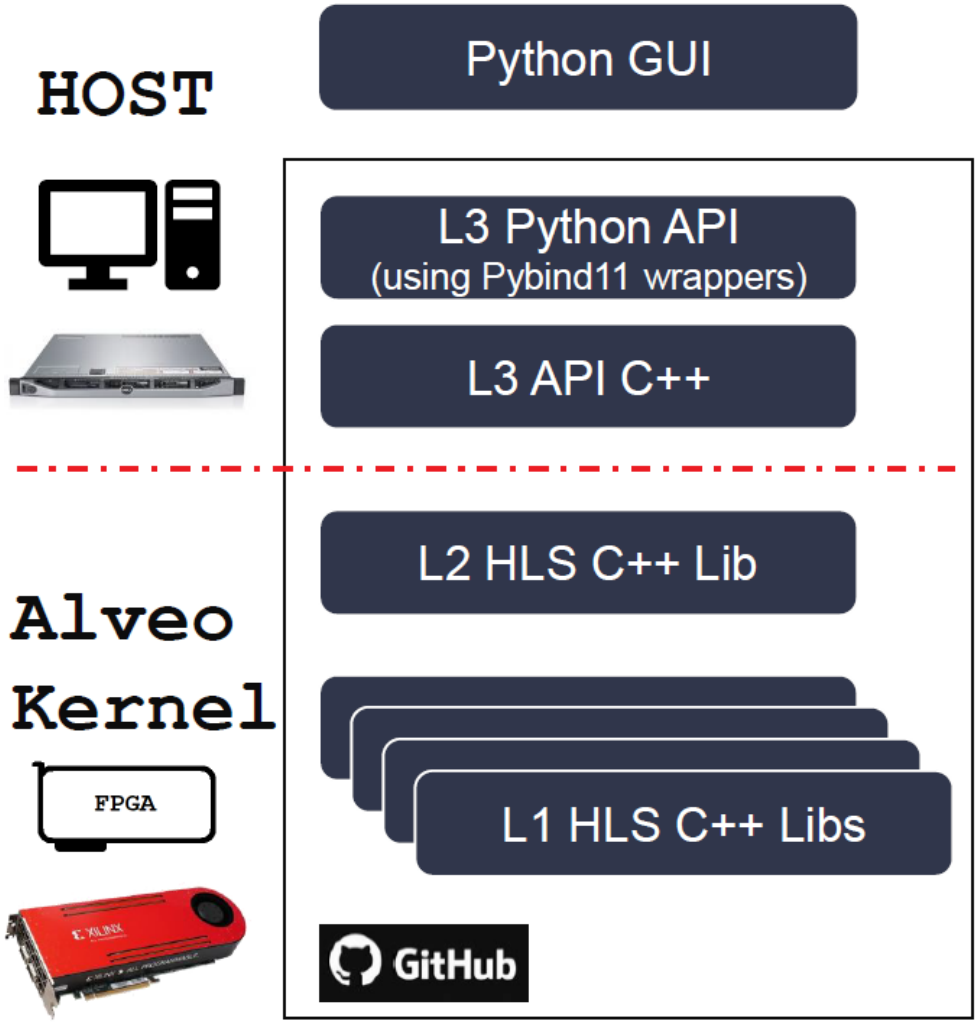
