Speculations about Computer Architecture in Next Three Years

shuchang.zhou@gmail.com

Jan. 20, 2018

About me

https://zsc.github.io/

● Source-to-source transformation ● Cache simulation	● Natural Language Question & Answer ● Indoor Navigation with INS ● Group Orbit Optimization	● OCR ● Quantized Neural Network ● Smart Camera ● Reinforcement Learning
Compiler Optimization	Machine Learning	Neural Network

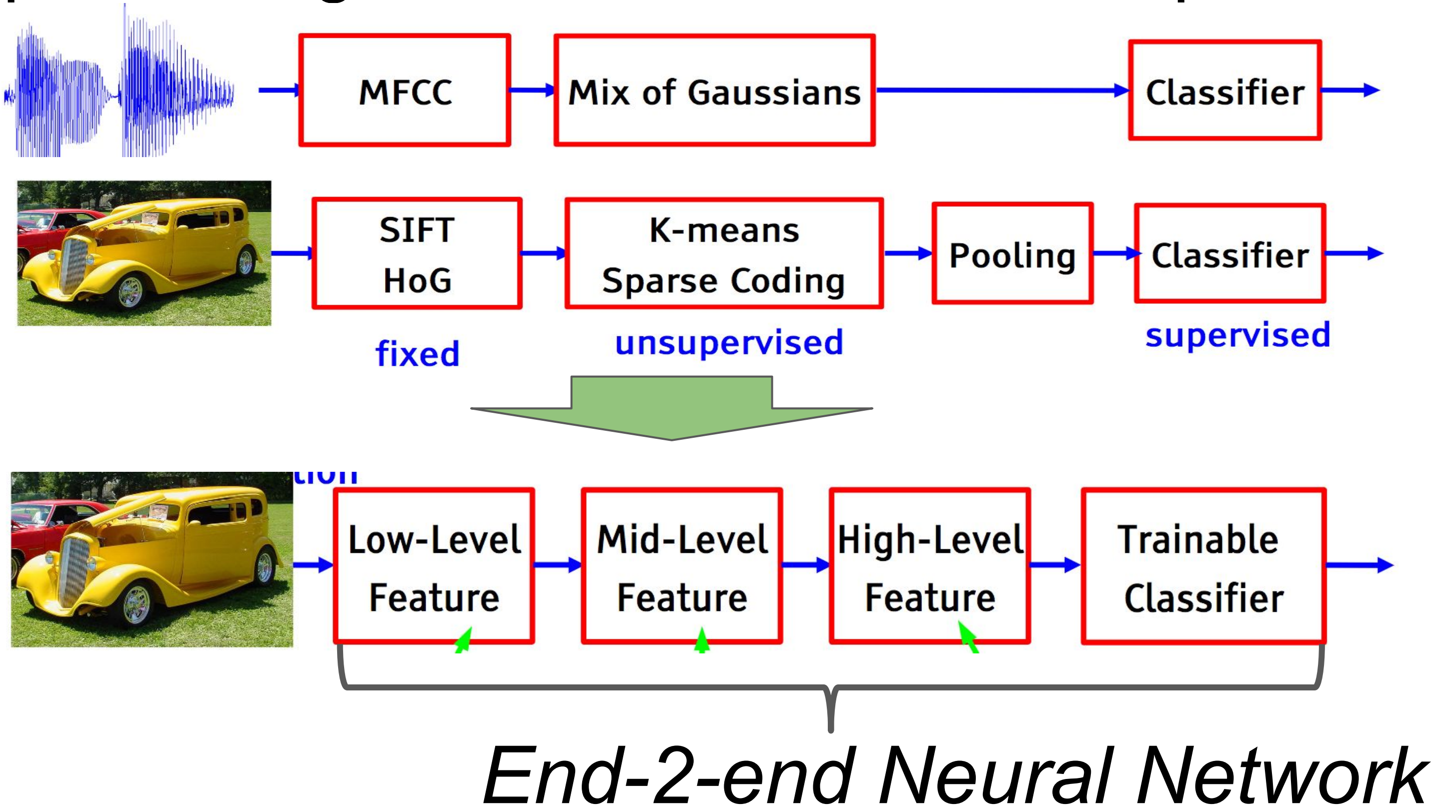
2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018

