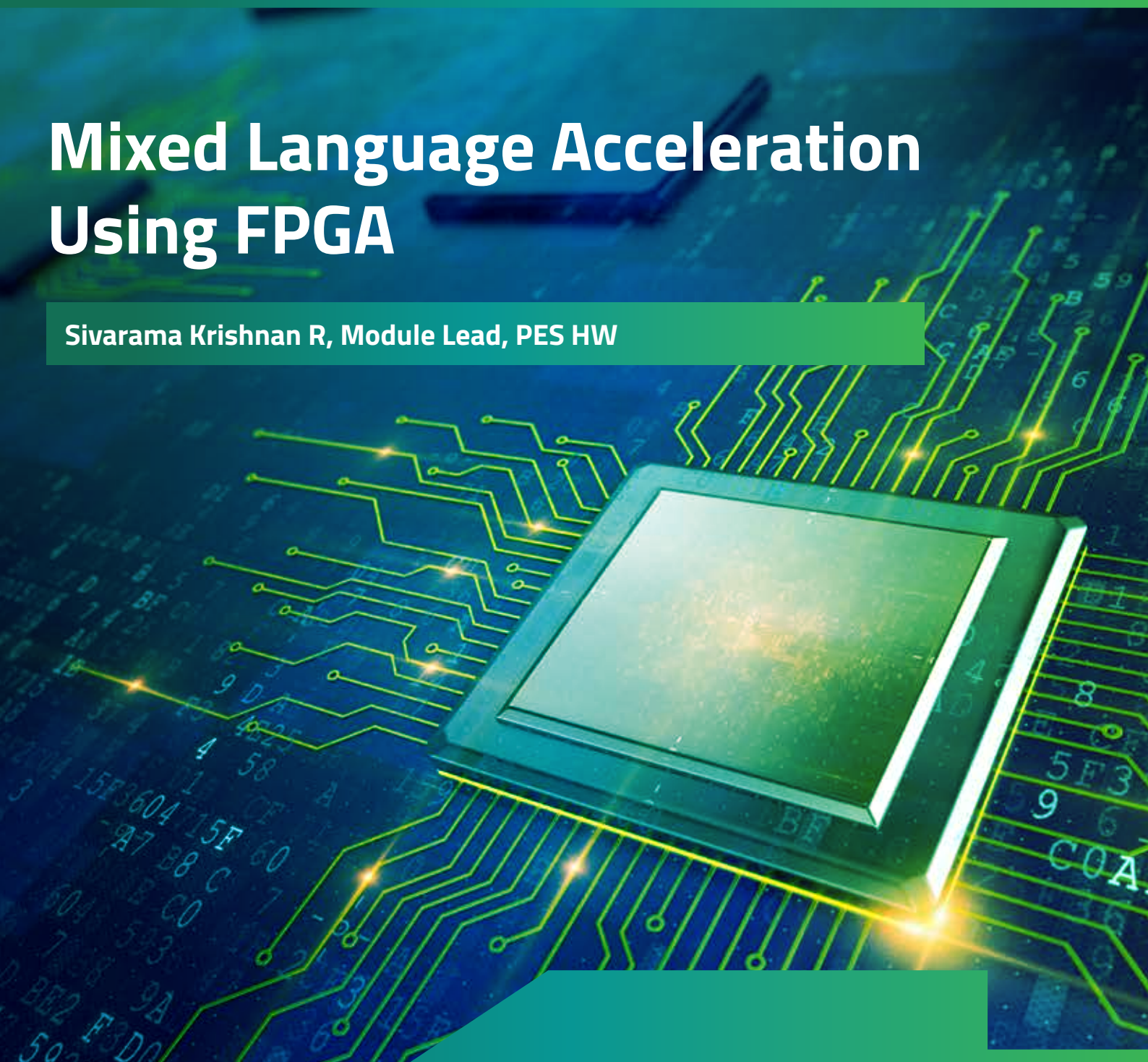
happiest minds
The Mindful IT Company

Mixed Language Acceleration Using FPGA

Sivarama Krishnan R, Module Lead, PES HW



Abstract

The fundamental question is, what does an FPGA offer in Heterogeneous computing? OpenCL™ based on the published Khronos Group Specification is the first open programming standard for cross-platform computing. Lately, FPGA vendors have developed tool chains supporting OpenCL compilers, adding another platform to the fore. The ability to perform compute intensive tasks in parallel using both HDL and OpenCL gives the FPGA platform an edge over the -CPU counterpart, in providing a coherent business solution.

