Imperial College London

Reconfigurable Computing and its Applications on Image Processing

Christos Bouganis
ccb98@ic.ac.uk

Imperial College London

Outline

- Reconfigurable Computing
  - Need for reconfigurable computing
  - Current devices
  - Programming models

- FPGA Optimizations: Word-length optimization

- Dimensionality Reduction Framework targeting FPGAs

Imperial College London

Reconfigurable computing

Spectrum of computational devices

ASIC — FPGAs — GPUs — DSPs — General purpose processors

- Low power/ computation
- Small area/ computation
- High Performance

- High power/ computation
- High area/computation

- Long development process
- Not flexible

- Short development process
- Flexible

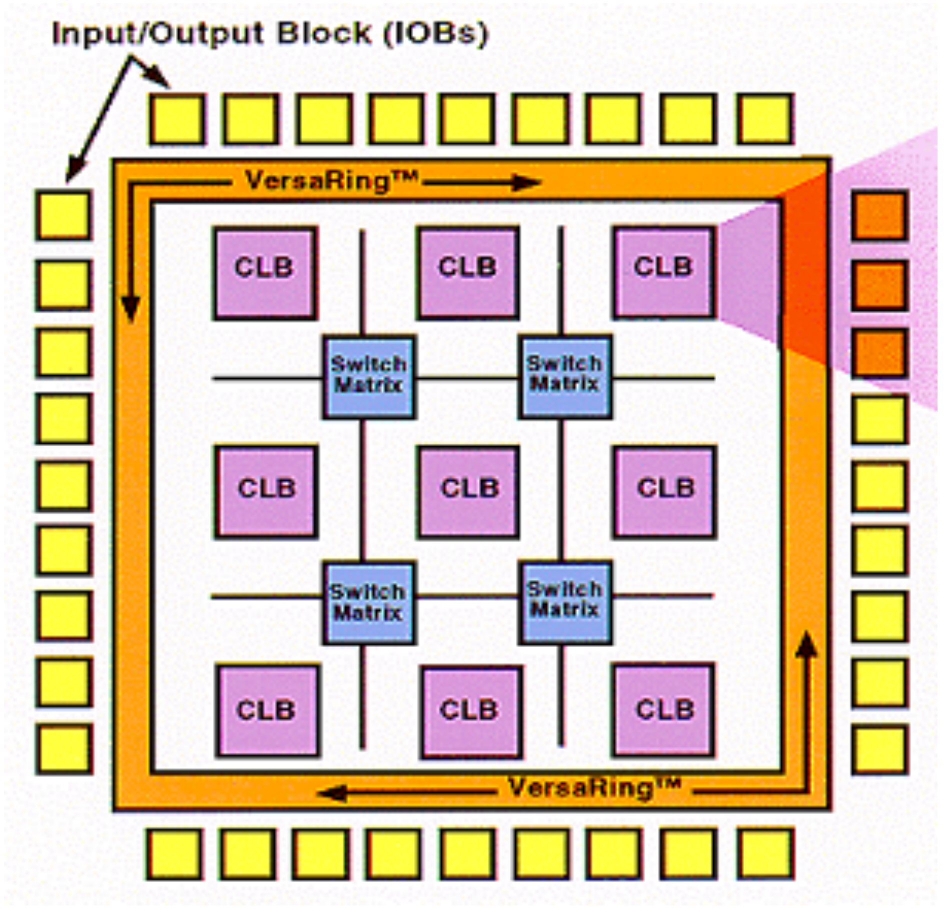
Benefits come by customizing your hardware to the application

Imperial College
London

Field Programmable Gate Array - FPGA

- Xilinx – first to introduce SRAM based FPGA using Lookup Tables (LUTs)

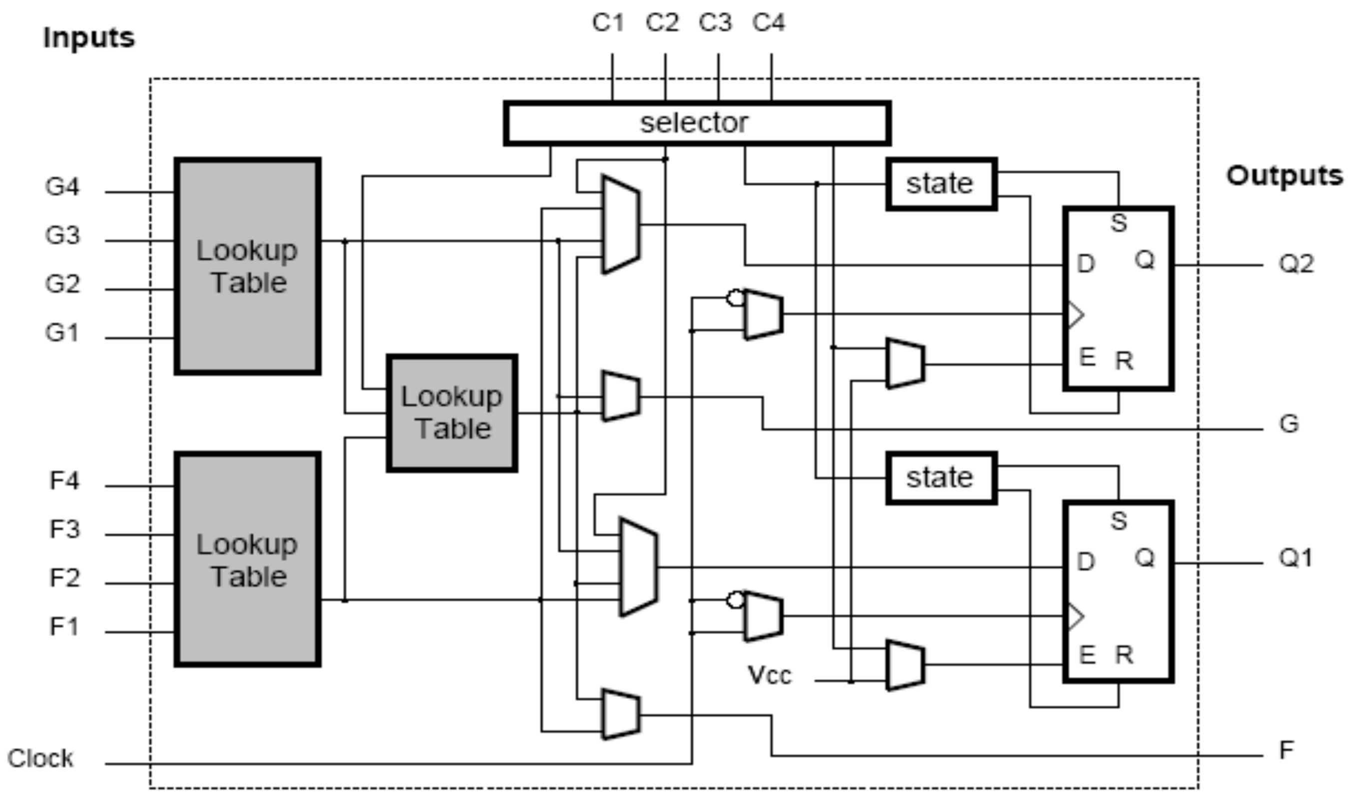
- Xilinx 4000 series contains four main building blocks:
  - Configurable Logic Block (CLB)
  - Switch Matrix
  - VersaRing
  - Input/Output Block



Imperial College London

Structure of FPGAs - CLB

- Each Configurable Logic Block (CLB) has 2 main Look-up Tables (LUTs) and 2 registers.
- The two LUTs implement two independent logic functions F and G.