CHINESE ACADEMY OF SCIENCES

Google

megvii

Deep Learning Revolution in Vision & Speech



MFCC

Mix of Gaussians

Classifier

SIFT
HoG

K-means
Sparse Coding

Pooling

Classifier

fixed

unsupervised

supervised

tion

Low-Level Feature

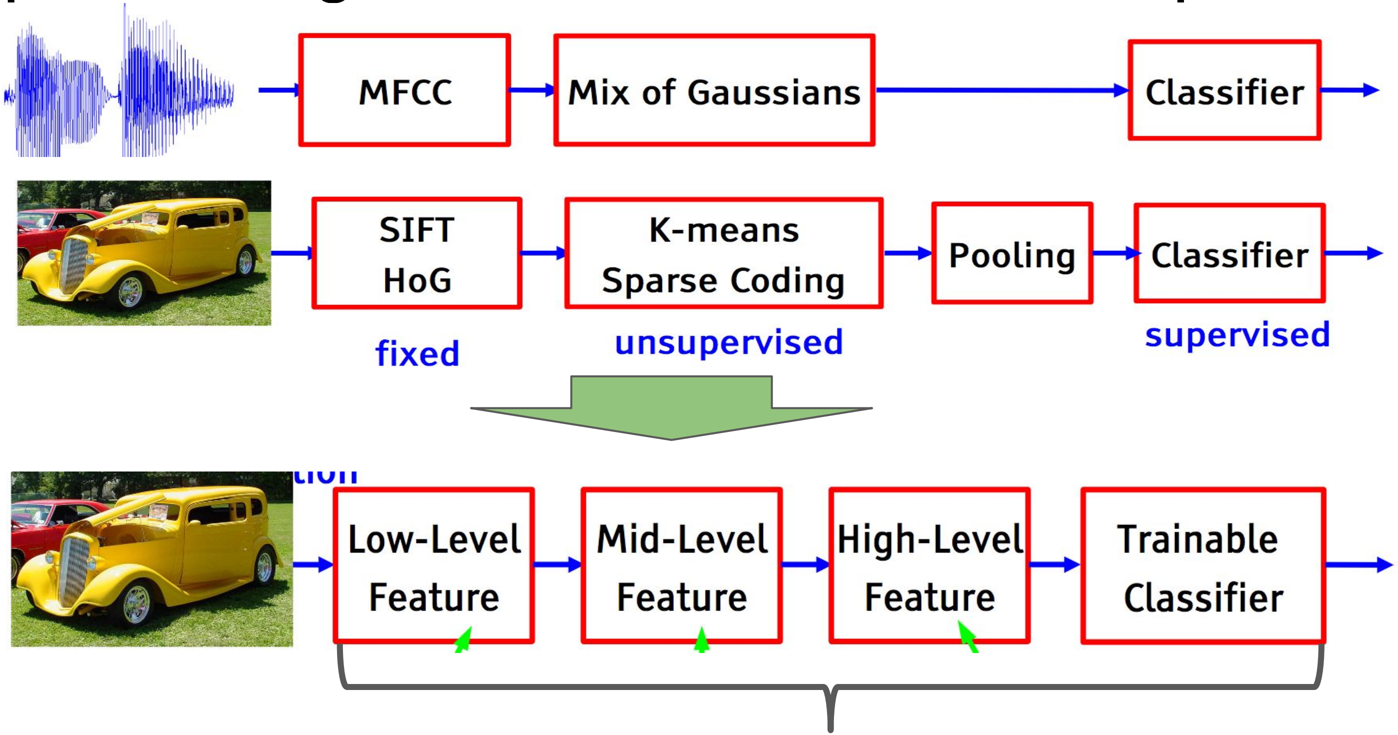
Mid-Level Feature

High-Level Feature

Trainable Classifier

End-2-end Neural Network

Deep Learning Revolution in Vision & Speech

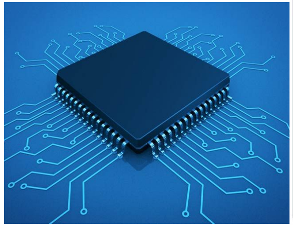


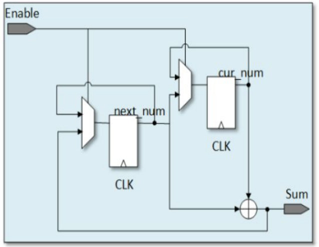
Differentiable Forward & Backward

Implications of Deep Learning

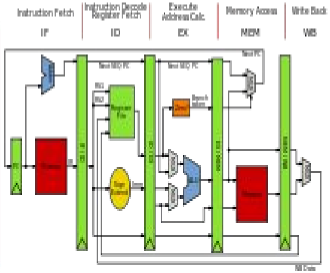
- Unification of Algorithms in Vision & Speech
  - Deep Learning v.s. “Traditional methods”
- Graph execution engine as the new platform
  - For CNN / RNN
- A new wave of data centers
  - Google / Facebook: millions of GPU
  - Startups: thousands of GPU
- Adjoints of Neural Networks
  - Image augmentor
  - Simulators

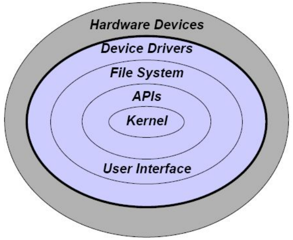
Computation Stack

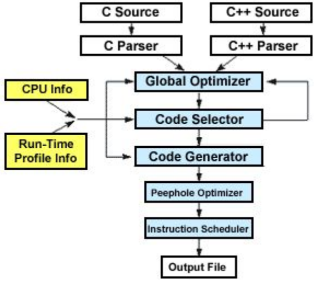


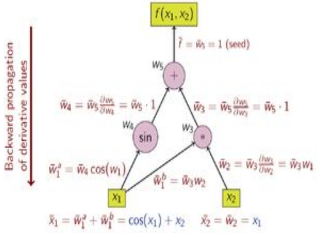


Circuit to Generate Fibonacci Series *(Fig. 13)*









Backward propagation of derivative values
$f(x_1, x_2)$
$\bar{f} = \bar{w}_5 = 1$ (seed)
w_5
+
$\bar{w}_4 = \bar{w}_5 \frac{\partial w_5}{\partial w_4} = \bar{w}_5 \cdot 1$
$\bar{w}_3 = \bar{w}_5 \frac{\partial w_5}{\partial w_3} = \bar{w}_5 \cdot 1$
w_4
sin
w_3
*
$\bar{w}_1^a = \bar{w}_4 \cos(w_1)$
$\bar{w}_1^b = \bar{w}_3 w_2$
$\bar{w}_2 = \bar{w}_3 \frac{\partial w_3}{\partial w_2} = \bar{w}_3 w_1$
x_1
x_2
$\bar{x}_1 = \bar{w}_1^a + \bar{w}_1^b = \cos(x_1) + x_2$
$\bar{x}_2 = \bar{w}_2 = x_1$

