

Alveo, Vitis and Libraries Product Overview

Rajiv Jain

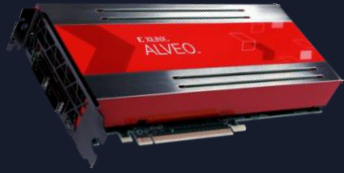
Data Center Group

Oct 2019

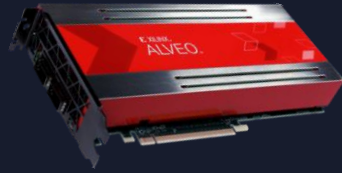


Xilinx Alveo Product Lineup

ALVEO™ U200



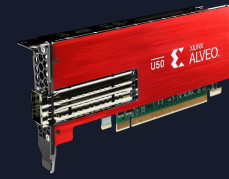
ALVEO™ U250



ALVEO™ U280



ALVEO™ U50



UltraScale+ Architecture

UltraScale+ Architecture

UltraScale+ Architecture

UltraScale+ Architecture

1,182k LUTs

1,728k LUTs

1,304k LUTs

872k LUTs

Dual slot, full height

Dual slot, full height

Dual slot, full height

Single slot, half height

64GB DDR, 77GB/sec

64GB DDR, 77GB/sec

8GB HBM2, 460GB/sec

8GB HBM2, 460GB/sec

PCIe Gen3

PCIe Gen3

PCIe Gen3, Gen4, CCIX

PCIe Gen3, Gen4, CCIX

2x QSFP 28 (100GbE)

2x QSFP 28 (100GbE)

2x QSFP 28 (100GbE)

1x QSFP 28 (100GbE)