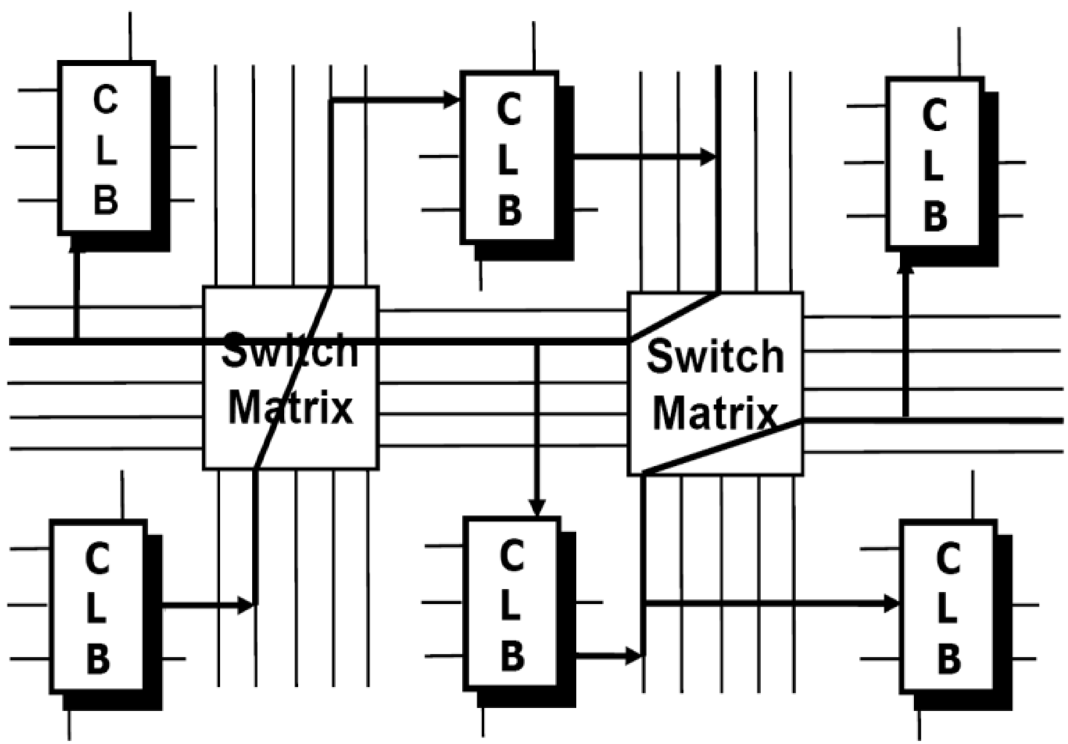


Imperial College London

Structure of FPGAs - Programmable Interconnect

Switch-box provides programmable interconnect

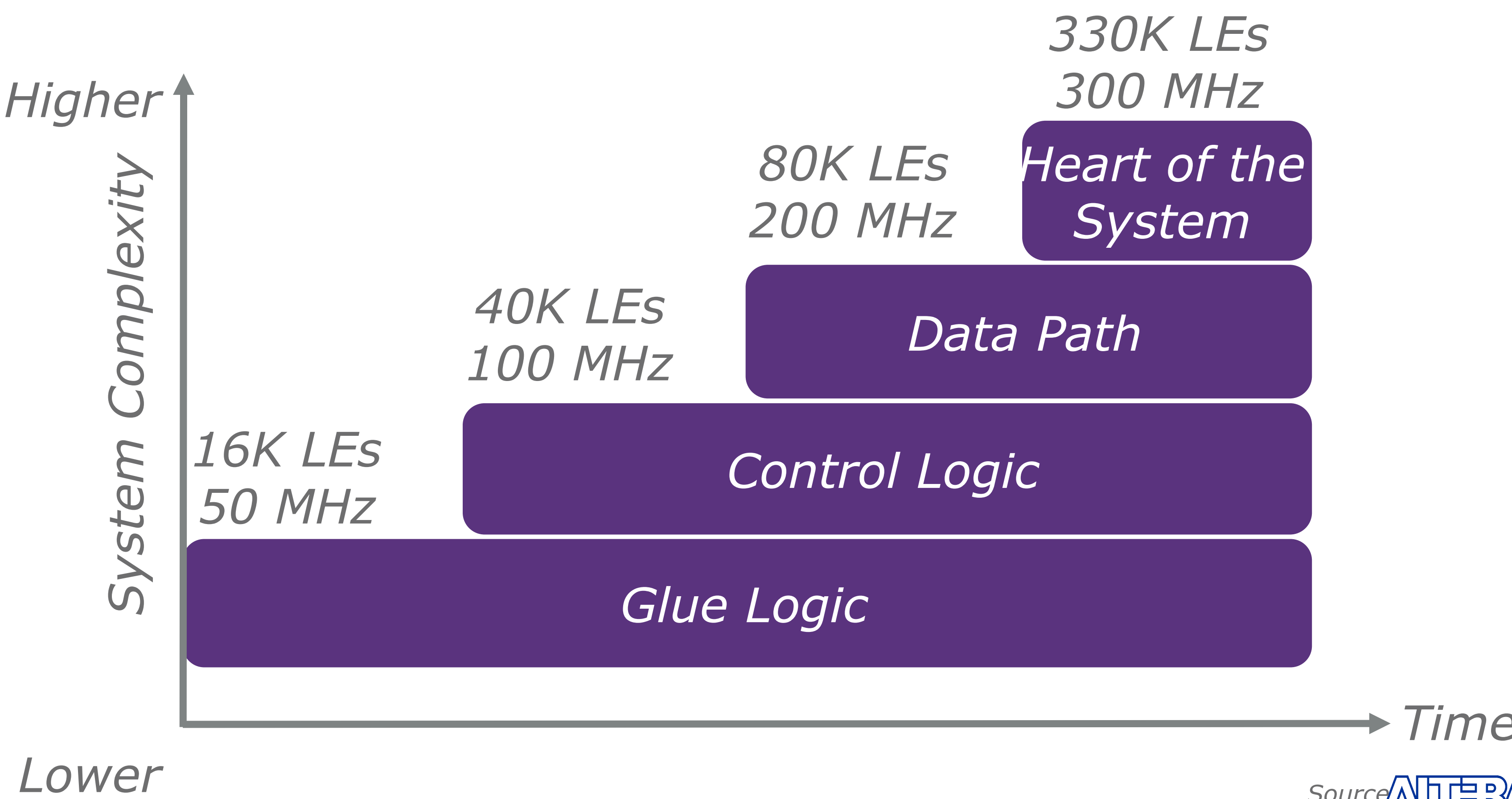
- Local interconnects are fast and short
- Horizontal and vertical interconnects are of various lengths



Imperial College London

FPGA's journey: from Glue Logic to SoC

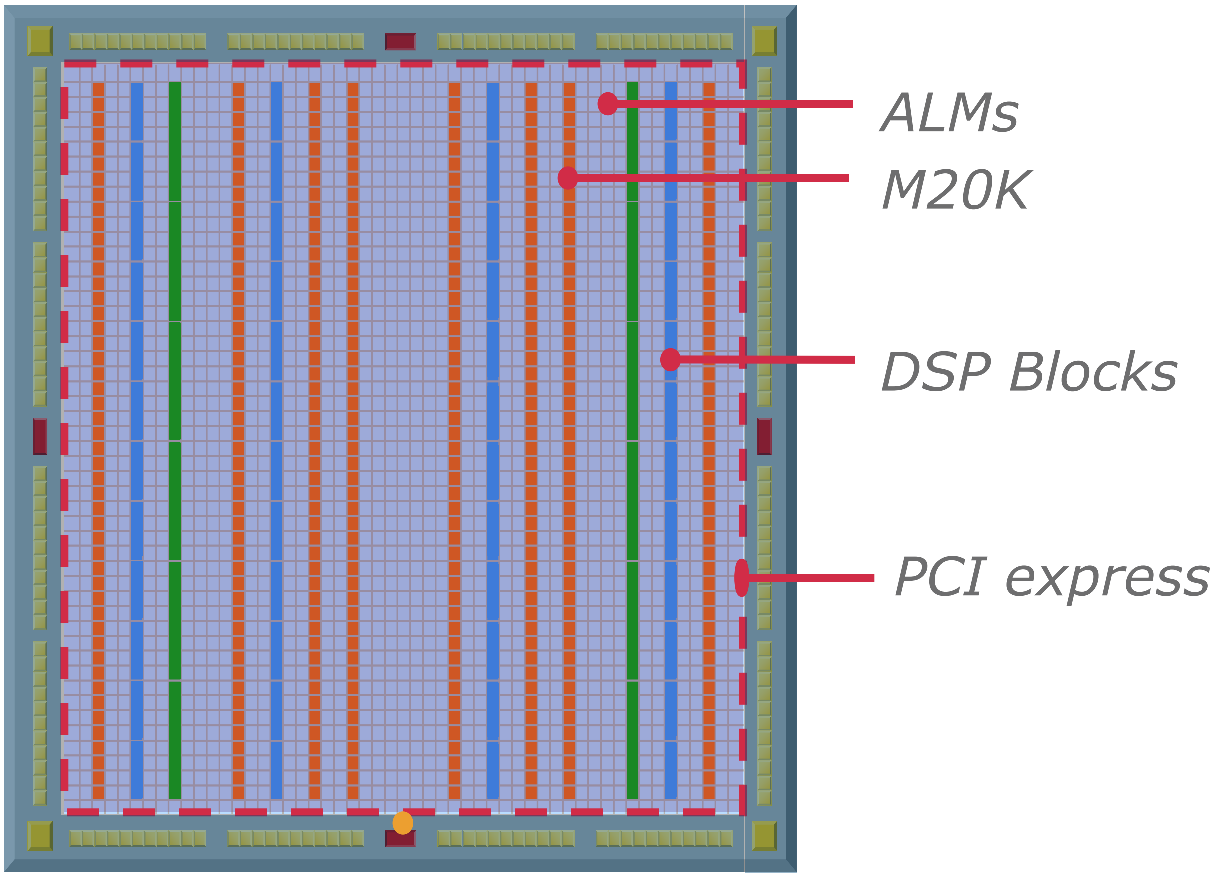
FPGAs are now being used everywhere!