Silicon

- Partitioning & Planning
- Place & Route
- Timing Closure

Verilog

- Karnaugh map
- Finite State Machine

Architecture

- ISA
- Micro-code
- Resource allocation

Operating System

- Page table
- File system
- Interrupts

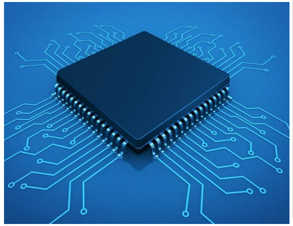
Compiler

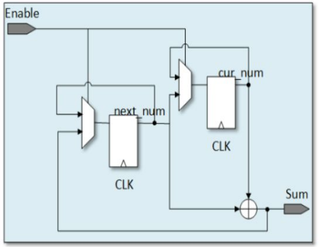
- Parallelism mining
- Memory latency hiding

Computation Graph Engine

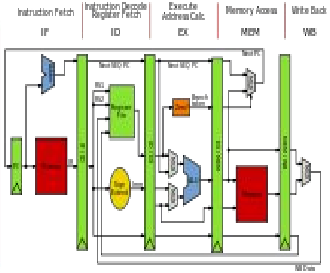
- Kernels
- Execution Plan

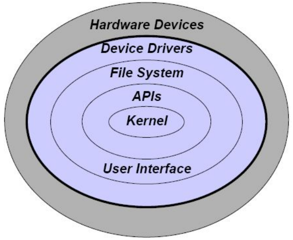
Computation Stack

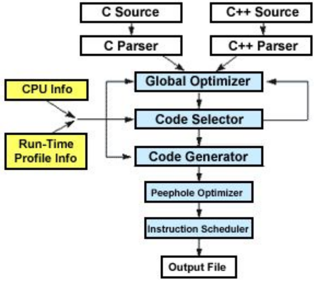


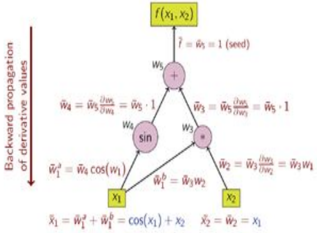


Circuit to Generate Fibonacci Series *(Fig. 13)*








Backward propagation of derivative values
$f(x_1, x_2)$
$\bar{f} = \bar{w}_5 = 1$ (seed)
w_5
+
$\bar{w}_4 = \bar{w}_5 \frac{\partial w_5}{\partial w_4} = \bar{w}_5 \cdot 1$
$\bar{w}_3 = \bar{w}_5 \frac{\partial w_5}{\partial w_3} = \bar{w}_5 \cdot 1$
w_4
sin
w_3
*
$\bar{w}_1^a = \bar{w}_4 \cos(w_1)$
$\bar{w}_1^b = \bar{w}_3 w_2$
$\bar{w}_2 = \bar{w}_3 \frac{\partial w_3}{\partial w_2} = \bar{w}_3 w_1$
x_1
x_2
$\bar{x}_1 = \bar{w}_1^a + \bar{w}_1^b = \cos(x_1) + x_2$
$\bar{x}_2 = \bar{w}_2 = x_1$

Silicon
- Partitioning & Planning
- Place & Route
- Timing Closure

Verilog
- Karnaugh map
- Finite State Machine

Architecture
- ISA
- Micro-code
- Resource allocation

Operating System
- Page table
- File system
- Interrupts

Compiler
- Parallelism mining
- Memory latency hiding

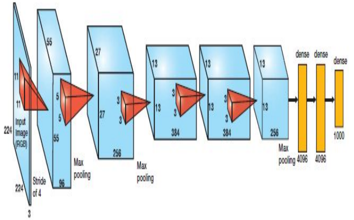
Computation Graph Engine
- Kernels
- Execution Plan

How will this stack deal with changes?