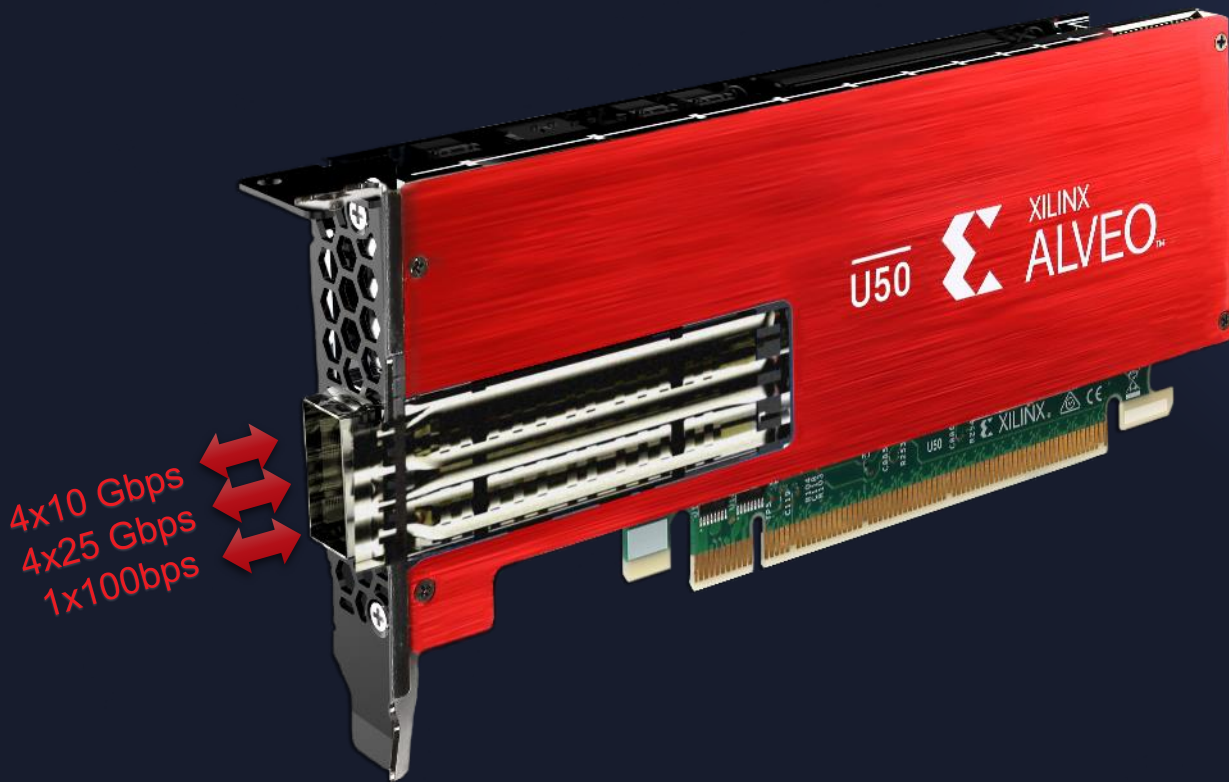
< 225W

< 225W

< 225W

< 75W

➤ Xilinx Alveo U50 Accelerator



UltraScale+ Architecture

Low-profile form factor

8GB HBM2 Memory, 460GB/sec

PCIe Gen4, CCIX, PCIe Gen3

QSFP 28 (100GbE)

< 75W

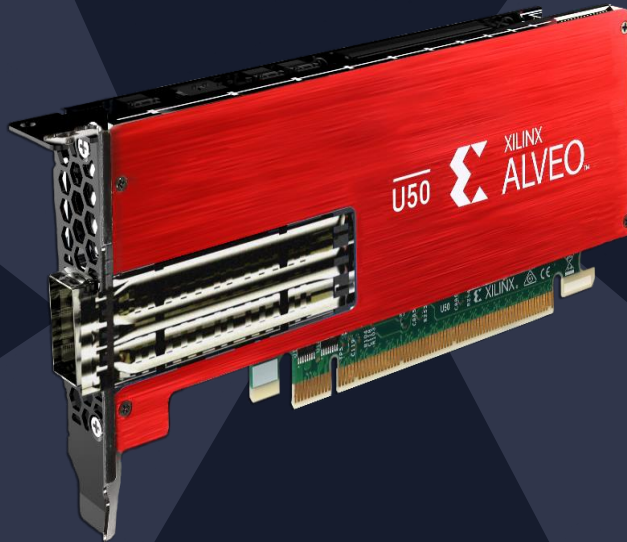
Alveo U50 – Low Profile Acceleration Card