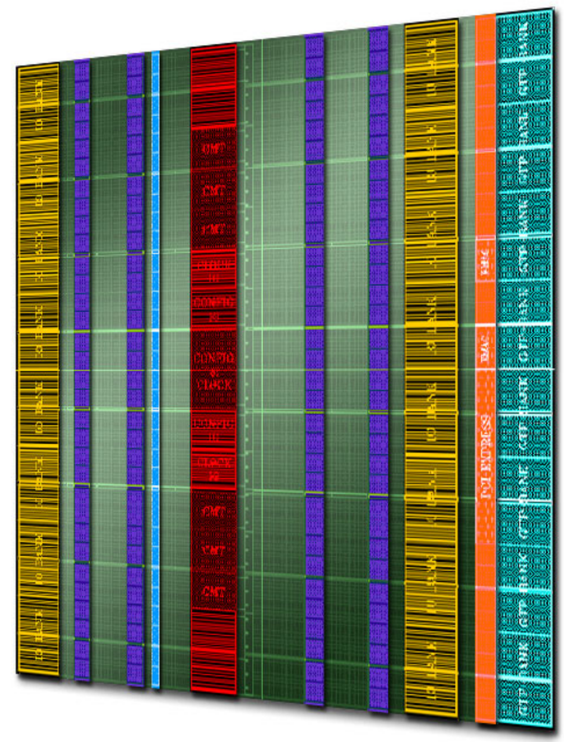
Source: ALTERA

Imperial College
London

Modern FPGA devices – More heterogeneous



Stratix V



Virtex-7

Programming tools

- VHDL
  - Low level programming language
  - Best performance
- Domain specific languages
  - System Generator for DSP (Xilinx)
  - DSP builder (Altera)
  - Simulink (Mathworks)
- High-level languages
  - C to RTL (Handel-C, CatapultC, …)
  - Matlab to RTL
- Specific tools to speed-up development
  - FloPoCo

Low level
Hard to program
High performance

Tool needs to
bridge the gap

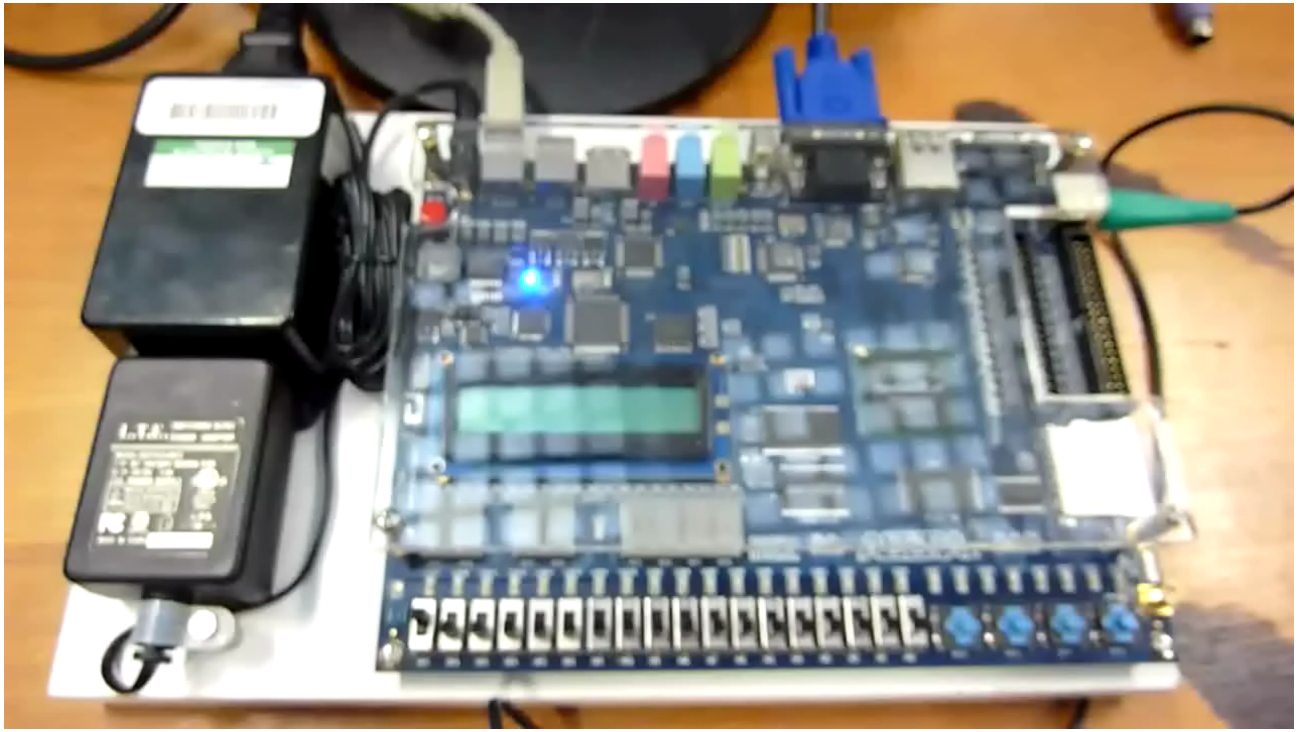
High level
Easy to program
Low performance

Imperial College London

High level programming potential

An example from a first year group project

- 4 weeks learning FPGA + Handel-C
- 4 weeks of work

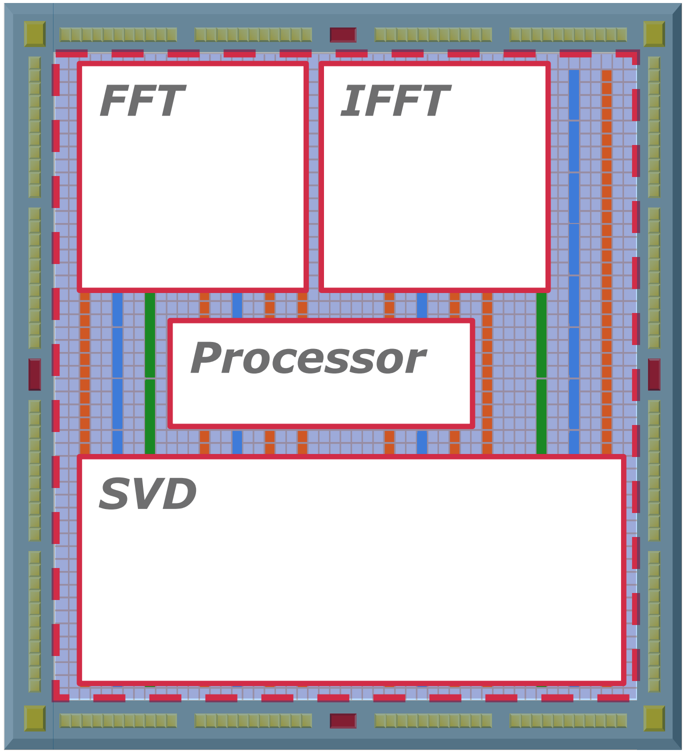


A system on an FPGA

Mapping of an application to an FPGA

- Modern FPGAs allow SoC
- Performance improvement => Parallelism

Resources (area) ⟷ Parallelism



Wordlength Optimization

- FPGAs operate on any number representation
  - Floating point (Double precision)
  - Floating point (Single precision)
  - Custom floating point
  - Fixed point number representation

High Precision
Large area

Low Precision
Small area

- *Example:*
  - *Mapping of an FIR filter to an FPGA*
  - *Input: pixels (8-bits)*

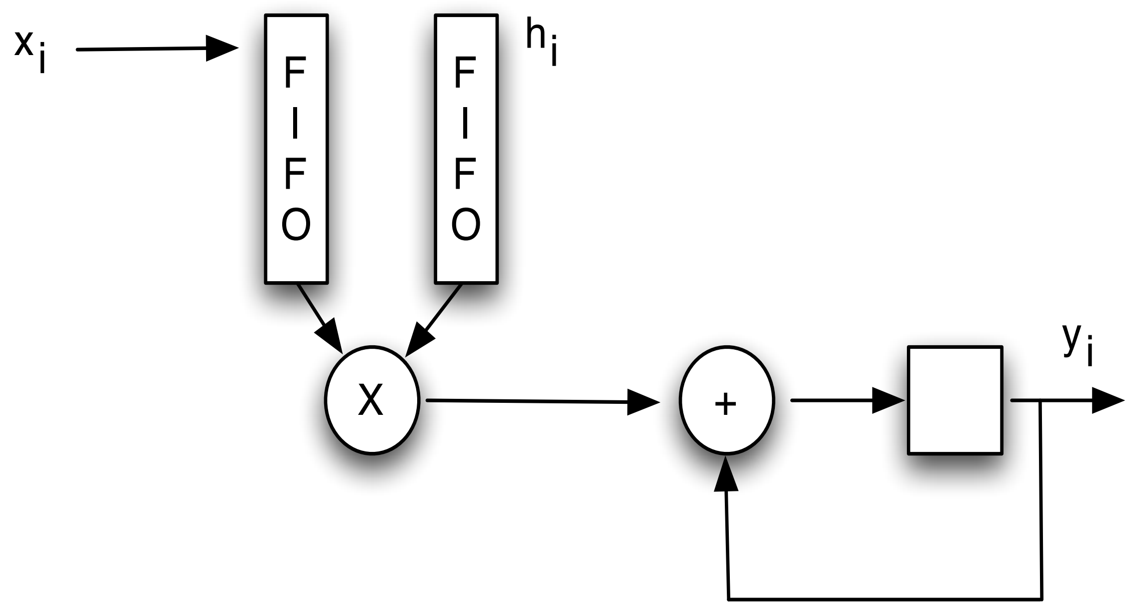
$$y(n) = \sum_{i=0}^{N-1} h(i) * x(n-i)$$

