Multimedia System-on-Chip Design

with specialization on

Application Acceleration with High-Level-Synthesis

賴瑾　Jiin Lai

Founder, CTO, VIA Technologies Inc.

- ***Bachelor's Degree of National Taiwan University majoring in Electrical Engineering 1983***
- ***Master of Science of University of Texas, Austin majoring in Computer Engineering 1987***

Jiin Lai was the Chief Technology Officer for VIA Technologies. He has over 30 years experience in the PC industry, and in the past 12 years in storage area. Early in his career, he is a software engineer developing EDA tools. Later he co-founded VIA technologies, developing PC chipsets, and x86 processor. He led the engineering team to develop Intel and AMD compatible chipsets, and x86-compatible processors. In the past decade, he developed SSD controller, and later, shift focus on developing distributed computational storage system. His responsibility including product and architecture development, with an eye toward to future computing architecture need. Mr. Lai holds over 50 US patents.

Topics

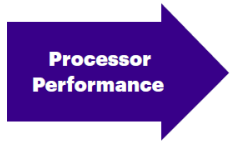
- Objectives of HLS
- HLS Course Logistics

High-Level Statements

- **Computational power hits a plateau**
- **Hardware accelerators to the rescue**

Figure 2: Relative hierarchy of hardware accelerators

(Source: Accenture analysis)



	Central Processing Unit (CPU)	Graphics Processing Unit (GPU)	Field Programmable Gate Array (FGPA)	Application-Specific Integrated Circuit (ASIC)
	Designed for general purpose applications	Designed for graphics-related computations	Array of programmable blocks with a programmable interconnect	Custom designed for specific functionality
Relative Performance	1	100	1000	10,000 – 100,000
Flexibility	General purpose	Special purpose processor	Field (re) programmable	Application specific
Market	Market agnostic	Somewhat restricted market	Somewhat restricted market	Market specific
Ease of Programming	Widely available programming skills	Requires specialized skills	Requires specialized skills	Rigid; interface only
Key Players	Intel, AMD, ARM	NVIDIA, AMD, Intel	Xilinz, Intel (Altera), Actel	NEC, LSI, Samsung

- **The rise of custom accelerator marketplace**

Software defined Hardware – the new era of computing infrastructure
https://www.accenture.com/_acnmedia/pdf-84/accenture-software-defined-hardware-v09.pdf

INTELLIGENT COMPUTING

FPGA Development Made Easy

- HW language are low-level and very difficult
- Know nothing about hardware design
- How software application interacts with FPGA

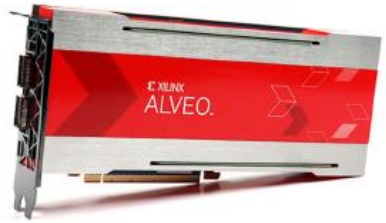
- Use C, C++, OpenCL, Python, or TensorFlow
- Parallel programming Concept apply
- Off-shelf Platform ready

INTELLIGENT COMPUTING

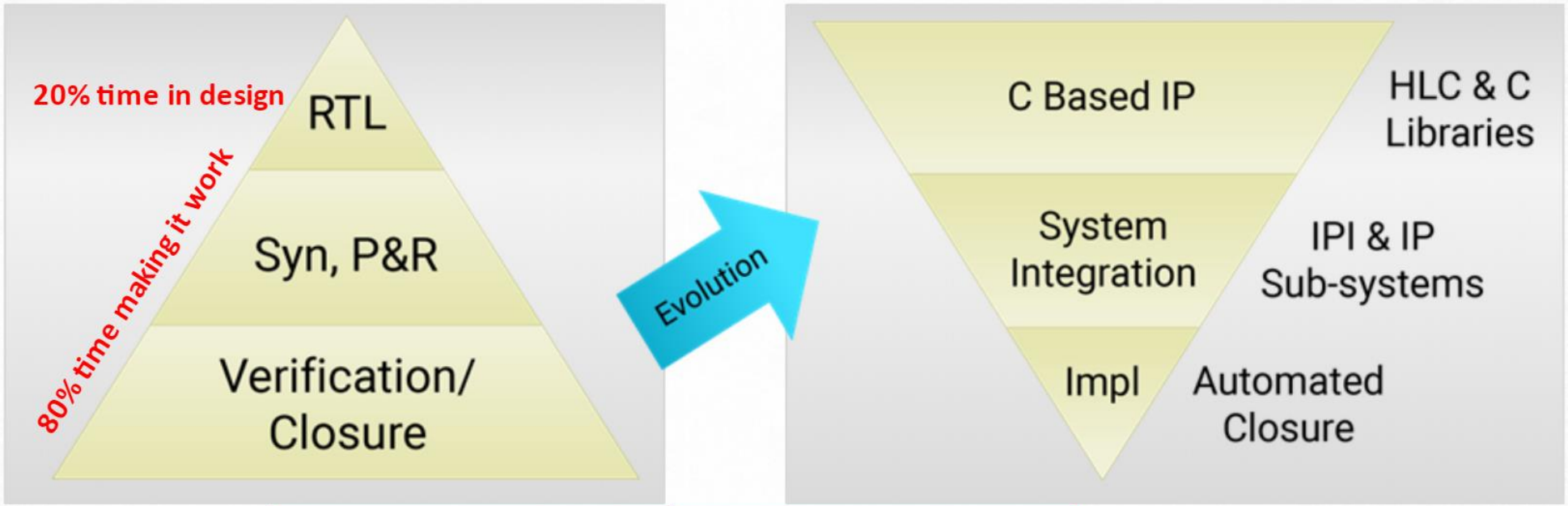
Use C, C++ to program FPGAs

```
// 2D Convolution (11x11)
for (int y = 0; y < height; y++) {
  for (int x = 0; x < width; x++) {
    window = in_stream.read();
    sum = 0;
    for(int row=0; row<11; row++) {
      for(int col=0; col<11; col++) {
        if (index_in_range(x+col-5, y+col-5, width, height)) {
          sum += window.val[row][col]*coeffs[row][col];
        }
      }
    }
    out_stream.write(sum);
  }
}
```