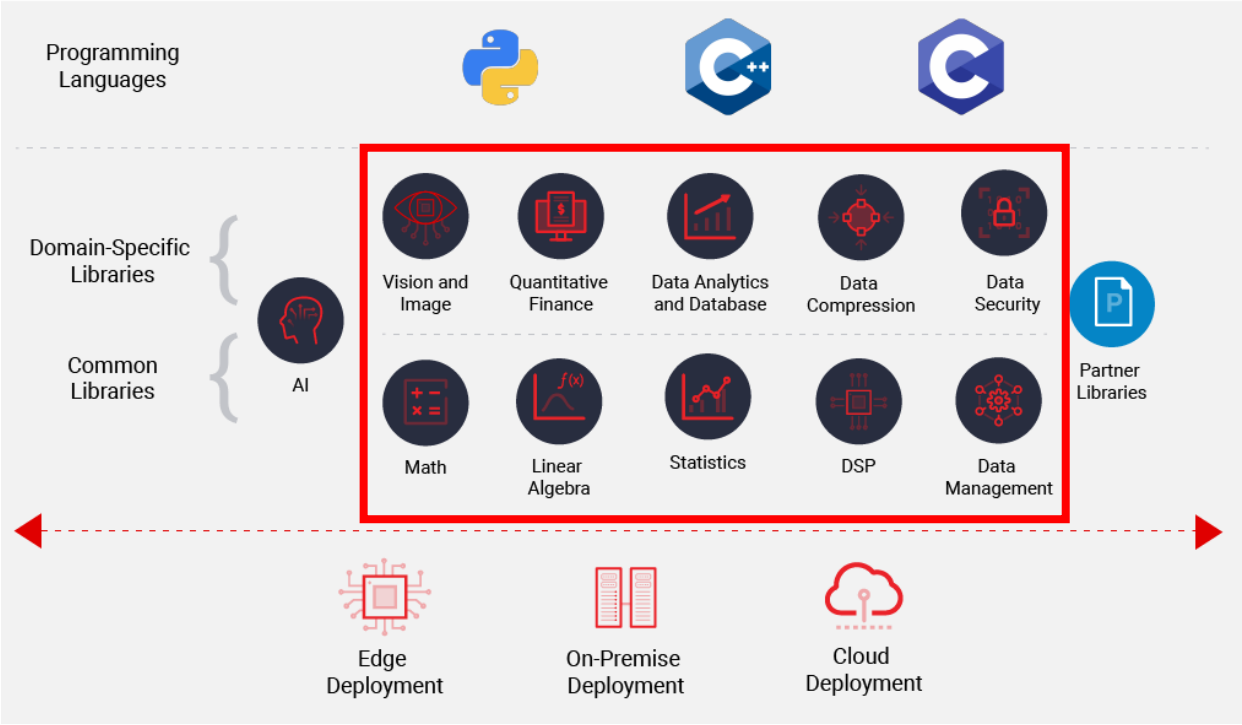
Lab-C Note