Architecture 1: Sharing resources

The architecture utilizes 1 multiplier + 1 adder

Maps very well in modern FPGAs

(MAC units + embedded RAMs)

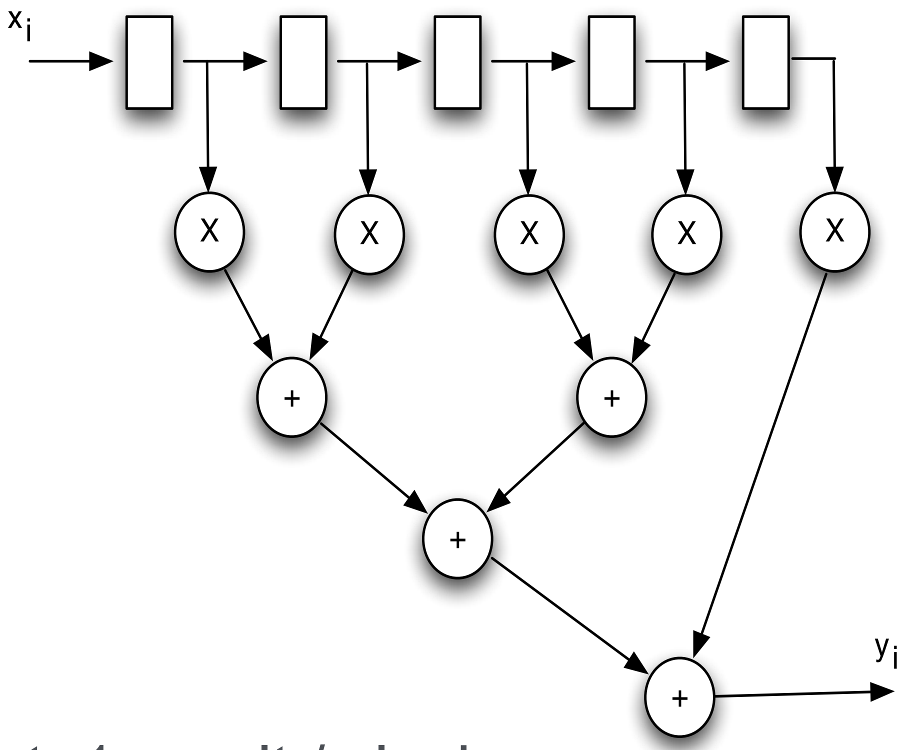


Throughput: 1 result / N clocks

Imperial College London

Architecture: Fully unrolled

The architecture utilizes N multipliers + N-1 adders (direct-form)



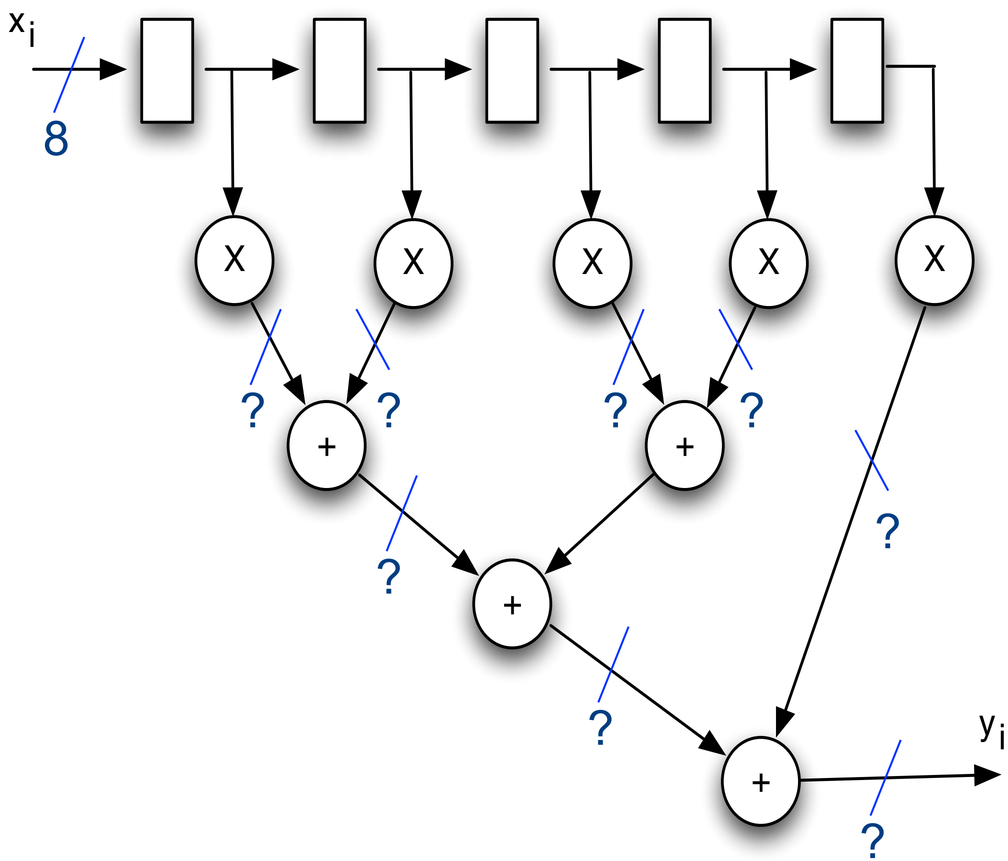
Constant coeff multipliers

Throughput: 1 result / clock

Imperial College London

Wordlength Optimization Problem

Select the wordlengths of the various signals optimizing an objective function



Possibilities

- Uniform selection
  - Similar to CPU/DSP
- Select wordlength for each signal
  - Interval arithmetic
  - Allow errors
    - Monte Carlo
    - Analytical methods (LTI)

Example

Dimensionality Reduction Framework targeting FPGAs

Motivation

Many applications require the representation of data using a set of fewer variables allowing a certain error in the approximation

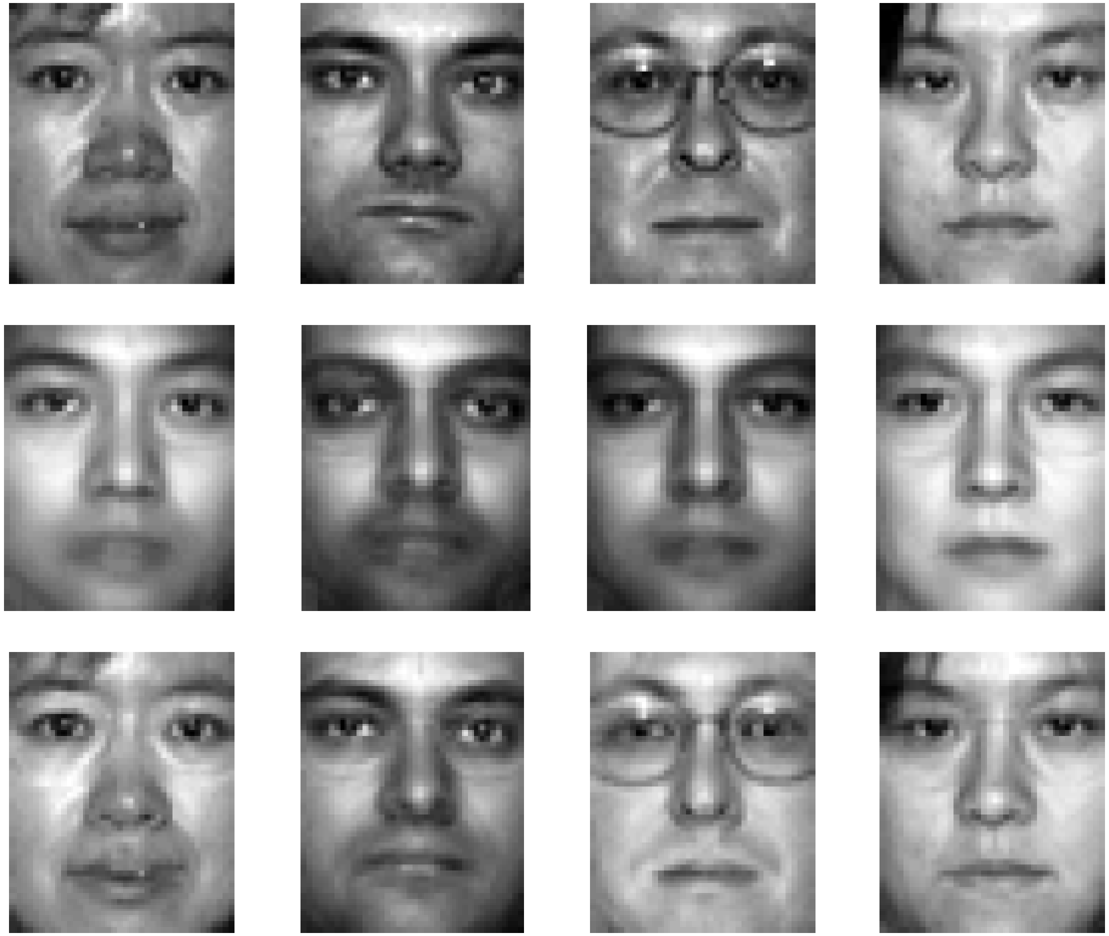
Dimensionality Reduction or *Feature Extraction* problem