HLS



RTL vs C-Based Design



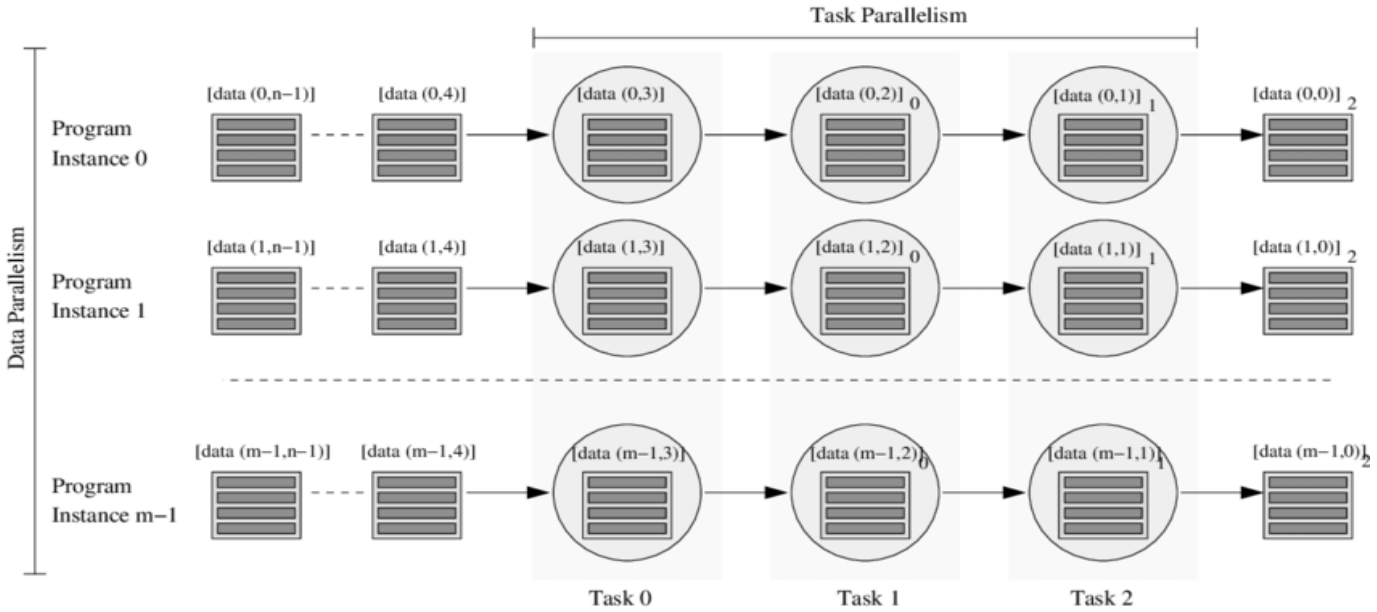
First Design	10X-15X Faster
Derivative Design	40X Faster
Typical QoR	0.7-1.2X