Lab#C (team) – Vitis Library





Step 1:
Identify what library to work on

Note: Each team works on different library

Vitis BLAS Library	blas
Vitis Codec Library	codec
Vitis Database Library	data_analytics
Vitis Data Analytics Library	data_compression
Vitis Data Compression Library	database
Vitis DSP Library	dsp
Vitis Graph Library	graph
Vitis HPC Library	hpc
Vitis Quantitative Finance Library	quantitative_finance
Vitis Security Library	security
Vitis Solver Library	solver
Vitis SPARSE Library	sparse
Vitis Utilities Library	utils
Vitis Vision Library	vision

Documentation: https://xilinx.github.io/Vitis_Libraries/
Github: https://github.com/Xilinx/Vitis_Libraries

Step 2. Identify L3 topic

- Look for L3, there are Benchmark or Examples from the Github
- Identify which benchmark or example to work on, for example in Vision Library https://github.com/Xilinx/Vitis_Libraries/tree/master/vision/L3
- /Benchmark
  - Blobfromimage
  - colordetect
- /Examples:
  - Cornertarcker
  - Gaussiandifference
  - ….

Step 3. Identify L2/L1 functions to perform L3 function selected

- Once L3 function is selected, track its function hierarchy
- Identify all the L2, L1 modules to implement L3
- Refer to

https://github.com/bol-edu/2021-fall-ntu/tree/main/Lab_C_Vitis_Libraries/Corner_Tracking_with_Optical_Flow/HLS_2021_FALL_LABC-master

Its ppt:

https://github.com/bol-edu/2021-fall-ntu/blob/main/Lab_C_Vitis_Libraries/Slides/team1_lab_c.pptx

Step 4. Validation plan

- Identify test program
- Identify test data
- Validation platform: PYNQ or U50

Step 4. Submit a page of report by 3/28

Provide the following information

Library:

L3 Function:

Modules: list of modules to implement L3

Validation: Test program, test data, and validation platform PYNQ or U50)

INTELLIGENT
COMPUTING

Vitis Library

Purpose

- Learn Coding Style for IP development - C++ class/template
- Learn kernel (HLS) + Datamover
- Host Validation Flow
- Analysis methodology
- A resource of hardware accelerated IP to leverage, to optimized for future need.

Disclaimer: I did not run the library, you will explore it.

Three Level of Development

- L1: Primitive Functions
  - HLS module level c-sim, synthesis, co-sim
  - HLS code can be leveraged by FPGA hardware developer
- L2: Kernels
  - Host code integrated with XRT
- L3: Software API
  - C/C++/Python APIs to allow pure software developer to offload acceleration functions

Note: Some L3 API use other libraries, consult other team for that library

Requirement and Delivery

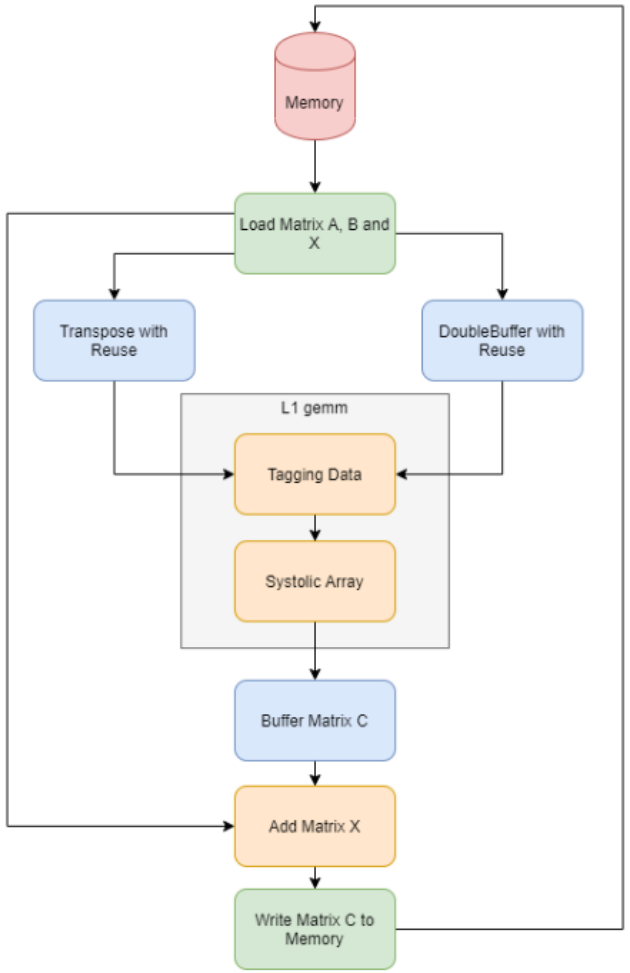
- Selected Primitive function
- Lab work
  - L1 test: c-sim, synthesis, cosim
  - L2 Kernel function test
  - L3 API – benchmark
- Report
  - Background introduction
  - Show lab result
  - Analysis
  - Suggestion for improvement

BLAS – Basic Linear Algebra Subroutines

- Primitive function: GEMM
- L1 – c-sim, synthesis, cosim)
- L2 - Benchmark – streamingKernel
- L3 – StreamingKernel