Introduction

FPGA as a GPGPU

For a while, GPGPU has been able to offload process intensive tasks to the GPU, rather than relying on the CPU to do everything. OpenCL has provided an open and uniform programming environment for accelerated processing using the combined power of CPU and GPU under a single platform. With FPGA coming into the foreground, just not certain tasks but most tasks can be offloaded to the configurable FPGA platform. The versatility of HDL as a low-level language combined with a high-level language such as OpenCL can be used to design custom functions that are both timing-driven as well as area accurate.

Vendor Advantage

Xilinx with SDAccel™ (Figure 1) and Intel FPGA with AOCL™ have developed mature toolchains to process OpenCL codes for FPGA platforms. FPGA can act as standalone accelerating system either with inbulit SOC (ARM or Xeon) or a soft core processor system. This significantly enables higher performance at a much lower power than using other technologies.

SDAccel- CPU/GPU Development Experience on FPGAs

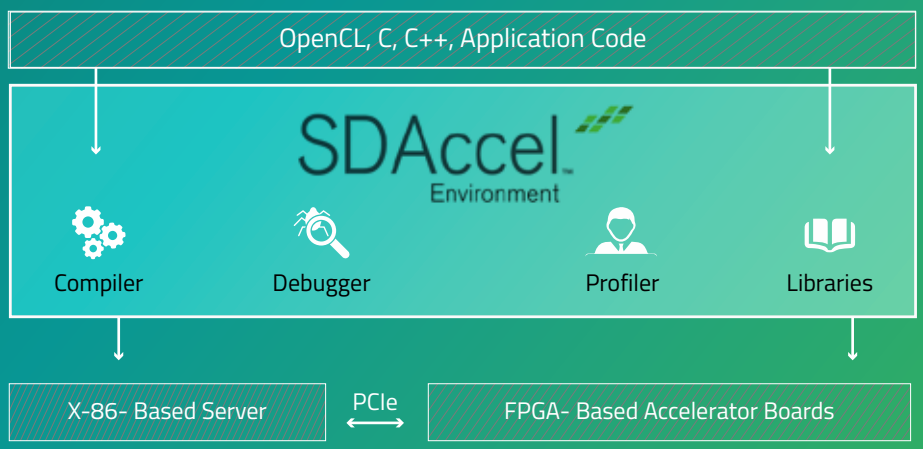


Fig. 1: Xilinx SDAccel, *[Image obtained from https://www.xilinx.com/products/design-tools/softwarezone/sdaccel.html in October 2017]*

The OpenCL potential

A platform-neutral language such as OpenCL enables coders to write generic code that can easily port to execute on different devices, cloud-based systems, and remote computing platforms. Whereas, non generic software developed for trade marked platform may or may not work with other vendors.

Applications

The OpenCL framework has been evolving over the years, the latest version being OpenCL 2.2 (Figure 2 shows the OpenCL timeline). We can notice that the number of platforms has also grown along with the revisions, thereby numerous pre-defined libraries have been introduced engaging a whole set of applications. Now, the idea is to make RTL engineers and software developers work together generating another set of libraries that could yield better results than just depending on the compiler. Dr. Randy Huang, FPGA Architect, Intel Programmable Solutions Group quotes, "FPGAs are powerful because they are adaptable and make it easy to implement changes by reusing an existing chip which lets a team go from an idea to prototype in 6 months versus 18 months to build an ASIC."