http://www.ecs.umass.edu/ece/labs/vlsicad/ece667/reading/hls-survey.pdf

INTELLIGENT COMPUTING

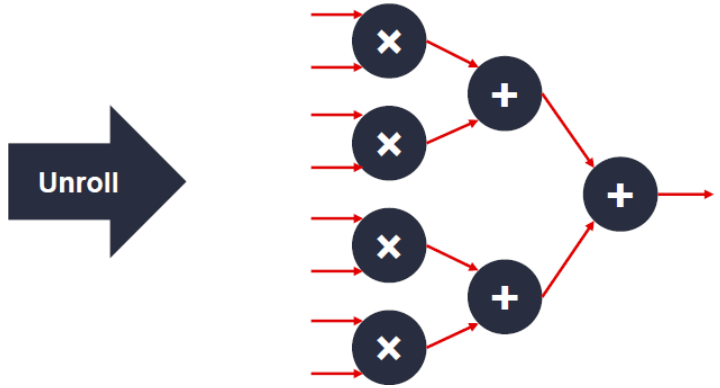
Think “Parallel”

- Data-level Parallelism
- Task-Level Parallelism

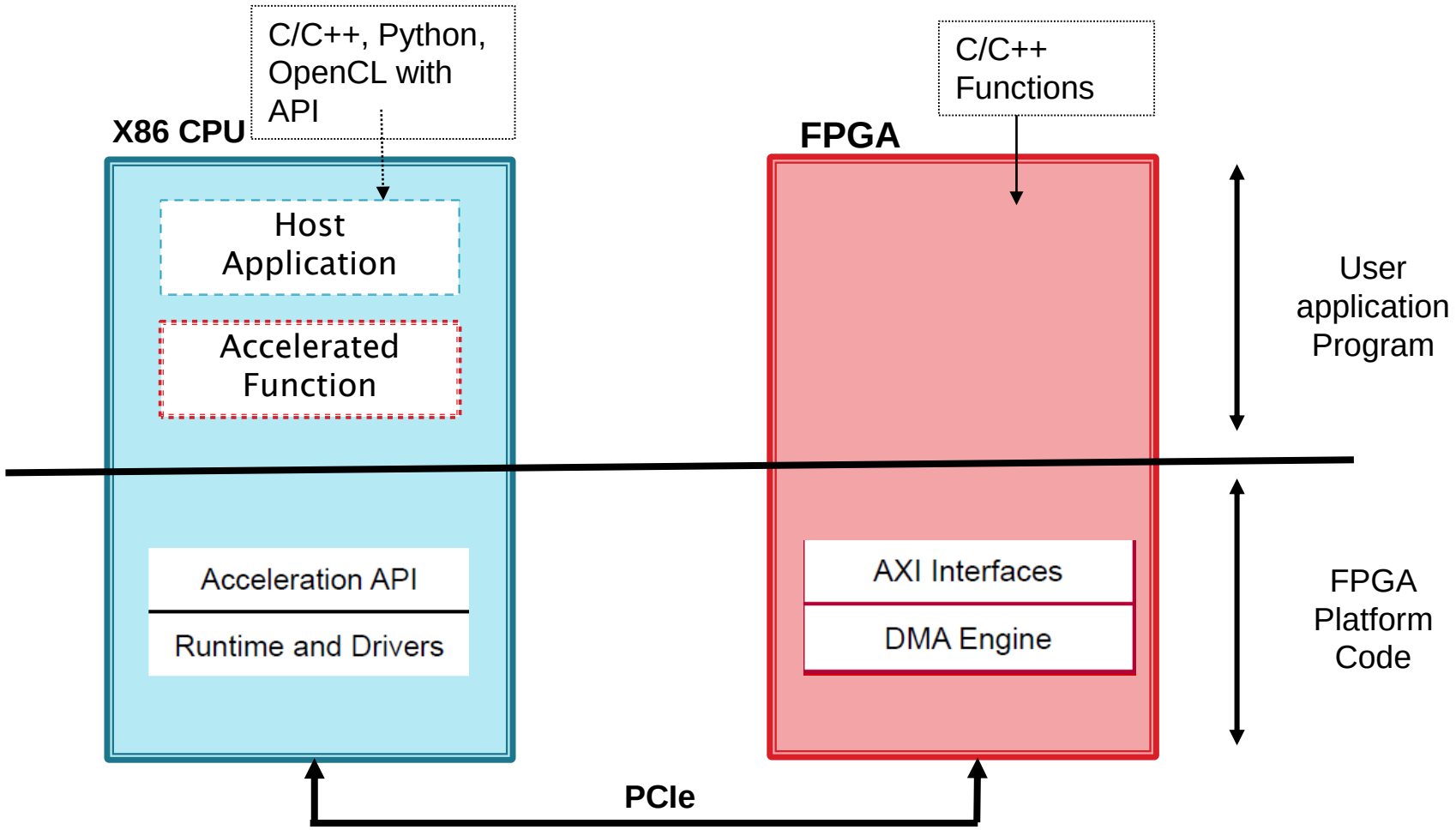


- Instruction (operator) - Level Parallelism

```
for (int i=0; i<N; i++)
{
  acc += A[i] * B[i];
}
```