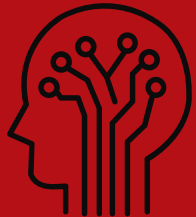
Fintech



Storage



Machine Learning



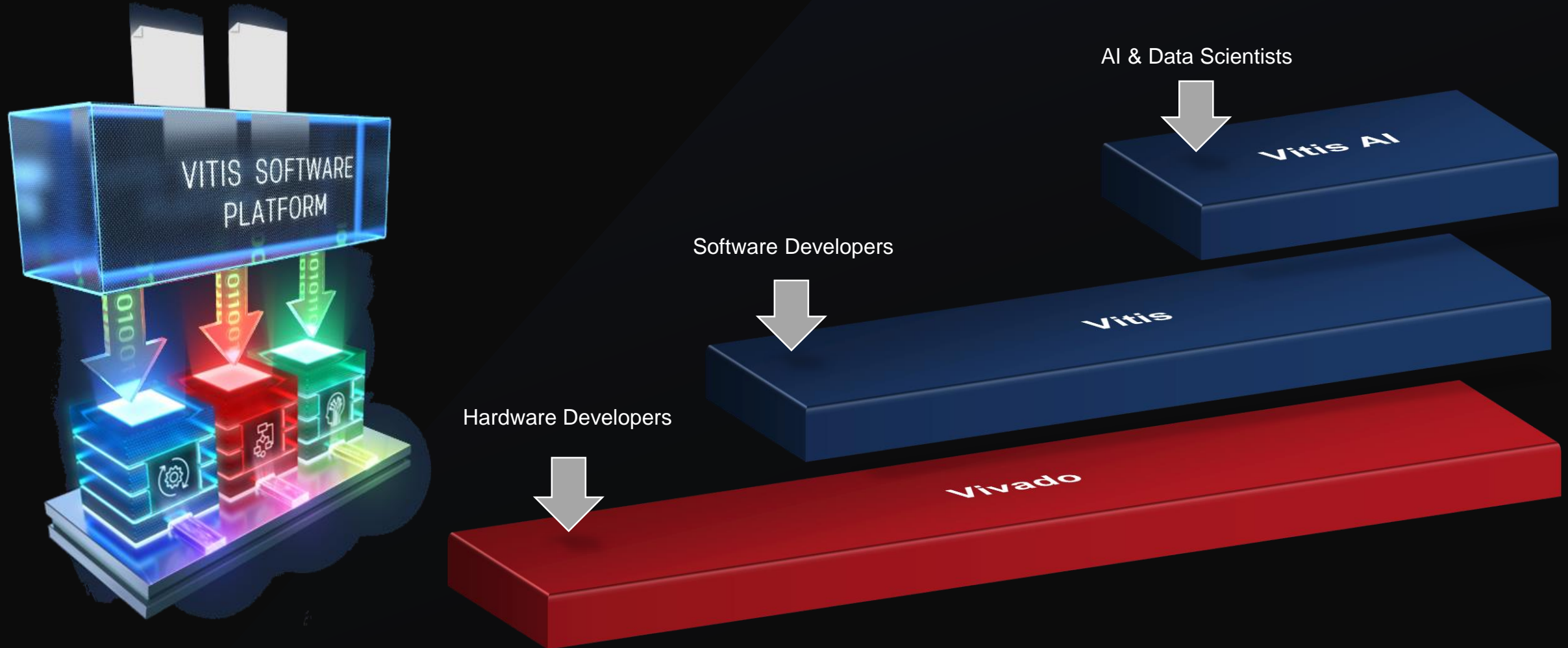
Database



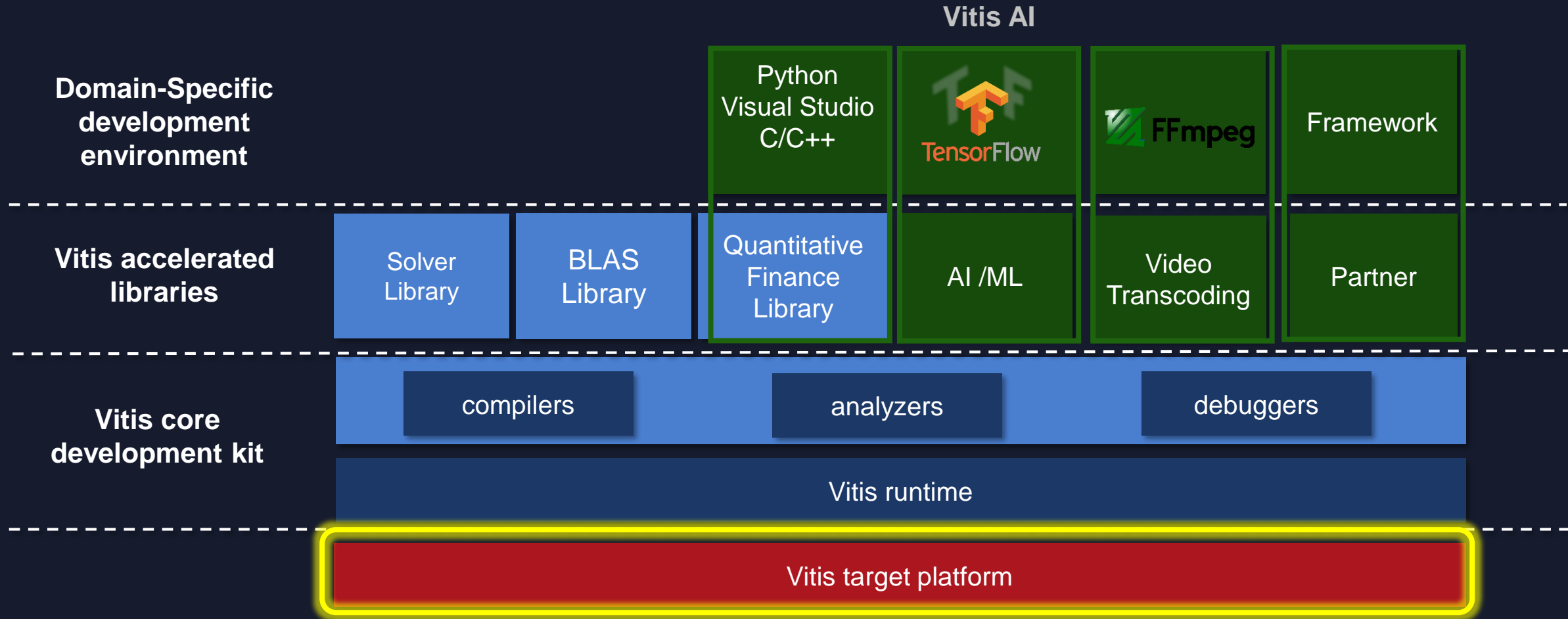
Card	Alveo U50
Primary Application	Fintech + Storage + Database + ML
FPGA Design	XCU50
CCIX	Yes
Device VCCINT	0.85V
Width	Single slot
Form Factor (Passive)	HHHL
Memory Target	8 GB HBM
Memory Config	Dual Stack, 32 pseudo-ports
PCIe	2x Gen4x8, 1x Gen4x8, Gen3x16, CCIX
Network I/F	2x SFP- DD* or 1X QSFP28
Thermal	Passive
Power (Max TDP)	75W
KLuts	872K