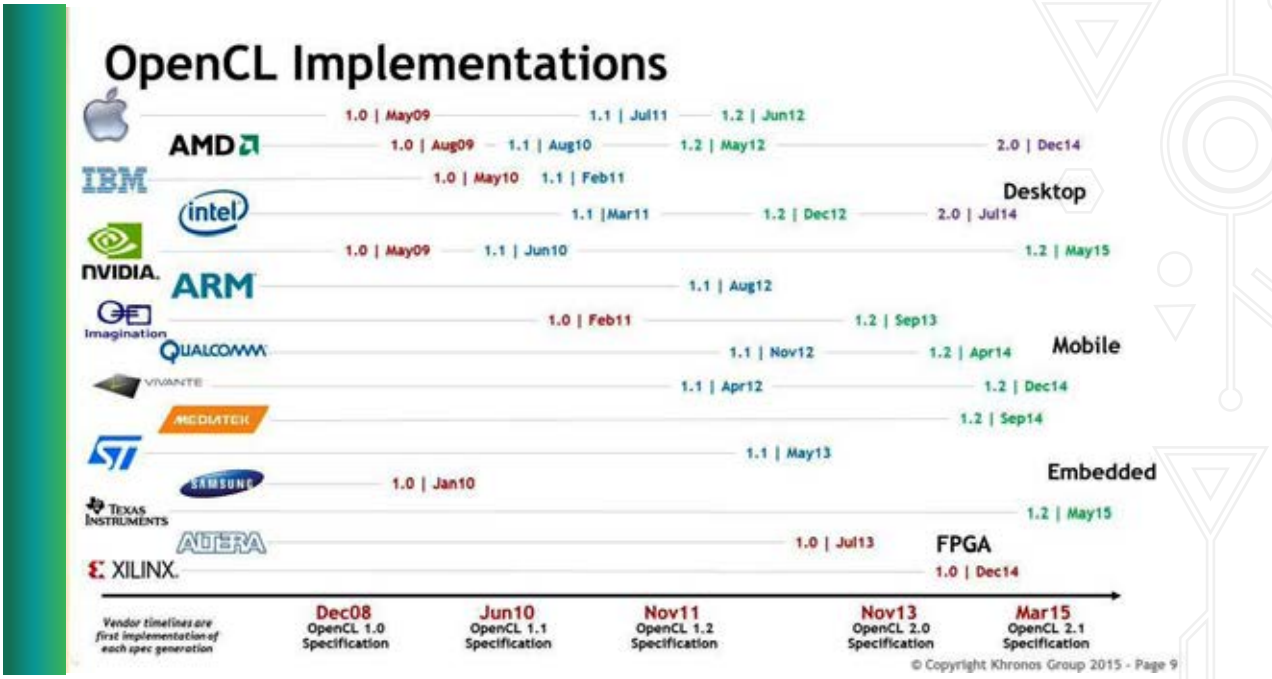


Fig. 2: OpenCL Timeline , *[© Khronos Group 2015, Page 9]*

- **MachineLearning**

  The data analytics industry is driven by Machine Learning algorithms. There has been an exponential growth of digital data propelling the need for analytics to yield the data meaningful. The classification of this data can be achieved effectively either by using CNN or DNN (both Intel FPGA and Xilinx have benchmarked their performances using image classification as an example) implementation of FPGA based OpenCL framework. Currently, FPGA implementations may not race ahead of GPU implementations, but as per the industry news, with evolving technology, the former might get close enough to GPU.

- **Data Center**

  Microsoft and Amazon have already gotten a head start, by stating that FPGA boards are being used for their cloud based services. Data from local search engines and web servers are being used for consumer specialized suggestions and ads. Moreover, storage being offered through cloud gives an opportunity for compression and encryption engine implementations offloaded to FPGAs. On the communication platform, FPGA can be used to create configurable switches, anomaly detection, and virtual packet processing applications.