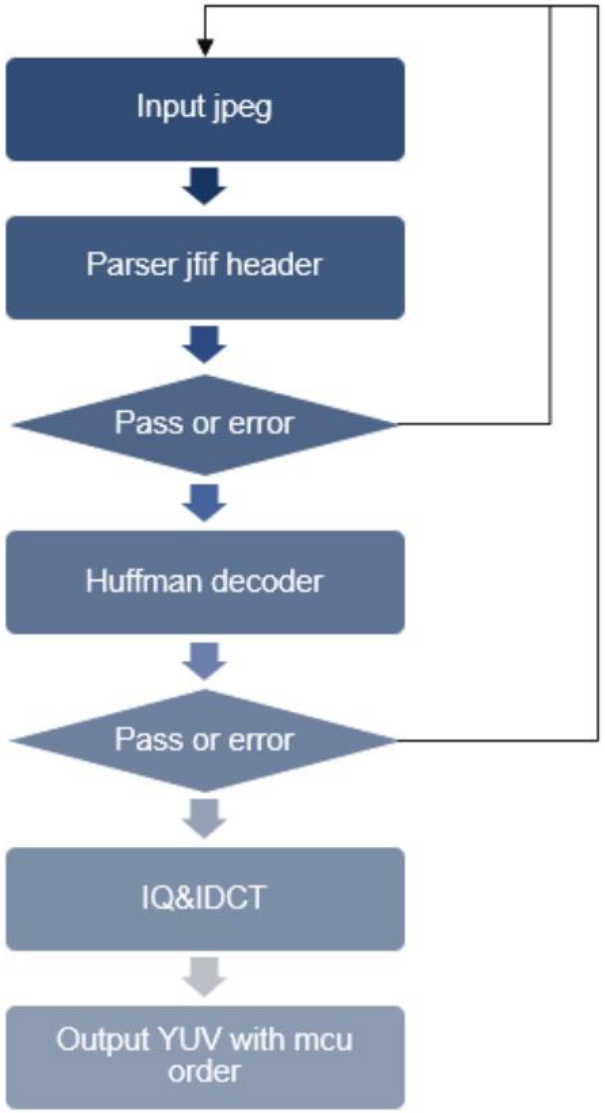
function	decription
amax	find the poistion of max in a vector
amin	find the position of min in a vector
asum	sum of the magnitude of vector elements
axpy	compute Y = alpha * X + Y
copy	Y = X
dot	dot product of vector x and y
gbmv	banded matrix-vector multiplication y=alpha*M*x + beta * y
gemv	general matrix-vector multiplication y = alpha*M*x + beta * y
gemm	general matrix-matrix multiplicaion
nrm2	Euclidean norm of the vector x
scal	compute X = alpha * X
swap	swap vector x and y
symv	symmetric maxtric-vector mulitplication
trmv	triangular matrix-vector multiplication

DataMover for Matrix Storage format
row-based storage in a contiguous array
packed storage for symmetric and triangular matrics
banded storage for banded matrix



Codec

- Tw video decoder
  - JPEG Decoder – Sequential DCT-based mode
    - Jfifparser - Direct parser the JPEG file header
    - Jpegdecoder – Hoffman decoder
  - Pik Encoder – Google PIK



Database – Accelerate SQL query execution

- Primitive: Insert, merge sort
- L3: gqe::BloomFilter API
- L1 Test
  - L1 benchmark: compound_sort

xf::database	
premitive	Description
Stream	single-bit (end of operation) + main data stream
Scan	transfer data from DR to internal FPGA
Hash	Lookup3/Murmur3 hash function
Filter	expression-based filter
Evaluation	calcuate expression using data columns
Bloom	filter redundant data before join processing
Join	Hash-Join, Merge-Join
Group-by Aggre	group-aggregate operator for sorted data
Hash Partition	disribute giant table into multiple sub-tables
Sort	Bitonic, Insert, Merge Sort
Glue Logic	Combine and Split Columns

Data Analytics – xf::data_analytic

- L3: Regular Expression Acceleration
  - data pre-processing to natural language processing, pattern matching, web scraping, data extraction