* During ES, U50 card will have 2 SFP-DD ports

VITIS – Unified Development Platform for ALL Developers (FPGA / ACAP / SOC) -- Free!

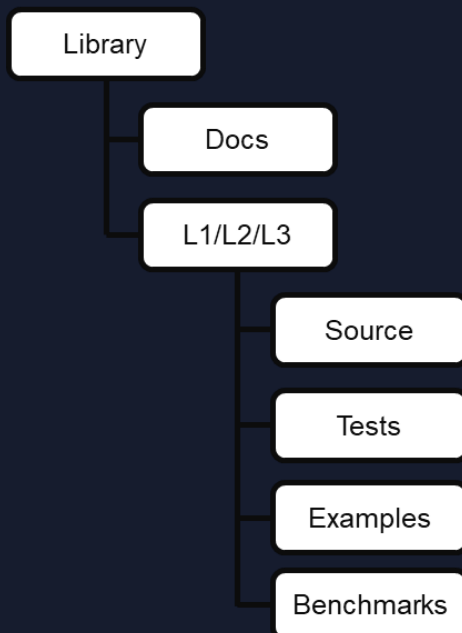


Vitis: Unified Software Platform



Vitis Accelerated Libraries – What?

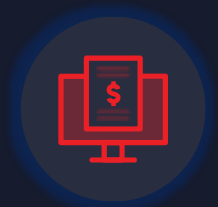
> Open-Source, performance-optimized libraries offering out-of-the-box acceleration.



Vision and Image



Quantitative Finance



Data Analytics and Database



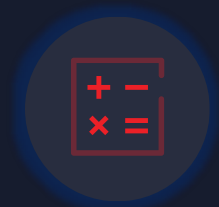
Data Compression



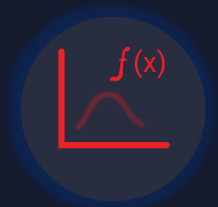
Data Security



AI



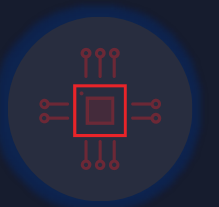
Math



BLAS



Statistics



Solver



Data Management

Xilinx Vitis Quantitative Finance Library

Equity Product

Credit Product

Interest Rate Product

Commodity Product

FX Product

Black-Scholes
Heston

European
American

Asian
Barrier

Digital
Cliquet

Linear Algebra
Cholesly Decomp
LU/SVD/QR Decomposition
Dense Matrix Multiply
Space Matrix Multiply
...

Statistics
Random Number Generator
Box-Muller Transform
Distributions
...

Financial
Finite Difference
Monte-Carlo Methods
Brownian Bridge
Closed Form Solutions
Greeks/Sensitivities
...