Software Interacts with FPGA



Speedup Development by Libraries

Use Extensive, Open Source Libraries

GitHub

Domain-Specific Libraries

Vision & Image

Quantitative Finance

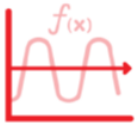
Data Analytics & Database

Data Compression

Data Security

Partner Libraries

Common Libraries



Math

Linear Algebra

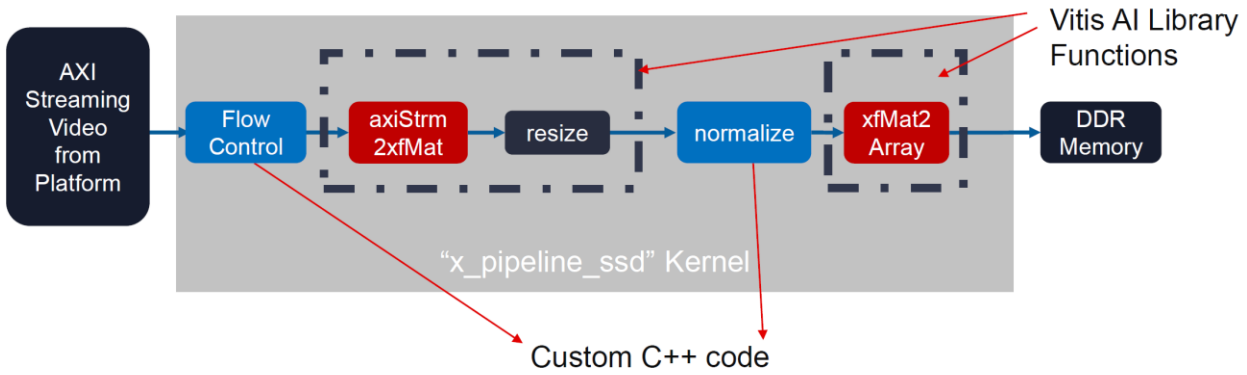
Statistics

DSP

Data Management

400+ functions across multiple libraries for performance-optimized out-of-the-box acceleration

INTELLIGENT COMPUTING



Code Snippet

Top Level Function Definition

```
void x_pipeline_ssd(hls::stream<ap_axiu<24, 3, 1, 1> >& image_in,
                    ap_uint<AXI_WIDTH> *image_out,
                    int width_in,  int height_in, int width_out,
                    int height_out, int use_mean,  float scale_r,
                    float scale_g, float scale_b, unsigned char mean_r,
                    unsigned char mean_g, unsigned char mean_b)
{
```

Interface definitions

```
#pragma HLS INTERFACE axis port = image_in
#pragma HLS INTERFACE m_axi port = image_out offset = slave bundle = image_out_gmem depth = 131072
#pragma HLS INTERFACE s_axilite port = image_out bundle = control
#pragma HLS INTERFACE s_axilite port = width_in bundle = control
#pragma HLS INTERFACE s_axilite port = height_in bundle = control
#pragma HLS INTERFACE s_axilite port = width_out bundle = control
#pragma HLS INTERFACE s_axilite port = height_out bundle = control
#pragma HLS INTERFACE s_axilite port = use_mean bundle = control
#pragma HLS INTERFACE s_axilite port = scale_r bundle = control
#pragma HLS INTERFACE s_axilite port = scale_g bundle = control
#pragma HLS INTERFACE s_axilite port = scale_b bundle = control
#pragma HLS INTERFACE s_axilite port = mean_r bundle = control
#pragma HLS INTERFACE s_axilite port = mean_g bundle = control
#pragma HLS INTERFACE s_axilite port = mean_b bundle = control
#pragma HLS INTERFACE s_axilite port = return bundle = control
```

Dataflow the processing

```
#pragma HLS DATAFLOW
```

Internal streaming variable declarations

```
hls::stream<ap_axiu<24, 0, 0, 0> > px_in_synced;
xf::cv::Mat<XF_8UC3, MAX_IN_HEIGHT, MAX_IN_WIDTH, NPC> in_mat(height_in, width_in);
#pragma HLS STREAM variable=in_mat.data depth=256 dim=1
xf::cv::Mat<XF_8UC3, MAX_IN_HEIGHT, MAX_IN_WIDTH, NPC> in_rgb(height_in, width_in);
#pragma HLS STREAM variable=in_rgb.data depth=256 dim=1
xf::cv::Mat<XF_8UC3, MAX_OUT_HEIGHT, MAX_OUT_WIDTH, NPC> out_rgb(height_out, width_out);
#pragma HLS STREAM variable=out_rgb.data depth=256 dim=1
xf::cv::Mat<XF_8UC3, MAX_OUT_HEIGHT, MAX_OUT_WIDTH, NPC> out_mat(height_out, width_out);
#pragma HLS STREAM variable=out_mat.data depth=256 dim=1
```