Examples:

- Face detection/recognition
- Image compression
- …

Map a dimensionality reduction system in a modern FPGA in an efficient way (resource usage)

Principal Component Analysis
Face recognition example



Original space (2000 dimensions)

*Reduced space * (3 dimensions)*

*Reduced space * (40 dimensions)*

** Using the PCA algorithm*

Background - Linear Projection

Linear projection

$$x = \Lambda f + \varepsilon$$

Original data → x

Projection basis
Factor Loadings matrix → Λ

Factors (Reduced space) → f

Error → ε

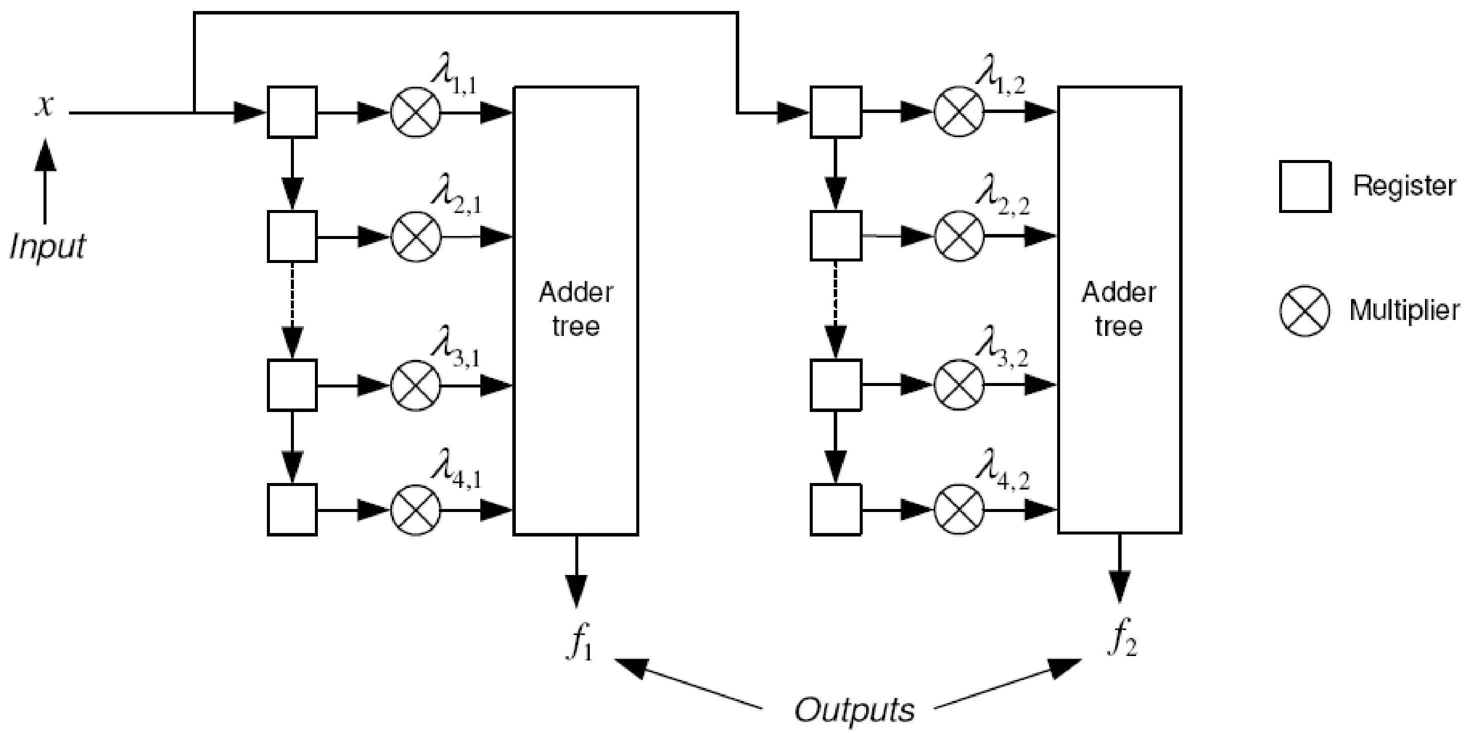
When projection vectors are orthogonal:

$$f = \Lambda' x$$

Imperial College London

Hardware implementation – Fully unrolled

Mapping: $Z^4 \rightarrow Z^2$

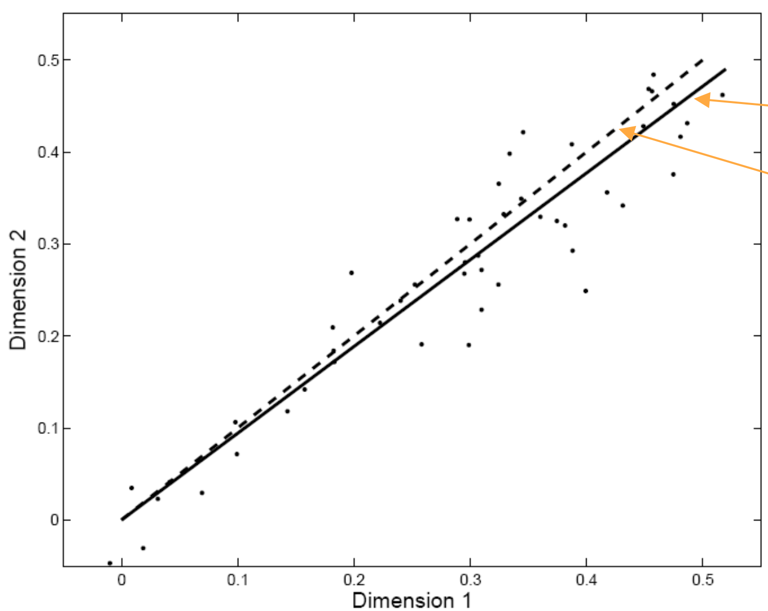


Related work

Current methodology

- Estimate the new space using PCA (in floating point)
- Quantize the coefficients to fixed point for hardware implementation

- *Drawback: Does not take into account the available hardware*
- *Illustration:*



$$\Lambda = [0.52 \quad 0.49]$$

$$\Lambda_{opt} = [0.50 \quad 0.50]$$

Imperial College London

Proposed algorithm

Main idea

- Couple the problem of new space basis calculation and hardware implementation optimization

Bayesian Factor Analysis Model

$$x = \Lambda f + \varepsilon$$

$$\sim \mathcal{N}(0, \Psi)$$

Bayesian Factor Analysis Model

Probability distribution of data

$$
\begin{aligned}
p(x^i|f^i,\Lambda,\Psi) &= \mathcal{N}(x^i|\Lambda f_i,\Psi) \\
&= (2\pi)^{-P/2}|\Psi|^{-1/2} \times \\
&\quad \exp\left(-\frac{1}{2}(x^i-\Lambda f^i)'\Psi^{-1}(x^i-\Lambda f^i)\right)
\end{aligned}
$$

Factors

Prior distr.

$$f^i \sim \mathcal{N}(0,\Sigma_F)$$

Posterior distr.

$$p(f^i|x^i,\Lambda,\Psi) \propto p(f^i)p(x^i|f^i,\Lambda,\Psi) = \mathcal{N}(f^i|m_F^*,\Sigma_F^*)$$

$$
\begin{aligned}
\Sigma_F^* &= (\Sigma_F + \Lambda'\Psi^{-1}\Lambda)^{-1} \\
m_F^* &= \Sigma_F^*\Lambda'\Psi^{-1}x^i
\end{aligned}
$$