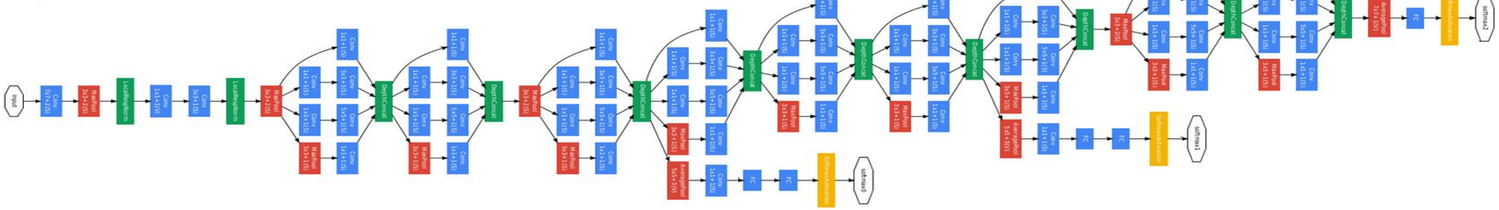
Case study: Large Neural Networks

Characteristics: many channels + side-branches + many layers

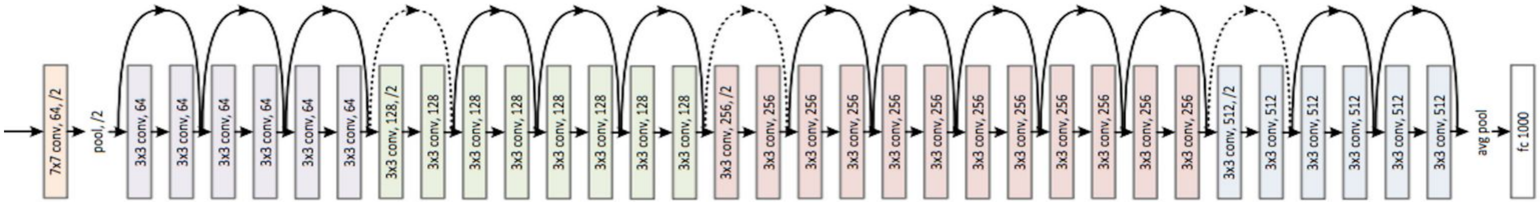
AlexNet



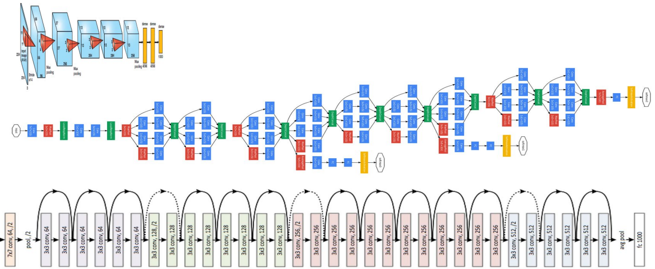
GoogLeNet



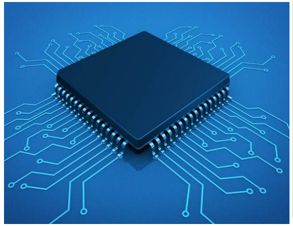
ResNet



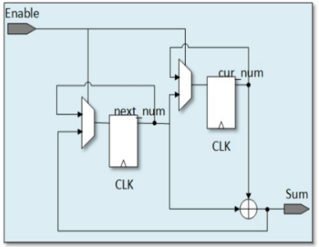
Case study: Large Neural Networks



On-Chip-Memory for caching feature maps



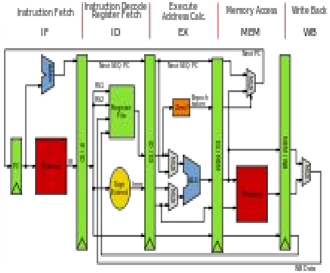
Silicon



Circuit to Generate Fibonacci Series *(Fig. 13)*

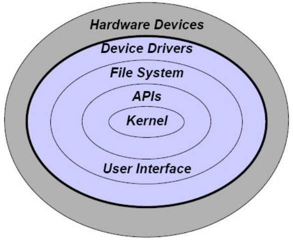
Verilog

- Instructions for convolutions & non-linearity
- Systolic Array



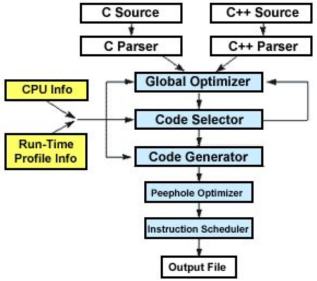
Architecture

Large page-table



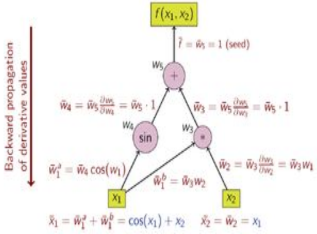
Operating System

Auto-SIMD



Compiler

Static analysis + dynamic profiling for kernel selection + execution plan

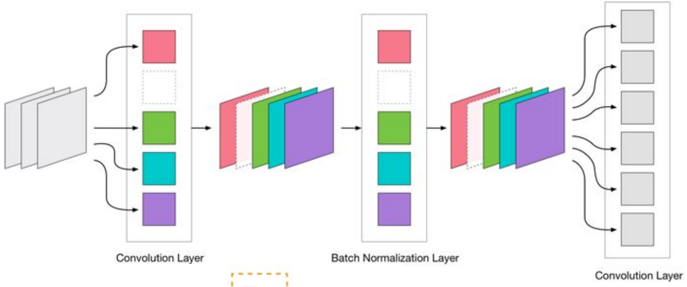


Computation Graph Engine

Case study: Small Neural Networks

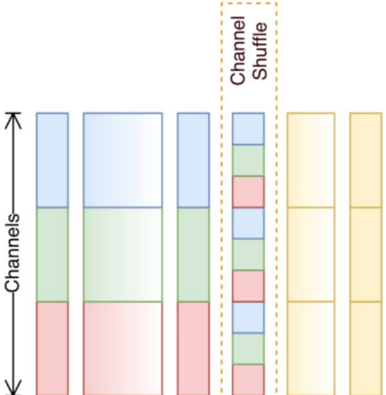
Characteristics: few channels + 1x1 convolutions

MobileNet



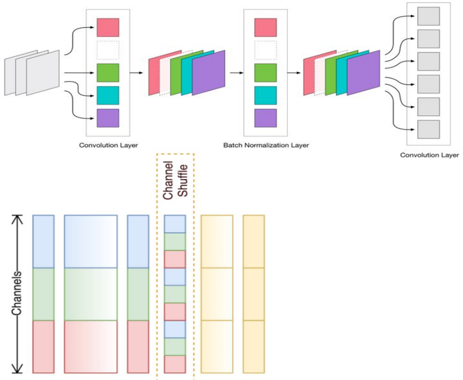
Lack of shortcut hurts its transfer learning ability.