Data Analtics		
group	function	
Classification	decisionTreePredict	
	axiVarColToStreams	AXI master to stream
	naiveBayersTrain	load training dataset, cout frequency, output likehood probability matrix
	naiveBayersPredict	classification by argmax function
	svmPredict	load weight,sampe, output classificaiton id
clusterig	KMeansPredict	cluster idex for each sample
Regression	decisionTreePredict	load tree, output category id
Text	editDistance	Levenshtein distance
	regexVM	regular expression VM (1 instruction per iteration)
	regexVM_opt	regular expression VM (2 instruction per iteration)
DataFrame	readFromDataFrame	read data from DDR, pack into object streams
	writeToDataFrame	write object stream data to DDR
Software	xf_re_compile	pre-compiling input regular expression
Class	LogisticRegression	linear least square regression predict
	linearLeaastSquare	linear least square regression predict
	LASSORegressionPredict	LASSO
	ridgeRegression	

Data Compression: xf::compression

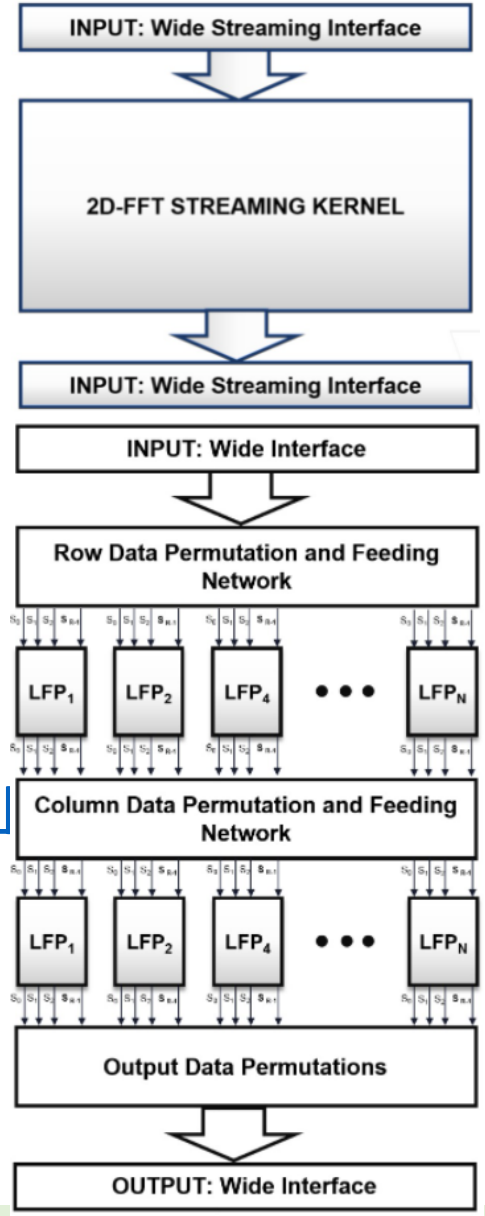
- L3: LZ4 Application

Data Compression	
function	Description
blockPacker	packs compressed data
huffmanDecoderLL	zlib/gzip fixed dynamic and stored block , huffman ecoded data, generate decoded data in LZ77
huffmanDecoder	
huffmanEncoderStream	zlib/gzip dynamic hoffman encoding
lz4Compress	separate input into literal stream, and offset stream
lz4Decompress	
lzDecompress	
lzMultByteDecompress	
lzBestMatchFilter	pick character with higher match length
lzBooster	impress compression ratio - max match
lzFilter	
snappyCompress	snappy algorithm
snappyDecompress	
zstdCompressStream	wrapper - ZSTD compressed
zstdDecompressStream	
zstdDecompressCore	