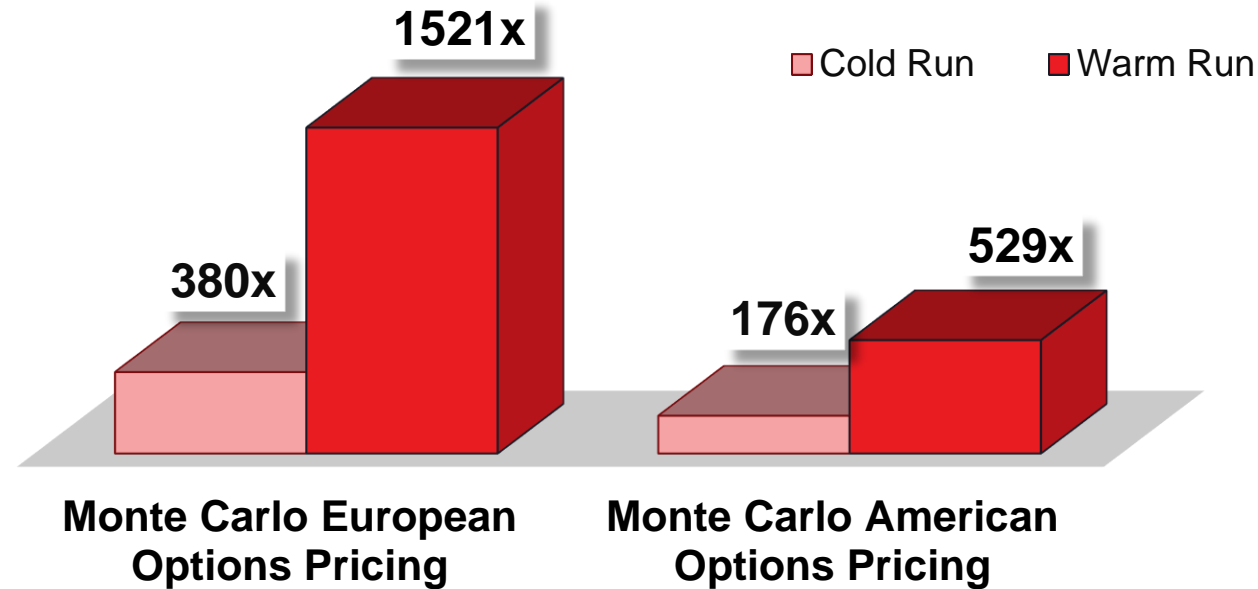
Solver
Tridiagonal Solver
Pentadiagonal Solver
PDE Craig Sneyd

Basic Math Function

sqrt, abs, fabs, exp, log, pow, sin, cos, asin, acos, sinh, cosh, floor, fmod, modf, etc.

Quantitative Finance Library (NON-STAC Benchmark)

Speed-Up Vs. QuantLib on CPU



Monte Carlo European Options Pricing		
	Cold Run	Warm Run
QuantLib	20.155 ms	20.155 ms
Vitis Quantitative Finance Library	0.053 ms	0.01325 ms
Speed-Up	380X	1521X

Monte Carlo American Options Pricing		
	Cold Run	Warm Run
QuantLib	1038.105 ms	1038.105 ms
Vitis Quantitative Finance Library	5.87 ms	1.96 ms
Speed-Up	176X	529X

CPU : 2 Intel(R) Xeon(R) CPU E5-2690 v4 @3.20GHz, 8 cores per processor and 2 threads per core.