Synchronize to start of frame

```
flow_control(image_in, px_in_synced, height_in, width_in);
```

Convert from axi stream to xf::Mat

```
xf::cv::axiStrm2xfMat<24, XF_8UC3, MAX_IN_HEIGHT, MAX_IN_WIDTH, NPC>(px_in_synced,in_mat);
```

Resize the image

```
xf::cv::resize<XF_INTERPOLATION_NN,
               XF_8UC3,
               MAX_IN_HEIGHT,
               MAX_IN_WIDTH,
               MAX_OUT_HEIGHT,
               MAX_OUT_WIDTH,
               NPC,
               MAX_DOWN_SCALE>(in_mat, out_rgb);
```

Subtract the mean values and apply scale

```
image_normalize(out_rgb, out_mat, use_mean, scale_r, scale_g, scale_b, mean_r, mean_g, mean_b);
```

Convert from xf::Mat to memory mapped interface

```
xf::cv::xfMat2Array<AXI_WIDTH, XF_8UC3, MAX_OUT_HEIGHT, MAX_OUT_WIDTH, NPC>(out_mat, image_out);
}
```

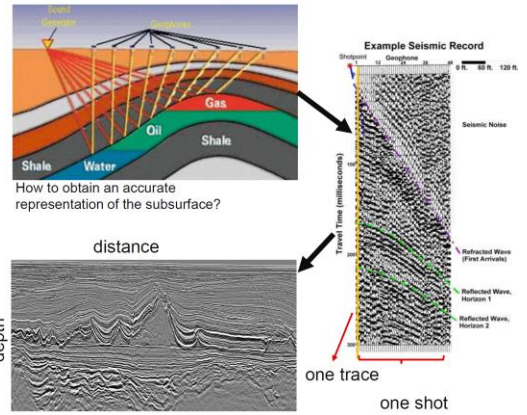
Example of Oil, Gas workload

Productivity

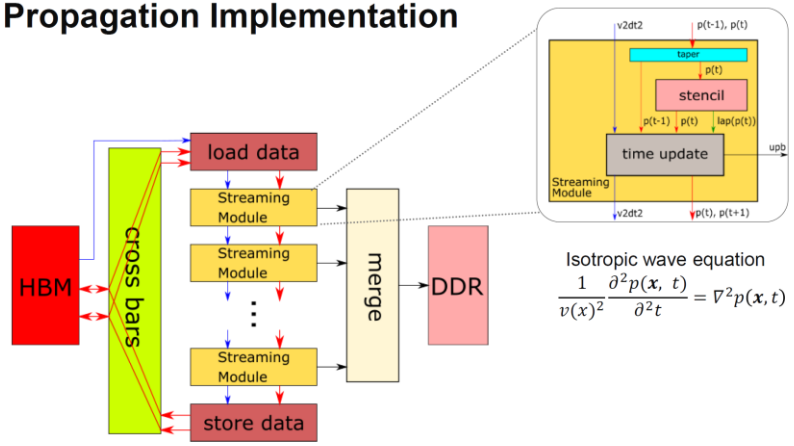
- **Not the traditional programming model for FPGAs:**
- One Software Engineer, no previous O&G experience, one month to describe & implement entire RTM Algo in C++
  - No optimized library calls, completely described in C++
  - < 500 lines of code, < 50 Pragmas
- Standard language, open source tools and libraries

Seismic Method for Oil and Gas industry

- Seismic Imaging Technology
  - Seismic Survey: Acoustic wave sampling
  - Seismic Imaging: Mathematically process the wave traces to create an image
- RTM (Reverse Time Migration)
  - High-fidelity algorithm for imaging complex sub-surface structures
  - Cross-correlation between source wavefield and receiver wavefield
  - Wavefield reconstruction by saved boundaries



Forward Propagation Implementation



Isotropic wave equation

$$\frac{1}{v(x)^2}\frac{\partial^2 p(\boldsymbol{x},\ t)}{\partial^2 t} = \nabla^2 p(\boldsymbol{x}, t)$$

INTELLIGENT COMPUTING

Message Intelligence Appliance Cortical.io

- Semantic Supercomputing for NLU (Natural Language Understanding)
- Automatically classifies message based on semantics/meaning of the content