Bayesian Factor Analysis Model (cont')

Factor Loadings matrix Λ

Prior distr. $$p(\Lambda) = \prod_{p=1}^{P} \prod_{k=1}^{K} p(\lambda_{pk})$$ (assume independence)

Posterior distr. $$p(\Lambda \mid X, F, \Psi) \propto p(X \mid F, \Lambda, \Psi)\, p(\Lambda)$$

$$= p(X|F, \Lambda, \Psi) \prod_{p=1}^{P} \prod_{k=1}^{K} p(\lambda_{pk})$$

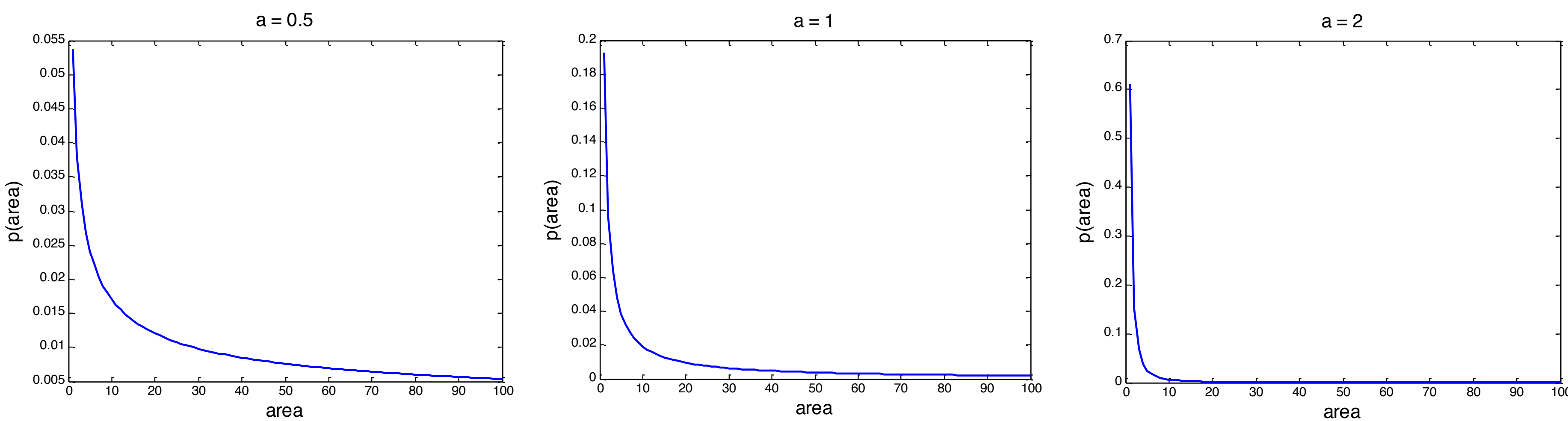
Insert any prior knowledge about the cost of the implementation

Mapping implementation cost to prior distribution of basis

Seek function that maps area cost to a probability distribution

- monotonically decreasing
- no negative
- sum to one

Current work: $$g(A(\lambda_{pk})) = c\left(A(\lambda_{pk})\right)^{-a} \quad , \quad a, c > 0$$



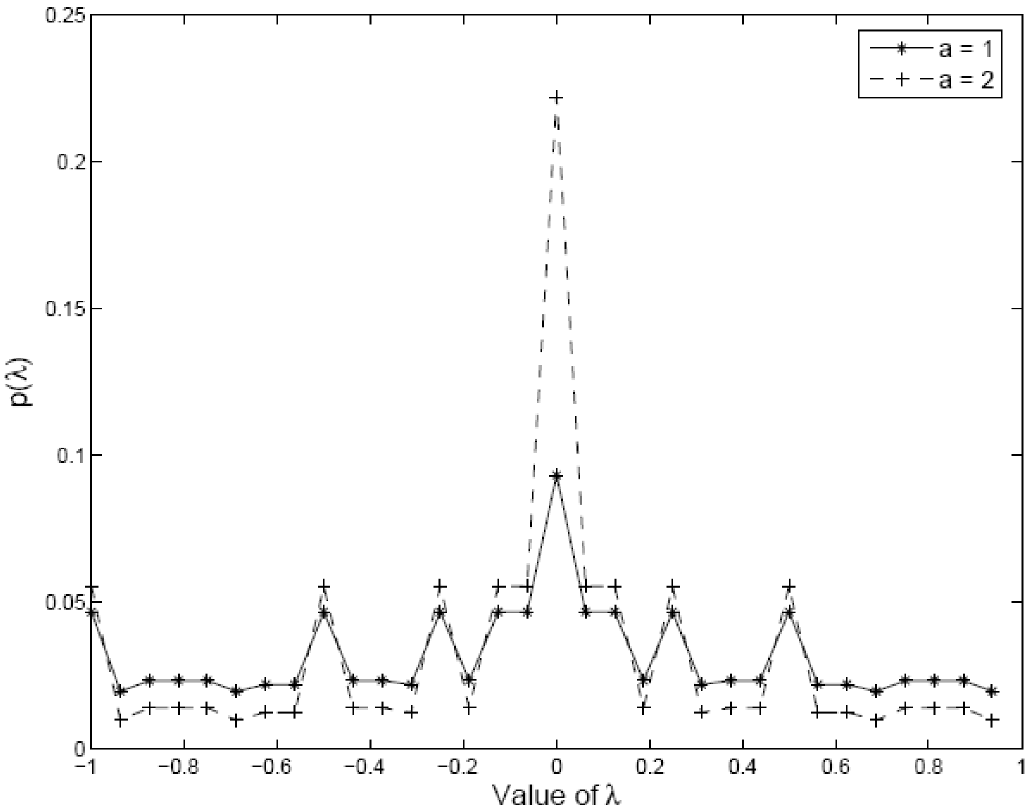
Imperial College London

Mapping area to probability

Target device: XC2V6000

CORE Generator to calculate the area of a multiplier

Flexible model to accommodate other devices



Targeting the heterogeneity of modern FPGAs

Aim: Efficient allocation of embedded multipliers

- *Indicator matrix*

$$Z = \begin{bmatrix} z_{1,1} & z_{1,2} \\ z_{2,1} & z_{2,2} \\ z_{3,1} & z_{3,2} \\ z_{4,1} & z_{4,2} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \qquad \Lambda = \begin{bmatrix} \lambda_{1,1} & \lambda_{1,2} \\ \lambda_{2,1} & \lambda_{2,2} \\ \lambda_{3,1} & \lambda_{3,2} \\ \lambda_{4,1} & \lambda_{4,2} \end{bmatrix}$$

1 indicates embedded multiplier, 0 slice based multiplier

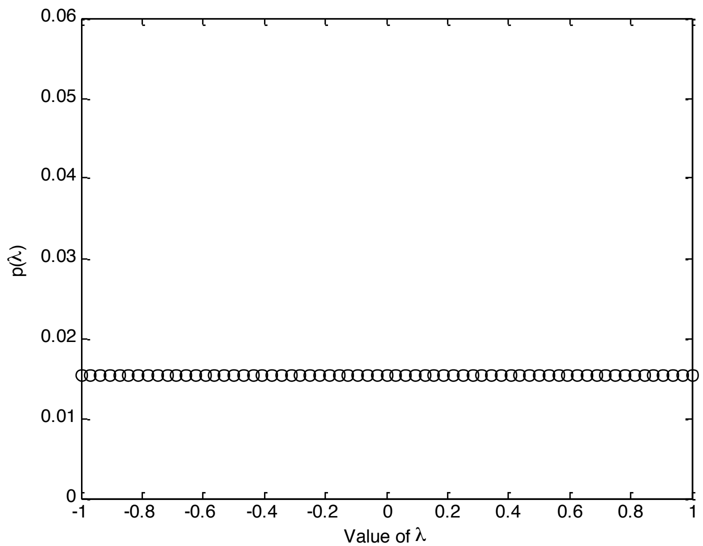
$$p(\lambda_{pk}, z_{pk} | X, F, \Psi) \propto p(X | F, \Lambda, \Psi, Z) \prod_{p=1}^{P} \prod_{k=1}^{K} p(\lambda_{pk}, z_{pk})$$

Targeting the heterogeneity of modern FPGAs (2)