ShuffleNet

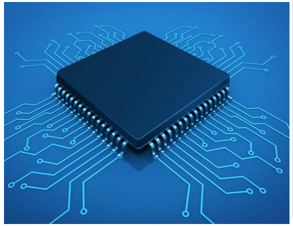


The unique shuffle operation slows its adoption.

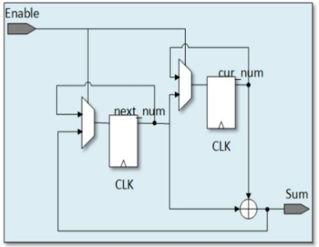
Case study: Small Neural Networks



On-Chip-Memory may be more important.



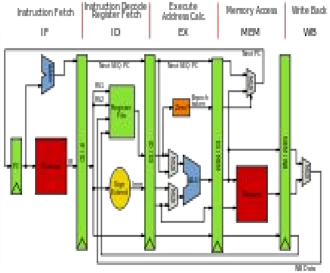
Silicon



Circuit to Generate Fibonacci Series *(Fig. 13)*

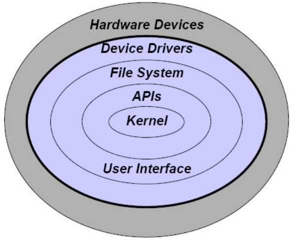
Verilog

- Specialized support for few channel layers and 1x1 convolutions.
- Different batching



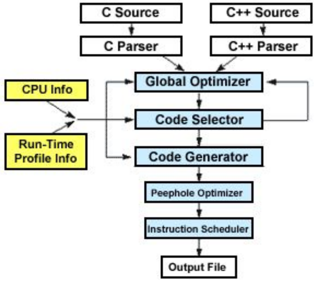
Architecture

Lower overhead



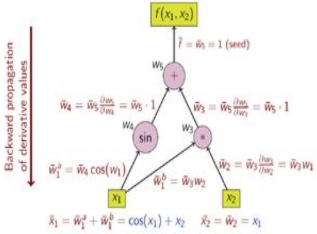
Operating System

Auto-SIMD



Compiler

Fusion of layers + handcrafted kernels


$f(x_1, x_2)$
$\bar{f} = \bar{w}_5 = 1$ (seed)
w_5
+
$\bar{w}_4 = \bar{w}_5 \frac{\partial w_5}{\partial w_4} = \bar{w}_5 \cdot 1$
$\bar{w}_3 = \bar{w}_5 \frac{\partial w_5}{\partial w_3} = \bar{w}_5 \cdot 1$
w_4
sin
w_3
*
$\bar{w}_1^a = \bar{w}_4 \cos(w_1)$
$\bar{w}_1^b = \bar{w}_3 w_2$
$\bar{w}_2 = \bar{w}_3 \frac{\partial w_3}{\partial w_2} = \bar{w}_3 w_1$
x_1
x_2
$\bar{x}_1 = \bar{w}_1^a + \bar{w}_1^b = \cos(x_1) + x_2$
$\bar{x}_2 = \bar{w}_2 = x_1$
Backward propagation of derivative values