Hardware Pre-configured
> Xilinx Alveo U50 Accelerator Card
> Supermicro AMD-based Server

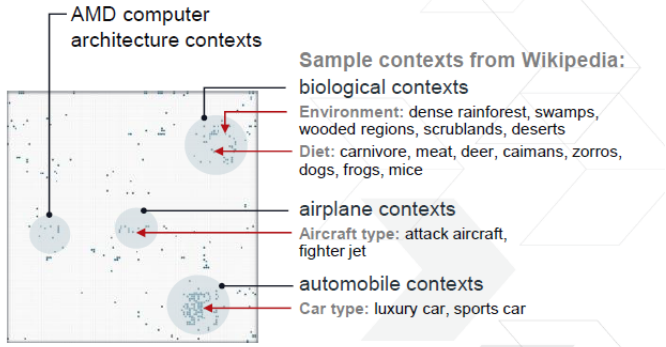
Software Pre-installed
> Cortical.io Message Intelligence

Semantic Folding Explained

Words, sentences & paragraphs are represented by a semantic fingerprints

> Each word is represented by **16K binary** contexts in a 2D vector
> All operations are **binary**
> **Minimal** source material required: reference material, textbooks, data sheets, emails, etc.
> Creation of the semantic fingerprints is **completely unsupervised**
> **All meanings** of a word are represented

Semantic Fingerprint

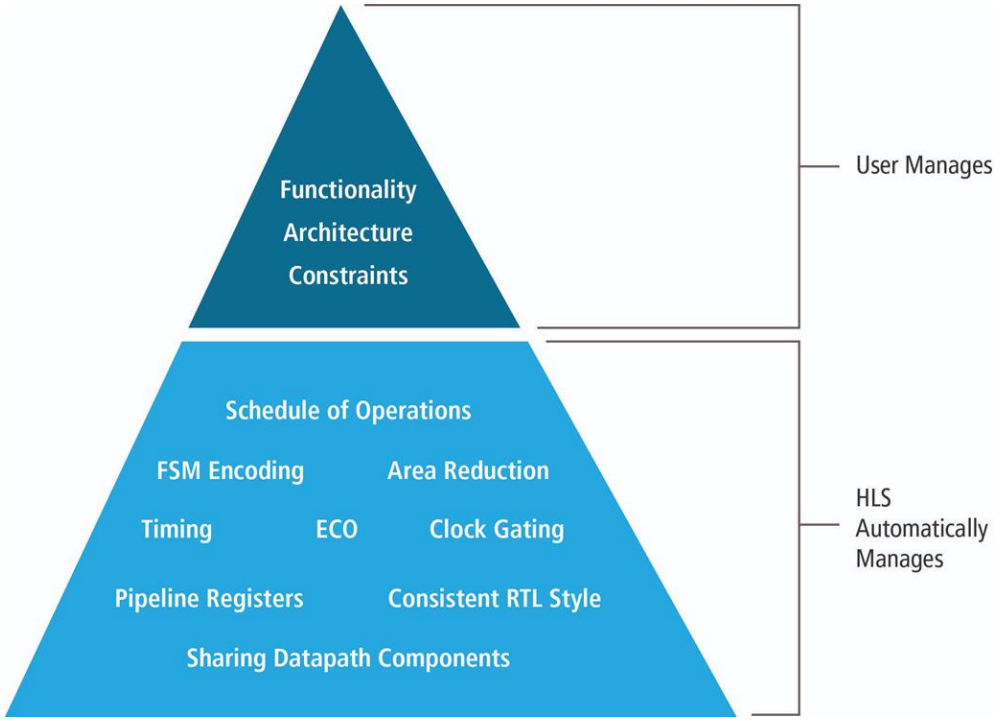


Semantic Fingerprint of the word "Jaguar"

Semantic folding theory white paper: https://www.cortical.io/static/downloads/semantic-folding-theory-white-paper.pdf

Why HLS?

- Productivity (Design and Verification)
- IP Reuse
- Better QoR
- End-to-end application acceleration by software designer
- For academic, A great tool/skill for research.



Course Objectives

Empower software designers to develop efficient application accelerator

Course Contents

- Course Texts:
  - R. Kastner, Parallel Programming for FPGAs, arXiv, 2018
  - Xilinx ug902
- Supplementary Materials:
  - Reference Papers
  - Manual/Datasheets
- Lecture ppt & video – 16 sessions
- Labs – ~200 optional lab references
- In-class presentation – 5 sessions + final project presentation
- Final project & presentation

Logistics

- Off-class lecture & lab/assignment
  - Lecture is self-paced
  - Lab/assignment is self-paced with lab-work submission
- In-class presentation & discussion
  - Sign-up by Google-Form (submit by Thursday 3pm)
  - Presentation selected based on available time slots, weight, and submission time.