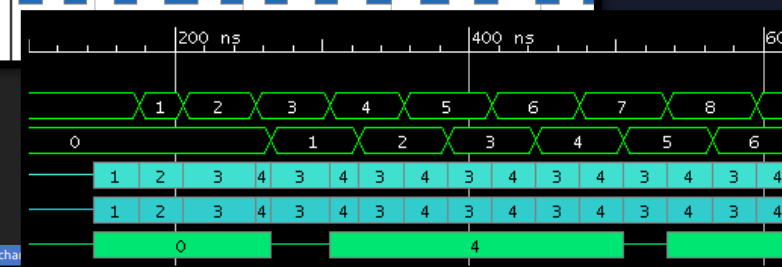
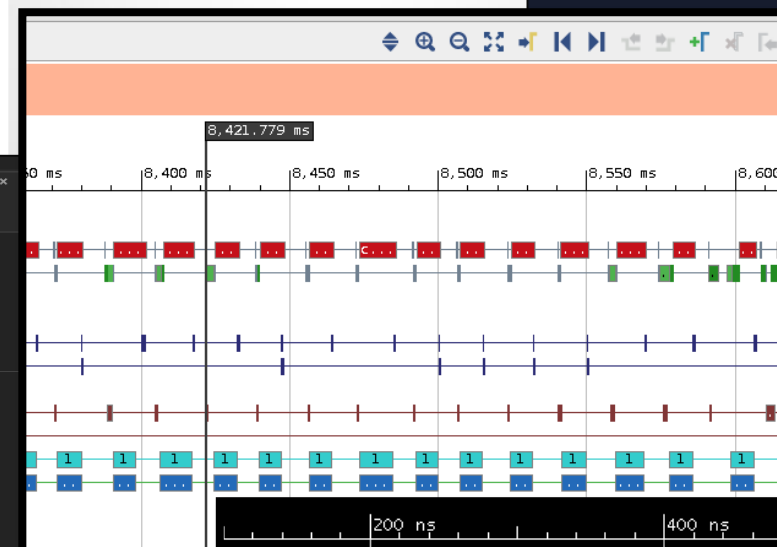
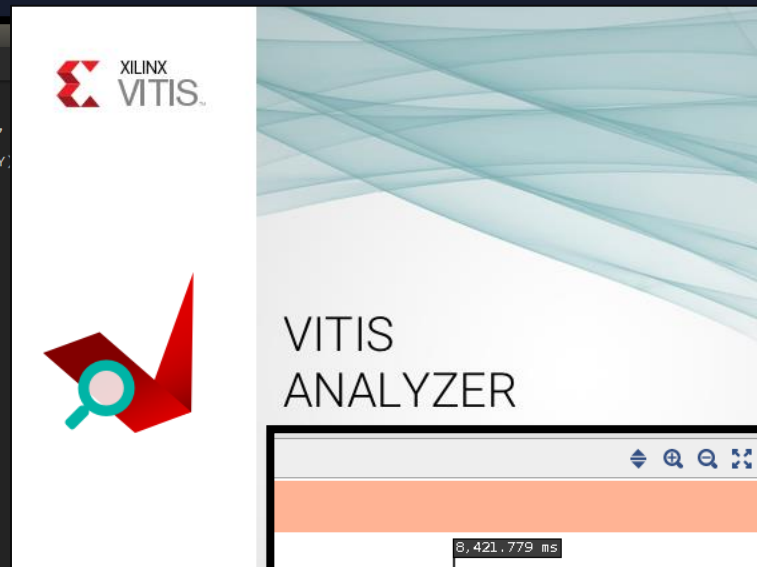
Xilinx : Vitis Quantitative Finance Library v1.0 running on 1 Alveo U250

Cold Run: Pricing Engine starts up in response to a request.

Warm Run: Pricing Engine is already running, with sufficient memory allocated to handle the request

Vitis -- Familiar SW Environment

```
365         if (!first) {
366             cv::Mat hw_image(IMAGE_HEIGHT, IMAGE_WIDTH, CV_8UC2,
367                             cv::Mat_<_> color;
368             cv::cvtColor(hw_image, color, cv::COLOR_YUV2BGR_UYVY);
369             cv::imwrite("out.png", color);
370             webcam_return_queue.push(cbbuf);
371         }
372     }
373
374     resizer_thread.join();
375     std::cout << "Resizer joined" << std::endl;
376     stop_cam = true;
377
378     break;
379 }
380
381 int frame;
382 if (!first) {
383     if (pflip_data.used_frames.try_pop(frame)) {
384         for (int i = 0; i < NUM_BUFS; i++) {
385             if (bufs[i]->id == frame) {
386                 webcam_return_queue.push(bufs[i]);
387                 break;
388             }
389         }
390     }
391 }
392 cbbuf->
393     if (cb
394         count
395         data
396         fd
397         id
398         drm.fl
399         initialized
400     } else {
401         num_fa
402         ocl_initialized
403         vbuf
404         first = false;
405     }
406     string(num_faces);
407 }
```