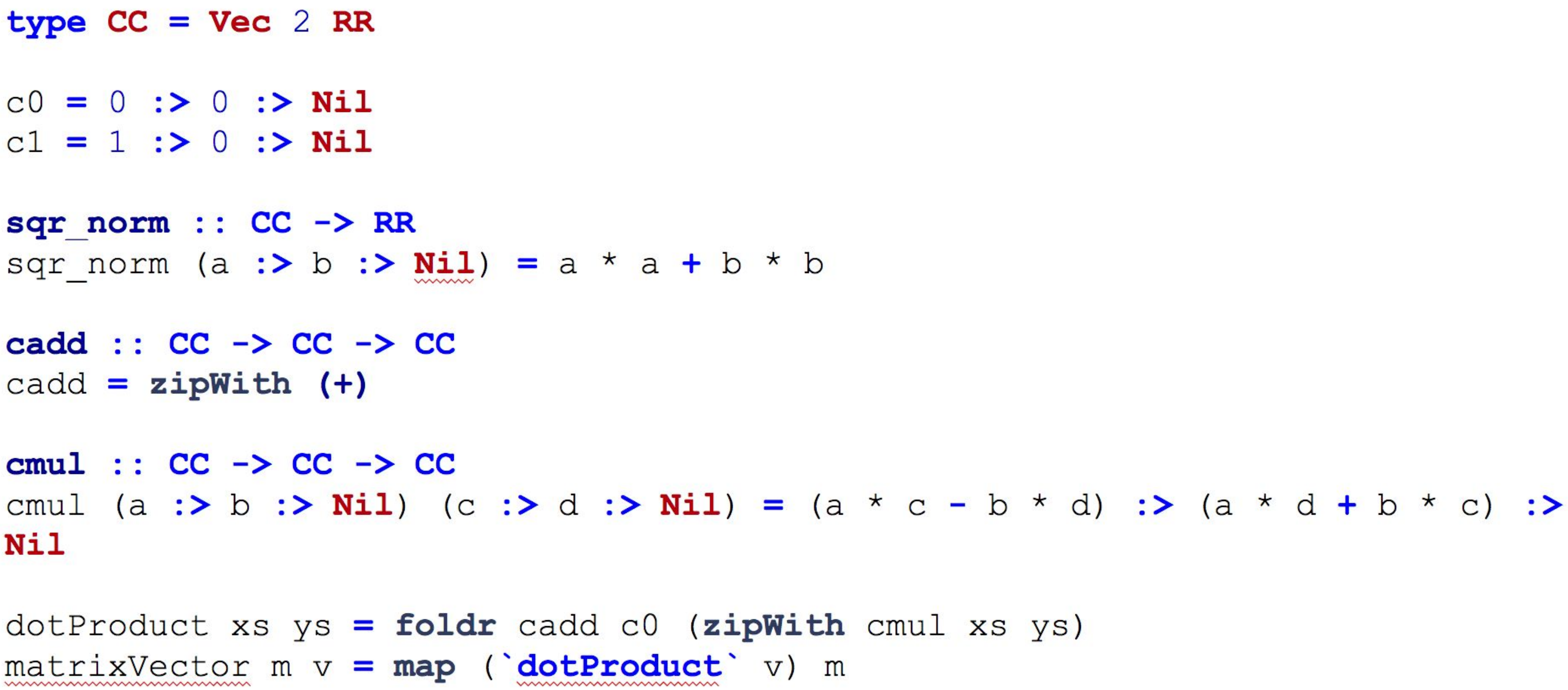
Computation Graph Engine

When a Neural Network Designers, a Computer Architect, a Compiler Expert and an OS Guru meet

- Designer wants
  - A reliable performance model
    - Open architecture design and assembly/microcode level exposure
  - Better profilers for runtime diagnostics and analyzers
  - Support for sparse matrices, dynamic operations
- Architect wants
  - Batch operations with constant delays
  - Regular memory access pattern subject to locality and many reuses
  - Streamlined memory/computation usage, no overwhelming peaks
  - Less number of operators
- Compiler Expert and OS Guru wants
  - To broker between the Designer and the Architect
    - Have a slow fallback for bizarre operators
    - Cutting peaks

Case study: Quantum Computing Simulator on FPGA

Clash/FPGA: implement Complex Number

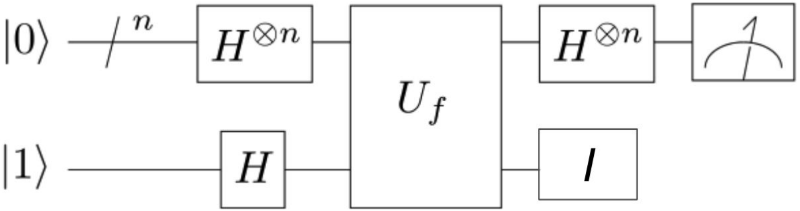


```
type CC = Vec 2 RR

c0 = 0 :> 0 :> Nil
c1 = 1 :> 0 :> Nil

sqr_norm :: CC -> RR
sqr_norm (a :> b :> Nil) = a * a + b * b

cadd :: CC -> CC -> CC
cadd = zipWith (+)

cmul :: CC -> CC -> CC
cmul (a :> b :> Nil) (c :> d :> Nil) = (a * c - b * d) :> (a * d + b * c) :>
Nil

dotProduct xs ys = foldr cadd c0 (zipWith cmul xs ys)
matrixVector m v = map (`dotProduct` v) m
```

Case study: Quantum Computing Simulator on FPGA

HLS may be sufficiently efficient and flexible

Deutsch-Jozsa's algorithm



```
deutsch_u :: Vec 2 RR -> Vec 4 CC -> Vec 4 CC
deutsch_u (f0 :> f1 :> Nil) =
  matrixVector (make complex (
                 ((1 - f0) :> f1 :> 0 :> 0 :> Nil) :>
                 (f0 :> (1 - f1) :> 0 :> 0 :> Nil) :>
                 (0 :> 0 :> (1 - f0) :> f1 :> Nil) :>
                 (0 :> 0 :> f0 :> (1 - f1) :> Nil) :> Nil))
```

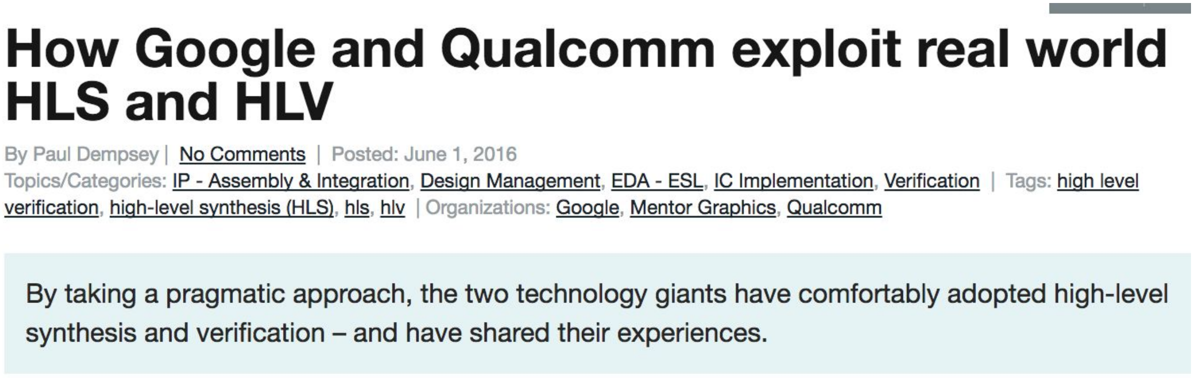
$$\begin{pmatrix} 1-f0 & f1 & 0 & 0 \\ f0 & 1-f1 & 0 & 0 \\ 0 & 0 & 1-f0 & f1 \\ 0 & 0 & f0 & 1-f1 \end{pmatrix}$$

```
hadamard_I :: Vec 4 CC -> Vec 4 CC
hadamard_I =
  matrixVector (make complex (
                 (h :> 0 :> h :> 0 :> Nil) :>
                 (0 :> h :> 0 :> h :> Nil) :>
                 (h :> 0 :> - h :> 0 :> Nil) :>
                 (0 :> h :> 0 :> - h :> Nil) :> Nil))
  where h = $$(fLit (1 / sqrt 2)) :: RR
```