DSP

- 1D SSR (Super Sample Rate) FFT
  - Multi-instance
  - Data Type
- 2D SSR FFT
- Run example – use GUI

https://github.com/Xilinx/Vitis_Libraries/tree/master/dsp/fir_129t_sym



Graph

- Choose one function/API from L1, L2, L3 examples

Graph	
Function	Description
L1	
desnseSimilarity	similarity function for dense graph (Jaccard, Cosine)
generalSimilarity	dense and sparse graph (Jaccard, Cosine Simularity)
sortTopK	sort top k function
sparseSimilarity	fo sparse graph
L2	
bfs	breath-first search algorithm
calcuDegree	calculate degree algorithm
calcuWeightedDegree	calculate weighted degree algorith
connectedComponents	compute connected component membership of each vertex for undirected graph
convertCsrCsc	Convert Csr Csc algorithm
labelPropagation	label propagation algorithm
kernelLouvainTop	louvain kernel implement
pageRankTop	pagerank algorithm
singleSourceShortest	single source shortest path, matrix in CSR format
stronglyConnectedComponents	
triangleCount	matrix in CSC format
twoHop	find how many 2-hop pathes between two vertics (CSR)
L3 https://xilinx.github.io/Vitis_Libraries/graph/2021.1/guide_L3/api.html	

HPC – High Performance Computing

- RTM (Reverse Time Migration) – seismic imaging
  - Stencil2D/RTM2D, Stencil3D/RTM3D
- CG (Conjugate Gradient Solver) with Jacobi preconditioner
  - iterative method to solve linear system – highly spare & large dimension
- MLP (Multilayer perceptron)
  - high-precision fully connected neural network and sigmoid activation function.

Quantitative Finance

- L3: Choose a model, e.g. Binomial Tree, Black Scholes …
- Exercise through L2, L1

Quantitatice Finance	
Function	**Description**
Random Number Generator	Uniform, Normal, Multi Variate
PRNG	pseudorandom number generator
SVD	Singlular Value Decomposition
Tridiagonal Matrix Solver	odd-even elimination
Pentadiagonal Matrix Solver	parallel cyclic reductio - odd-even elimination
SSG Sobol Sequence Generator	quasi-random distribution
Brownian Bridge Transformation	continuoustie stochastic process
Stochastic Process	
Ornstein-Uhlenbeck Process	generate locations for mesher - drift/diffusion
Meshers	finite-difference method
Numerical integration method1Adaptive Trapezoidal, Adaptive Simpson, Romberg	
PCA - rincipal Component Analysis	statistical procedure - orthongonal transformation, correlated variables inot a set of linearly uncorrelated variables
Covariance Matrix and Regularization	
Probablility Distribution	Bernoulli, Binomial, Normal, Lognormal, Poisson, Gamma
Interpolation	Linear, Cubic, Bicubic spline
L1 API: https://xilinx.github.io/Vitis_Libraries/quantitative_finance/2021.1/guide_L1/L1.html#l1-module-	

Vision

- L3: Choose a Vision example, e.g. isppipeline

https://github.com/Xilinx/Vitis_Libraries/tree/master/vision/L3/examples/isppipeline

- Go through L2, L1 test flow

Solver

- matrix decomposition operations, linear solvers and eigenvalue solvers.

Solver	
Matrix decomposition	Cholesky decomposition for symmetric positive definite matrix
	LU decomposition without pivoting and with partial pivoting
	QR decomposition
	SVD decomposition
Linear solver	Tridiagonal linear solver (Parallel cyclic reduction method)
	Linear solver for triangular matrix
	Linear solver for symmetric and on-symetric matrix
	Matrix inverse
Eigenvalue solver	Jacobi eigenvalue solver for symmetric matrix

Security

- Synmmetric Block Cipher – aes, rsa, dsa
- Symmetirc Stream Cipher,
- Asymmetric Cryptography,
- Cipher Modes of Operations,
- Message Authentication Code
- Hash Function

Utilities

- Memory Access
  - AXI Burst Read & Write
- Dynamic Routing within Streams
- Data Reshaping