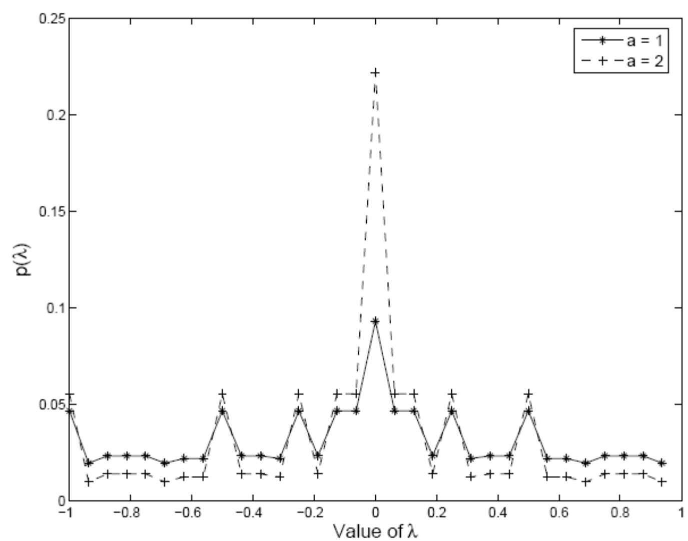
Sampling

- Sample Z through uniform distribution imposing $\sum z_{pk} = N$
- Prior probability distribution has two forms depending on z_{pk}

$$p(\lambda_{pk}, z_{pk} = 1)$$



$$p(\lambda_{pk}, z_{pk} = 0)$$

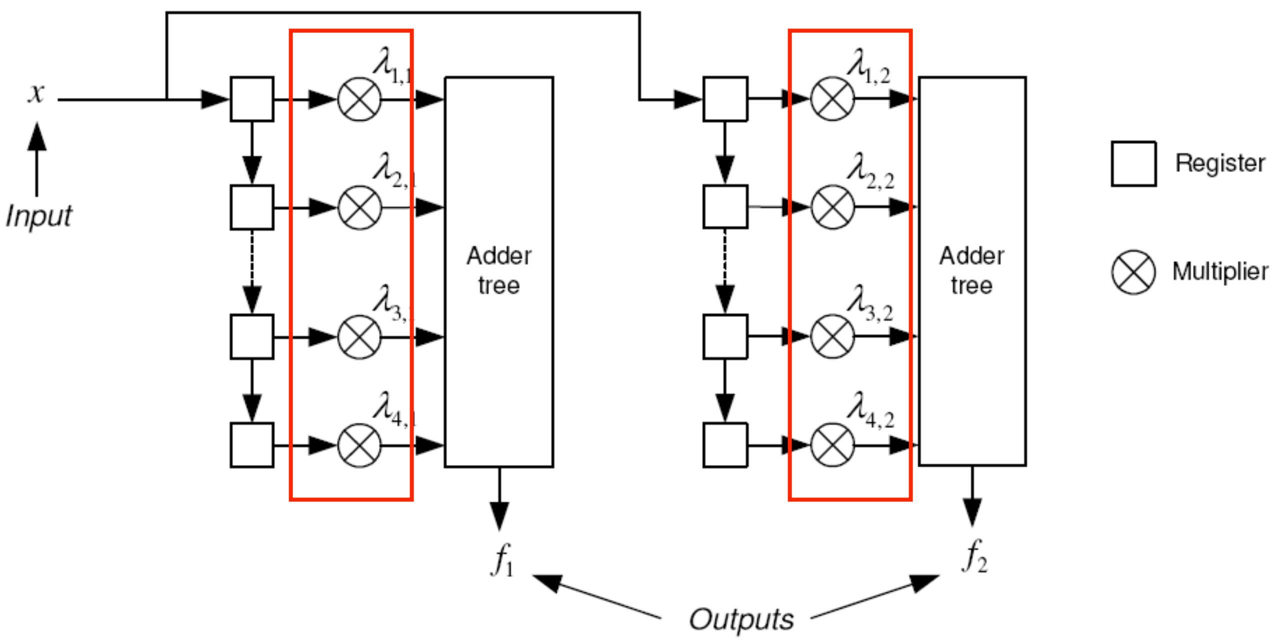


Assumption of independence

Factor Loading matrix

Prior distr.

$$p(\Lambda) = \prod_{p=1}^{P} \prod_{k=1}^{K} p(\lambda_{pk}) \Rightarrow Cost(\Lambda) = \sum_{p=1}^{P} \sum_{k=1}^{K} Cost(\lambda_{pk})$$

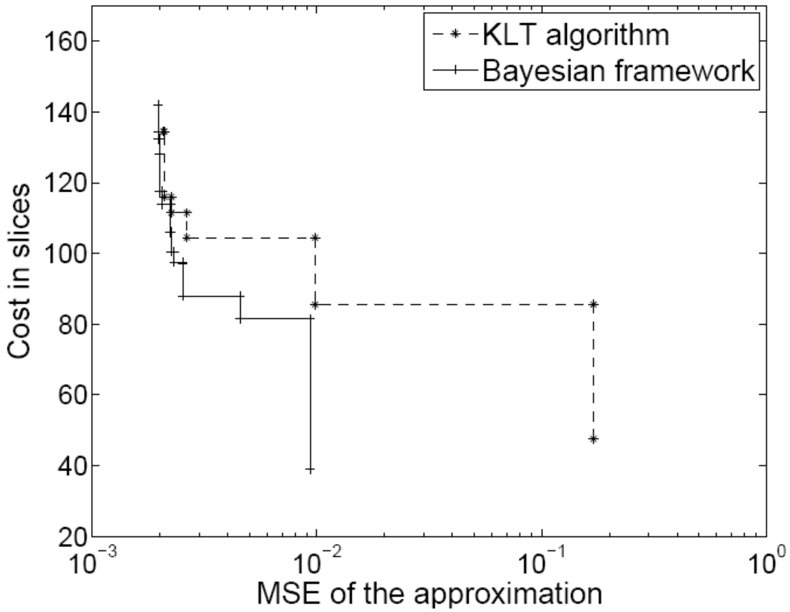


$$\Lambda = \begin{bmatrix} \lambda_{1,1} & \lambda_{1,2} \\ \lambda_{2,1} & \lambda_{2,2} \\ \lambda_{3,1} & \lambda_{3,2} \\ \lambda_{4,1} & \lambda_{4,2} \end{bmatrix}$$