$$H \otimes I = \begin{pmatrix} h & 0 & h & 0 \\ 0 & h & 0 & h \\ h & 0 & -h & 0 \\ 0 & h & 0 & -h \end{pmatrix}$$

A possible future

Design Silicon Compiler!



How Google and Qualcomm exploit real world HLS and HLV

By Paul Dempsey | No Comments | Posted: June 1, 2016
Topics/Categories: IP - Assembly & Integration, Design Management, EDA - ESL, IC Implementation, Verification | Tags: high level verification, high-level synthesis (HLS), hls, hlv | Organizations: Google, Mentor Graphics, Qualcomm

By taking a pragmatic approach, the two technology giants have comfortably adopted high-level synthesis and verification – and have shared their experiences.

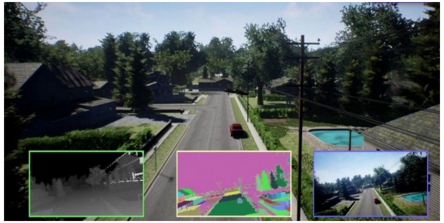
Case study: Reinforcement Learning

Characteristics: require fast & complex simulations

OpenSim

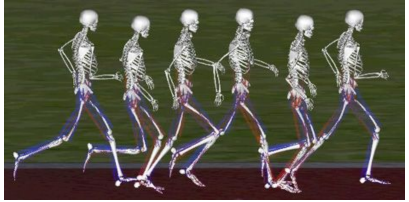
A human skeleton model for locomotive task modeling.

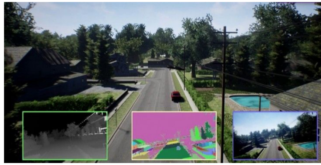
GTA 5
AirSim



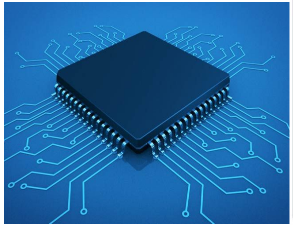
Simulation for self-driving car/ADAS and Drones.