The Mixed Language Model

As discussed, OpenCL provides an abstracted software flow which does not require any knowledge about the underlying hardware. This, in turn, accelerates time-to-market scenario. But, what if we could create custom HDL libraries and link them to the OpenCL framework? The Intel FPGA SDK for OpenCL supports both RTL & OpenCL based libraries and, the typical tool flow is depicted in Figure 3. The mixed language model proposes a balanced approach for acceleration where a certain part of the OpenCL code may be implemented using RTL based functions. This gives developers options to create custom accelerator functions that run on Intel FPGAs. An ADC data processing system based case study is discussed in the sections to follow. Readers are requested to go through the AOCL programming guide [2] for full details.

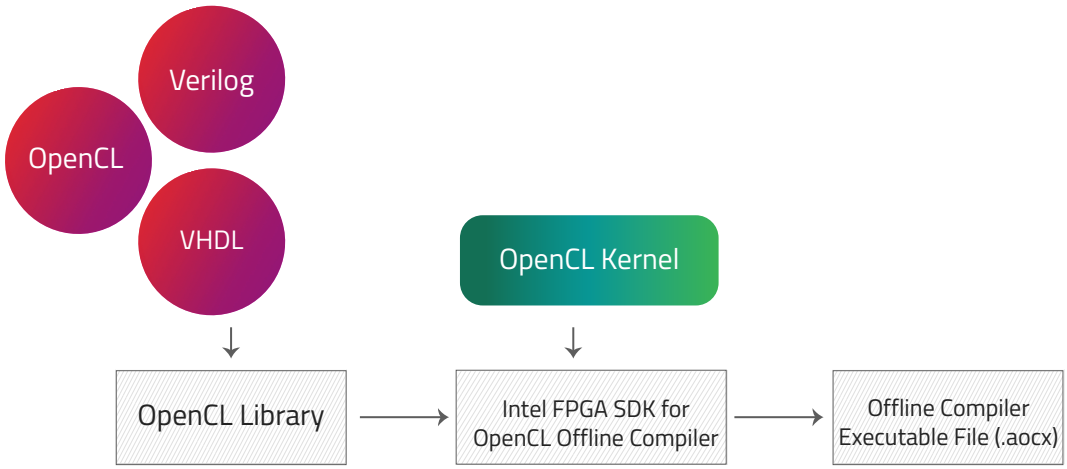


Fig. 3: Intel FPGA Library Flow, *[Image obtained from aocl programming guide [2] in October 2017*

- **An OpenCL library**

  An OpenCL library is a file that may contain multiple functions. Each function shall comprise of data processing logic that may work at any clock frequency. You can create an OpenCL library either in OpenCL or RTL or both. This library file shall be included along with the kernel/s file for hardware generation using the prescribed vendor tool. A library function can be called like any other normal C function inside yourOpenCL kernels.

  The advantages of using RTL as a function are given below:

  - Timing optimized, area-constrained and verified RTL.
  - Implementation of OpenCL kernel functionality that you cannot express effectively in OpenCL

- **ADC data processing system as a case study**

  Let us consider a PCIe based real-time data acquisition and processing card with an FPGA as a co-processor available to the CPU. The processor shall handle the movement and plotting of data and, the FPGA shall handle the actual data processing. This is a candidate for an OpenCL framework since the CPU can offload the heavier tasks to FPGA while it handles the reception and analysis of data using a User Interface. An FIR filter is one of the main components in a data processing chain. implementation is shown in figure 4. Writing an FIR filter OpenCL kernel might be a considerable effort, hence, RTL engineers can use the already available IP from the vendor, which can be easily simulated and verified