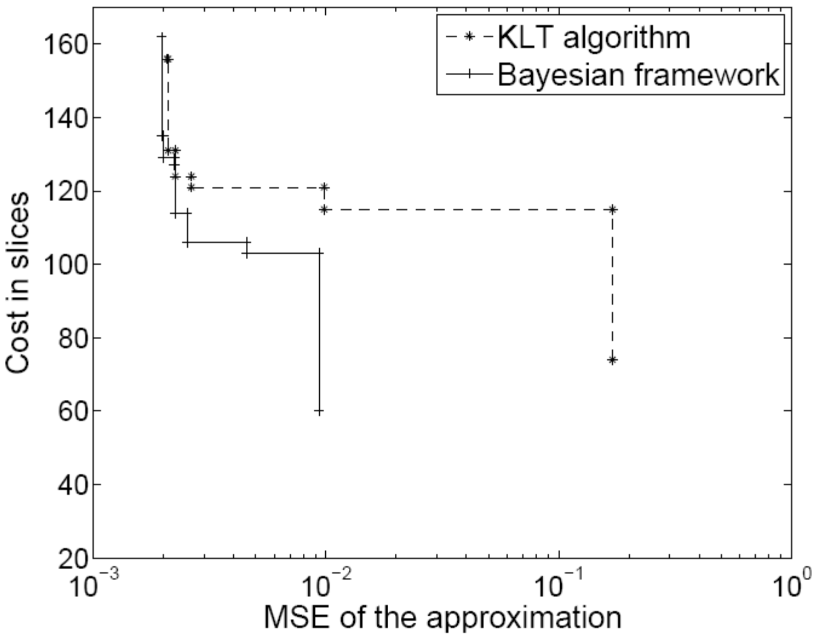
Performance Evaluation

Mapping: $R^3 \rightarrow Z^2$

Estimated results



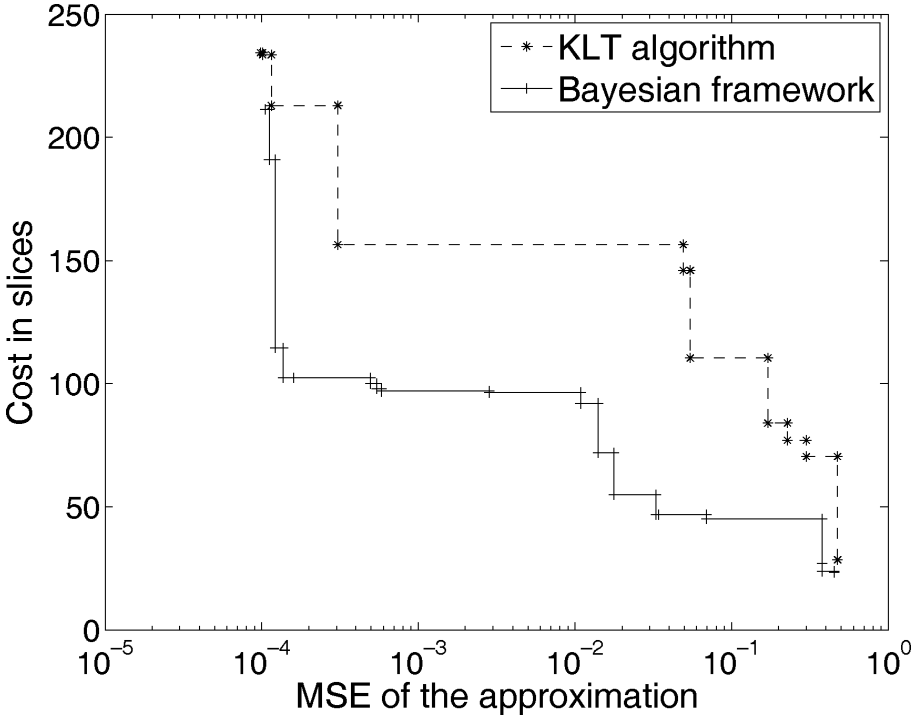
Synthesized results



Imperial College London

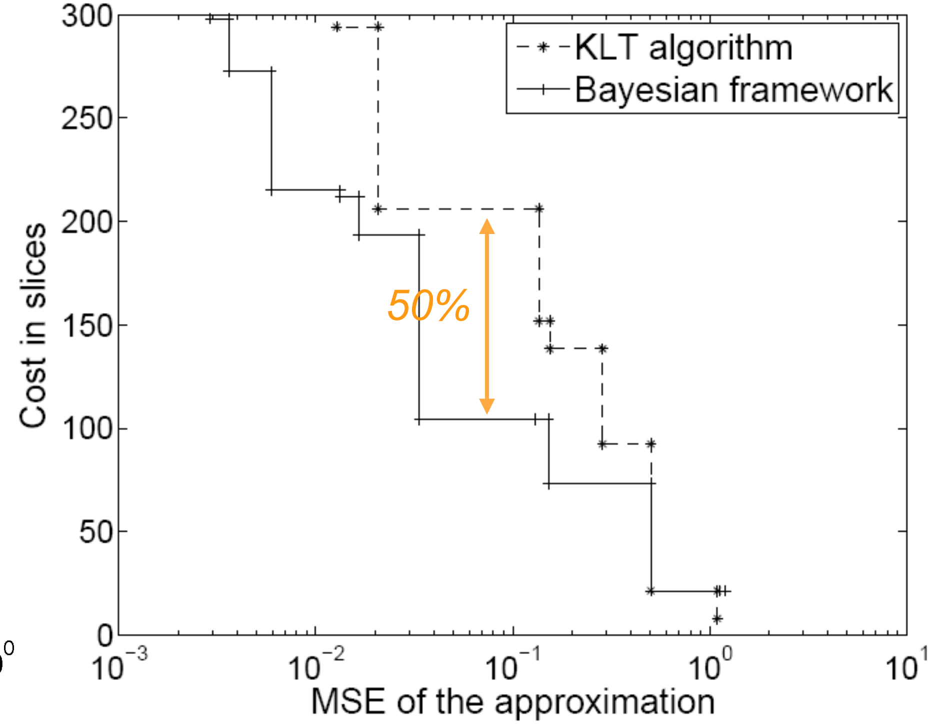
Performance Evaluation (cont')

$$R^4 \rightarrow Z^2$$



2 DSPs and 2 BlockRAMs available

$$R^6 \rightarrow Z^c, c < 6$$

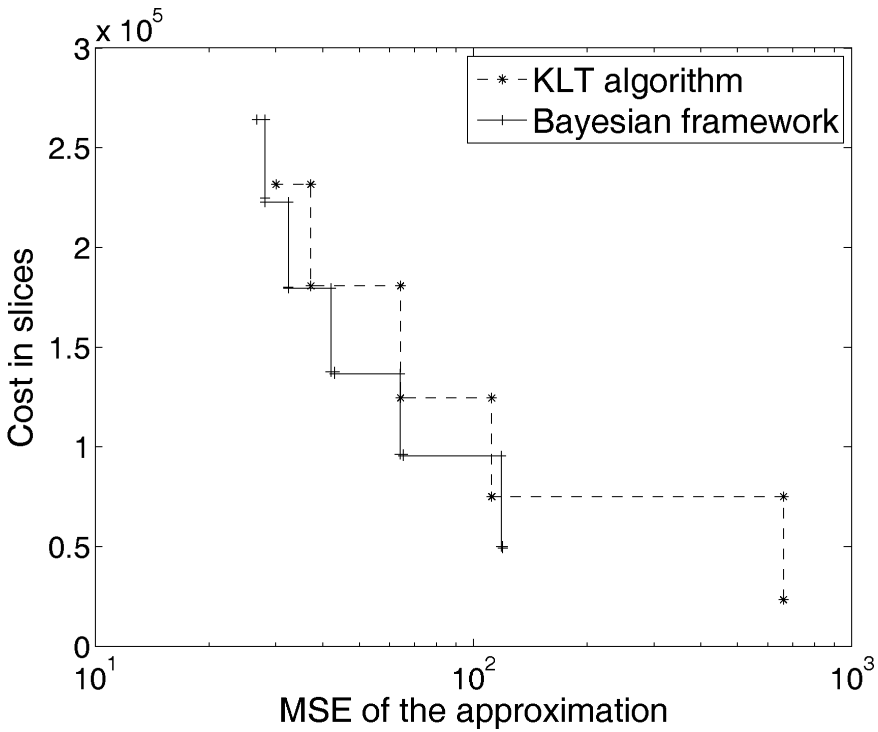


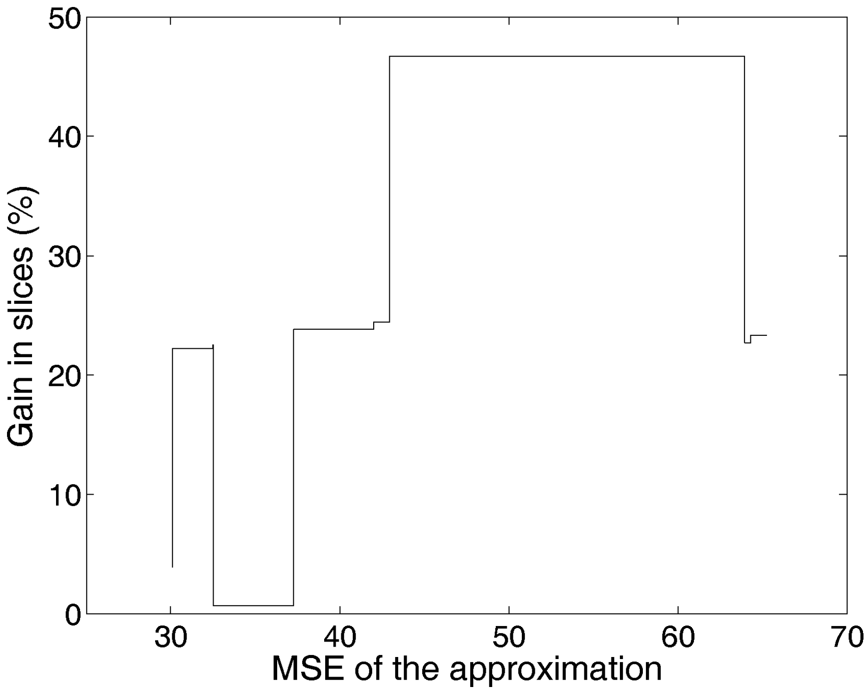
Unconstrained reduced space

Imperial College London

Performance Evaluation - Faces

$$Z^{500} \rightarrow Z^{40}$$





Performance Evaluation - Faces

Same MSE

Original *KLT* *Proposed*

Same FPGA area

Original *KLT* *Proposed*

30% area reduction

Other works in my group

- Ego-motion estimation for UAV navigation
- Real-Time Super-resolution Sensor
- Object detection / recognition (training/classification)
- Design reliable systems with unreliable hardware
- Acceleration of SVM training/classification stages
- Acceleration of Monte Carlo Markov Chain
  - Parallel Tempering for Bayesian Inference
  - Adaptive datapaths for financial instrument calculation

Imperial College London

Summary

- FPGAs offer a good computational platform
  - Power reduction
  - Exploit any parallelism in the algorithm
  - ASIC becomes more and more expensive => FPGA alternative platform

- FPGAs are suitable for image processing
  - Custom number representation
  - Accommodate error => leads to interesting trade-offs

- Current FPGA trends
  - Large devices: (various hard blocks => more coarse grain)
  - Small devices: Low power

- Work on high level tools (languages, libraries, …)
  - Bridge the gap between productivity and available resources

Future

Is FPGA the future?

Probably not.

Heterogeneous Systems + Tools + Libraries

The future will be interesting