Further simplify coding for FPGA in HLS



Application

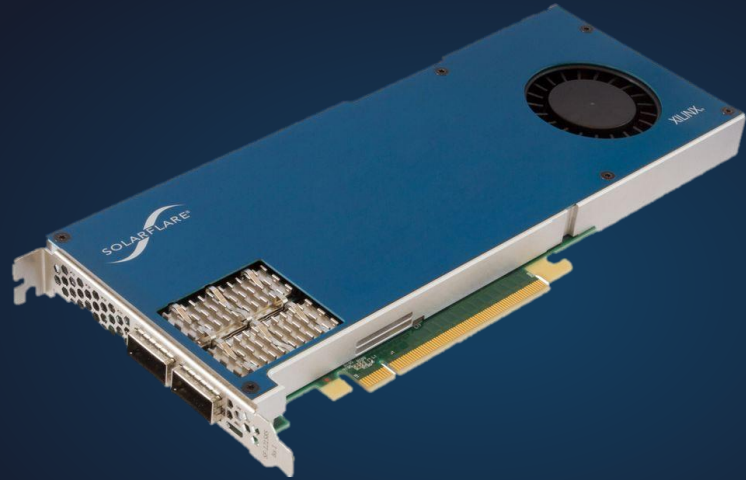
```
void vadd(  
    const unsigned int *in1,  
    const unsigned int *in2,  
    unsigned int *out,  
    int size  
)  
{  
#pragma HLS INTERFACE m_axi port=in1 offset=slave bundle=gmem  
#pragma HLS INTERFACE m_axi port=in2 offset=slave bundle=gmem  
#pragma HLS INTERFACE m_axi port=out offset=slave bundle=gmem  
#pragma HLS INTERFACE s_axilite port=in1 bundle=control  
#pragma HLS INTERFACE s_axilite port=in2 bundle=control  
#pragma HLS INTERFACE s_axilite port=out bundle=control  
#pragma HLS INTERFACE s_axilite port=size bundle=control  
#pragma HLS INTERFACE s_axilite port=return bundle=control  
  
    unsigned int v1_buffer[BUFFER_SIZE];  
    unsigned int v2_buffer[BUFFER_SIZE];  
    unsigned int vout_buffer[BUFFER_SIZE];  
  
    for(int i = 0; i < size; i += BUFFER_SIZE)  
    {  
        int chunk_size = BUFFER_SIZE;  
        if ((i + BUFFER_SIZE) > size)  
            chunk_size = size - i;  
  
        read1: for (int j = 0 ; j < chunk_size ; j++){  
            v1_buffer[j] = in1[i + j];  
        }  
        read2: for (int j = 0 ; j < chunk_size ; j++){  
            v2_buffer[j] = in2[i + j];  
        }  
  
        vadd: for (int j = 0 ; j < chunk_size; j ++){  
#pragma HLS PIPELINE II=1  
            vout_buffer[j] = v1_buffer[j] + v2_buffer[j];  
        }  
        write: for (int j = 0 ; j < chunk_size ; j++){  
            out[i + j] = vout_buffer[j];  
        }  
    }  
}
```

Accelerator Code (Before)

```
void vadd(  
    const unsigned int *in1,  
    const unsigned int *in2,  
    unsigned int *out,  
    int size)  
{  
    for (int i = 0; i < size; i++)  
        out[i] = in1[i] + in2[i];  
}
```

Accelerator Code (After)

Solarflare – now a Xilinx Company



Enterprise enabled next generation
SmartNICs powered by Onload

Market Leading Low Latency NIC ready for Enterprise



World Class Networking Talent
Key Technologies:

Application acceleration
Server security
Network adapters

Adaptable.
Intelligent.