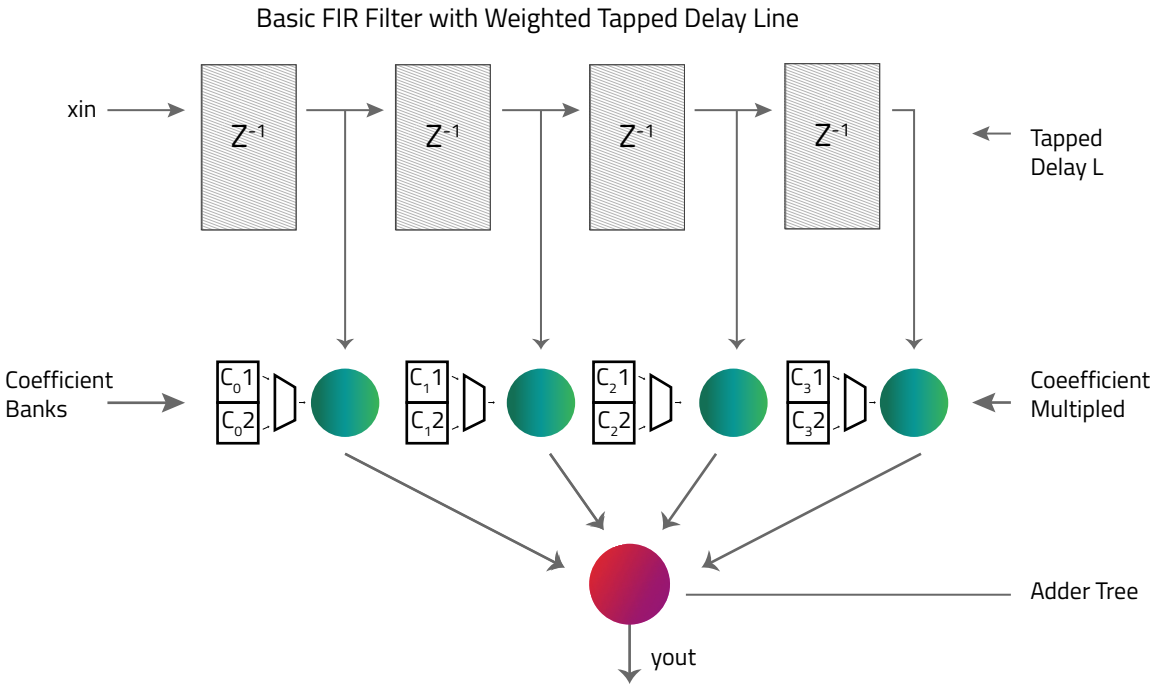


Fig. 4: FIR Filter, *[Image obtained from FIR filter user guide [3] in October 2017]*

- **RTL function in the OpenCL pipelined architecture**

An FIR filter Kernel is shown in Figure 5, with the RTL unction - FIRFilter(). The kernel FilterKernel accepts an integer data ADCIn and produces a floating point output FilterOut. The RTL function is responsible for the filtering of data and, the OpenCL code shall be a wrapper handling memory access of data. As we all know, OpenCL kernels are synthesized into a pipelined architecture by the AOCL tool [2]. This architecture yields a fixed latency but constant throughput every clock cycle. On the other hand, the specifics of an RTL function is unknown to the AOCL tool since it is just a black box. Hence, the question that remains is that, how does an RTL function satisfy the above criterion of latency and, how do we fit the module in the pipeline?

```
extern float FIRFilter(int);

void kernel FilterKernel (global int* ADCIn, global float*
FilterOut) {

int gid = get_global_id (0);

int a = ADCIn[gid];

FilterOut[gid] = FIRFilter (a);

}
```

Fig. 5: FIR Filter Kernel

Basically, there are three features which an RTL module should quantify so that it could be included as a function and they are given below:

- The module shall be able to work with the kernel clock.
- The module shall have handshake signals which generate the variable latency for the OpenCL kernel.
- If the module has a fixed latency, handshake signals are not required.

These features shall then be included as a part of the module description in an XML file [2]. A balanced latency allows the threads of the RTL module to execute without stalling the pipeline. Suppose the filter module has a fixed latency of 4 cycles, the OpenCL synthesized structure shall look like Figure 6.