  - Refer to “HLS Course Plan.doc”

In-class schedule and subjects

https://cool.ntu.edu.tw/courses/3773/modules/items/110288

session	Date	Suggest lecture title - self-paced	pdf	video	In-classs discussion topics	Assignment
1	18-Sep	Course Introduction	§	§		HLS flow
		Introduction PYNQ & Lab2	§	§		ug871 labs **ug871-[1:7]**
		Vitis OpenCL XRT and Lab3	§	§		Lab1: Tool Installation
		Introduction to FPGA				Lab2: PYNQ axi-m & stream
						Lab3: OpenCL/XRT
2	16-Oct	Kernal IO Interface	§	§	ug871 Labs ug871-[1:7]	Xlinx Training Lab -**xtrain-[1:13]**
		Introduction to High Level Synthesis	§	§	HLS, Vivado, Vitis usage experience	Xilinx HLS Coding Style
		FPGA - CLB	§	§	Lab1, Lab2, Lab3 sharing	Xilinx HLS Design - **xdesign-[1:15]**
		FPGA - Memory	§	§		
			§	§		
3	30-Oct	System Optimization - Host	§	§	Xlinx Training Lab -xtrain-[1:13]	Vitis Tutorial - **vitis-[1:7]**
		System Optimizatin - Kernel	§	§	Xilinx HLS Design - xdesign-[1:15]	UCSD Lab **ucsd-[1:5]**
		FPGA - DSP	§	§		Cornell - ECE5775 **cornell-[1:4]**
		FPGA - Interconnect	§	§		
4	13-Nov	Kernel Optimization - Area	§	§	Vitis Tutorial - vitis-[1:7]	**pp4fpga-[1:8]**
		Kernel Optimization - Latency	§	§	UCSD Lab ucsd-[1:5]	Xilinx HLx Examples **xhls-[1:18]**
		Kernel Optimization - Pipeline	§	§	Cornell - ECE5775 cornell-[1:4]	
5	27-Nov	Design Examples	§	§	pp4fpga-[1:8]	
		Application Cases	§	§	Xilinx HLx Examples xhls-[1:18]	Xilinx Application Notes **xapp-[1:13]**
6	11-Dec				All topics	Final Project
						Refer to project resource weight=10
	22-Jan	Final Project presentation				

INTELLIGENT COMPUTING

Lectures – Self-Paced

1. **Tools & Platform**
   a. Introduction to PYNQ & Lab2
   b. Vitis OpenCL XRT and Lab3
2. **FPGA (Xilinx)**
   a. Introduction to FPGA
   b. FPGA – CLB
   c. FPGA – Memory
   d. FPGA – DSP
   e. FPGA – Interconnect
3. **Concept of System Performance and Optimization**
   a. Host Optimization
   b. Kernel Optimization

1. **HLS Development**
   a. Introduction to High Level Synthesis
   b. Kernel IO Interface
   c. Kernel Optimization – Area
   d. Kernel Optimization – Latency
   e. Kernel Optimization – Pipeline
2. **Design Examples and Application**
   a. Design Examples
   b. Application Cases

INTELLIGENT COMPUTING

Platform/Tools for Labs

- Vivado HLS - For Kernel optimization
  - Vivado HLS – C-sim, Synthesis, Co-Sim, IP-generation
  - Analyze resource, latency, timeline/scheduling, waveform
- Pynq (MPSOC - AXI) – Embedded System
  - Run HLS – C-sim, Co-sim, IP-generation
  - Vivado – IP integration, block-design, generate bit-stream
  - Download to Zedboard/PYNQ-Z2 and run Jupyter Notebook
- Vitis & AWS-F1 (FPGA-PCIe) – Cloud Application
  - Run HLS – C-sim, Co-sim, IP-generation
  - Vitis - run SW-emulation, HW-emulation, Bitstream generation
  - Upload to AWS, run application (host code) at host PC
  - Profiling and analyzing application performance

Xilinx Tools & Exercise

- Exercises/tutorials provided to gain proficiency in design flow
  - Vivado HLS 2019.2
  - Vivado Design Suite 2019.2
  - Xilinx Vitis IDE/Makefile
  - AWS

Refer to “Xilinx Tool Flow.ppt”

Develop Basic Skill/Tools in the first two weeks

- The following three labs in the first two weeks
  - Lab#1 - Tool installation and Implementation Flow.
  - Lab#2 - Application Acceleration for Embedded System (PYNQ-Zedboard).
  - Lab#3 - Application Acceleration for Cloud Environment (Amazon)