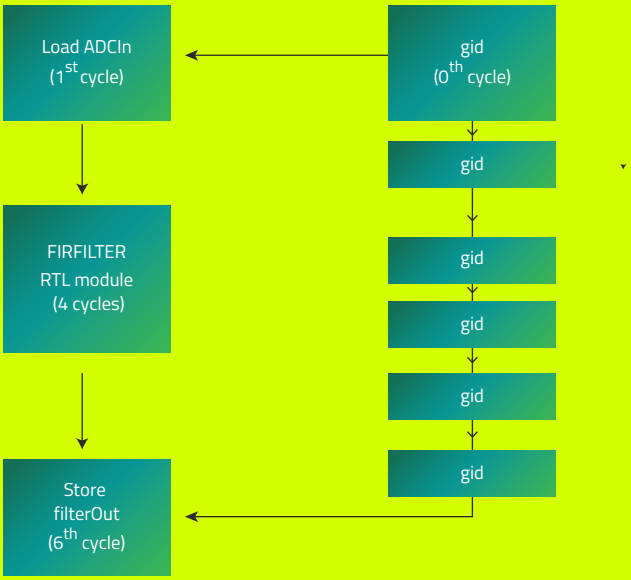


Fig. 6: OpenCL pipeline structure, *A representive structure that could be produced by Intel FPGA AOCL tool*

The global ID (gid) is accessed in the zeroth clock cycle and correspondingly the data is accessed in the next. Now, with a function latency of 4 cycles, the AOCL tool applies an additional 4 clocks pipeline to the global ID to satisfy the RTL function latency as per the XML description of the module. Hence, with a fixed latency we shall be able to achieve the required through put after an initial delay.

RTL component in an OpenCL library

As seen above, RTL module can be placed in an OpenCL pipeline but, there are restrictions and constraints that need to be followed for a successful integration. A step by step description is available in the AOCL programming guide [2]. Also, the RTL should be verified separately before the integration.If the above steps are followed precisely we would achieve a robust system.

Conclusion

There has been a constant debate on when to use an accelerator or how much of an acceleration is necessary. FPGA as an accelerator has grown from a novice device to a major player in recent times and the convergence of platforms might be the norm for the future. The discussion in this paper hasa tradeoff with respect to the portability of the OpenCL code, since it uses custom low-level RTL code as functions but, this will definitely yield better results in an FPGA driven acceleration environment.

References

- FirePro OpenCL Whitepaper.pdf
- AOCL programming guide.pdf
- FIR filter user guide

Author

Sivarama Krishnan R is an FPGA engineer, with several complex FPGA designs to his credit. He has several years of experience in FPGA Technology, RTL design, Verification and Validation. He holds a Bachelor's degree in Electronics and Communication Engineering from VTU, Karnataka.

Happiest Minds has expertise in developing FPGA based designs using devices ranging from low density CPLDs to multi-million gate FPGAs/ SOCs. Our design team has profound experience working with FPGAs from leading vendors such as Actel,Altera, Lattice and Xilinx. Our end-to-end capability in FPGA design and quick development cycle positions us to be the right partner for any FPGA based design activity.

Happiest Minds enables Digital Transformation for Enterprises and Technology providers by delivering seamless Customer Experience, Business Efficiency and Actionable Insights through an integrated set of Disruptive Technologies: Big Data Analytics, Internet of Things, SDN & NFV, Mobility, Cloud, Security, Unified Communications, etc. Happiest Minds offers domain centric solutions applying skills, IPs and functional expertise in IT Services, Product Engineering, Infrastructure Management and Security. These services have applicability across industry sectors such as Retail, Consumer Packaged Goods, Ecommerce, Banking, Insurance, Hi-tech, Engineering R&D, Manufacturing, Automotive and Travel/Transportation/Hospitality. Headquartered in Bangalore, India, Happiest Minds has operations in the US, UK, Singapore, Australia and has secured $63 million Series-A funding. Its investors are JPMorgan Private Equity Group, Intel Capital and Ashok Soota.