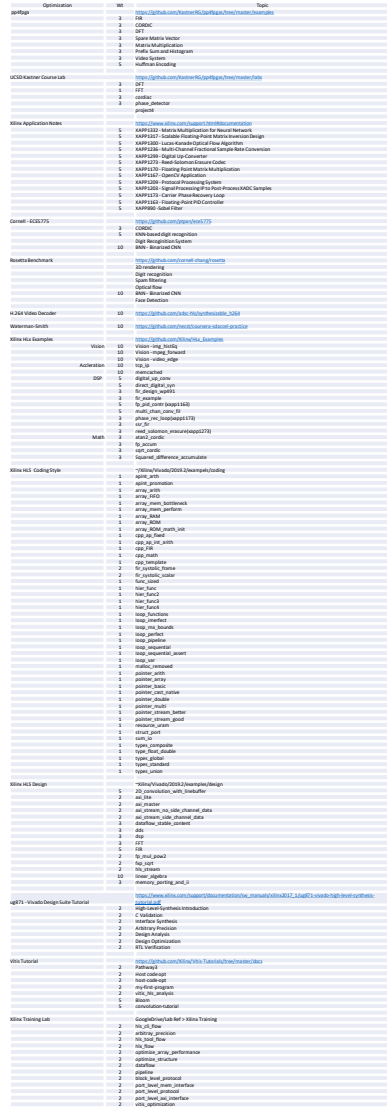
Lab/Assignment & Submission Criteria

- Lab/Project reference resources
  - Refer to “HLS Lab Project Resources.xls” https://cool.ntu.edu.tw/courses/3773/modules/items/110289
- Weight Categories

Weight	Description	Submission
1,2	Single item, exercise optimization pragma, coding style, setup/runing/analysis effort: 30min-1hr	1. Screen dump: HLS, latency, resource, io interface, timeline 2. Vitis summary, HLS synthesis_report 3. ppt/word: description of observation and learning
3,4	code hoist, Exercise mulitple optimization, comparative analysis, effort: 2-3 hr	
5-9	algorithm level: code hoist, comparative analysis, effort: days	1. **ppt & presentation** - introduce domain knowledge/theorem - optimization method - comparison of optimization merit / tradeoff 2. **github submission for publication**
10	application level, need domain knowledge/background, effort: weeks Candidate for final team project	

- Other Lab/Project proposal is welcomed. Weight will be assigned.

Lab/Project References



Lab/Project Resources

LabName	ID	Wt	Topic	
pp4fpga			https://github.com/KastnerRG/pp4fpgas/tree/master/examples	Kastnter pp4fpga text book
	1	3	FIR	
	2	3	CORDIC	
	3	3	DFT	
	4	3	Spare Matrix Vector	
	5	3	Matrix Multiplication	
	6	3	Prefix Sum and Histogram	
	7	3	Video System	
	8	5	Huffman Encoding	
185	**h264**	10	https://github.com/adsc-hls/synthesizable_h264	H.264 Video Decoder
186	**swater**	10	https://github.com/necst/coursera-sdaccel-practice	Smith-Waterman - gene sequence
187	**cirrna**	10	https://github.com/necst/circFAXOHW18public	circular RNA aligner
188	**point5**	10	https://bitbucket.org/necst/xohw18_5points_public/src/master/	five point relative pose problem
189	**sha256**	5	https://github.com/dowenberghmark/FPGA-SHA256	SHA256
190	**beamf**	5	https://developer.xilinx.com/en/articles/beamforming-acceleration.html	beamforming
191	**ethash**	10	https://developer.xilinx.com/en/articles/part1-introduction-to-ethash.html	blockchain - hashing for Ethereum
192	**mcarlo**	10	https://github.com/KitAway/FinancialModels_AmazonF1/tree/master	Monte Carlo financial models
193	**profax**	10	https://bitbucket.org/necst/profax-src/src/master/	Protein Folding Algorithm
194	**graph**	10	https://github.com/Xtra-Computing/ThunderGP	Graph Processing

Course Credits

- Earn credits from the followings:
  - Submit Lab/Assignment choose from lab/project references or propose yours
    - Credit based on the weight category
  - Class Presentation – 5 minutes presentation
    - weight category * quality <**0.7 - 1.3**>（**insight** + presentation skill)
  - Final project – 10 minutes presentation
    - Weight <5-10> * quality <**0.7 - 1.3**> (**insight** + presentation skill)
- Where is the insight from
  - Fully understand the material
  - Deeper observation on the analysis report
  - Try out different optimization, make trade-off, and comparative analysis