Case study: Reinforcement Learning

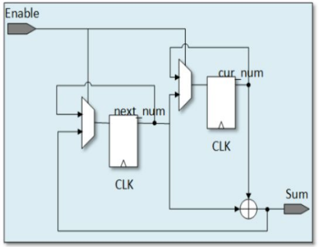




Typical CPU load, but need to integrate with Neural Network Accelerator

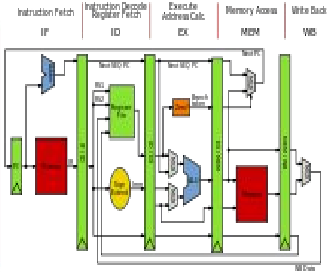


Silicon

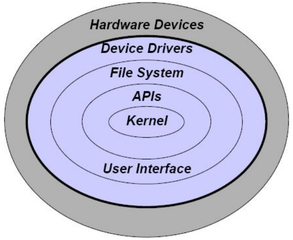


Circuit to Generate Fibonacci Series *(Fig. 13)*

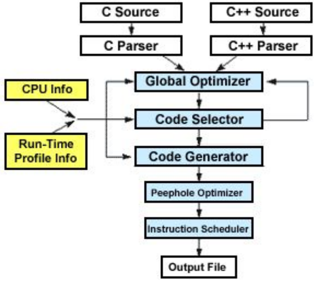
Verilog



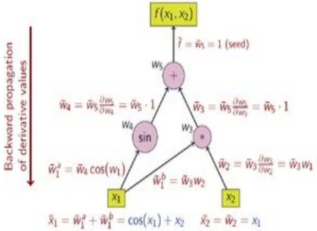
Architecture



Operating System



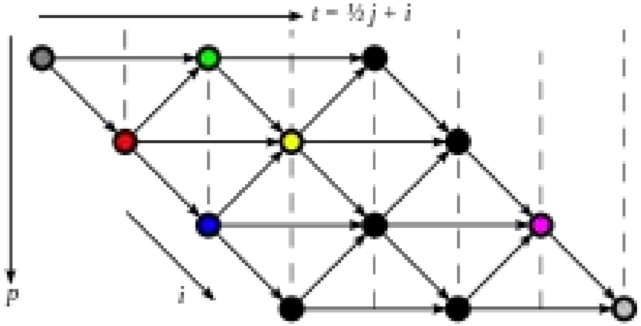
Compiler



Computation Graph Engine

A possible future

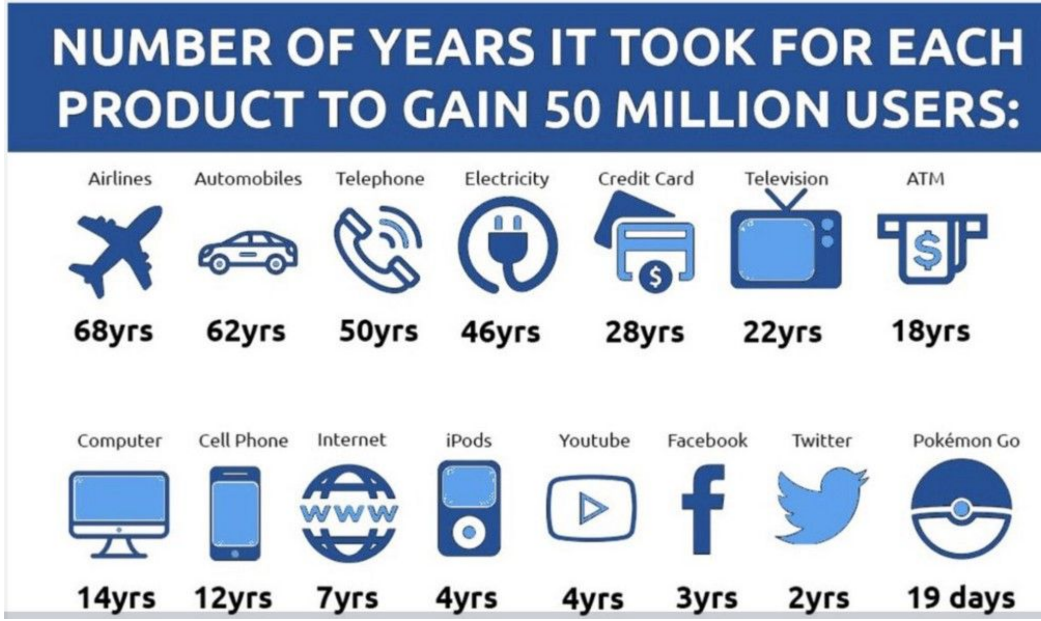
Revival of Compiler Optimizations!



Should we prepare a benchmark of simulators?

The Age of Instant Response

- Old School
  - Compiler cannot change code
  - Developer as the dictator
  - Batch operation and buffering
  - Conference & Journal
- New School
  - Compiler can offer suggestions
  - User Community
    - User code contributions
    - Peer-to-peer helping
  - Low latency is critical
  - Arxiv & http://www.arxiv-sanity.com/



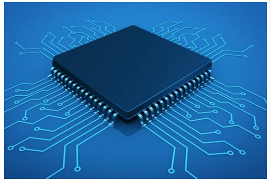
NUMBER OF YEARS IT TOOK FOR EACH PRODUCT TO GAIN 50 MILLION USERS:

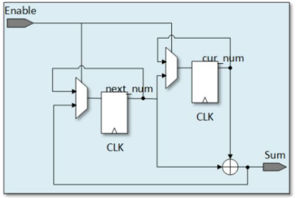
Airlines	68yrs
Automobiles	62yrs
Telephone	50yrs
Electricity	46yrs
Credit Card	28yrs
Television	22yrs
ATM	18yrs
Computer	14yrs
Cell Phone	12yrs
Internet	7yrs
iPods	4yrs
Youtube	4yrs
Facebook	3yrs
Twitter	2yrs
Pokémon Go	19 days

The combined future ...

Performance critical

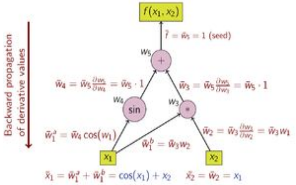
TPU / FPGA



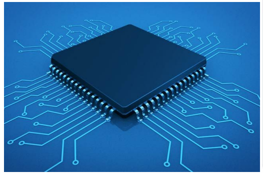


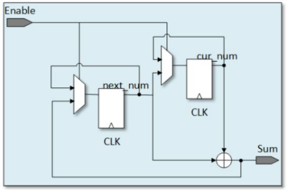
Circuit to Generate Fibonacci Series *(Fig. 13)*

TensorFlow

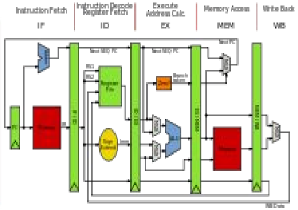

$f(x_1, x_2)$
$\bar{f} = \bar{w}_5 = 1$ (seed)
w_5
+
$\bar{w}_4 = \bar{w}_5 \frac{\partial w_5}{\partial w_4} = \bar{w}_5 \cdot 1$
$\bar{w}_3 = \bar{w}_5 \frac{\partial w_5}{\partial w_3} = \bar{w}_5 \cdot 1$
w_4
sin
w_3
*
Backward propagation of derivative values
$\bar{w}_1^a = \bar{w}_4 \cos(w_1)$
$\bar{w}_1^b = \bar{w}_3 w_2$
$\bar{w}_2 = \bar{w}_3 \frac{\partial w_3}{\partial w_2} = \bar{w}_3 w_1$
x_1
x_2
$\bar{x}_1 = \bar{w}_1^a + \bar{w}_1^b = \cos(x_1) + x_2$
$\bar{x}_2 = \bar{w}_2 = x_1$

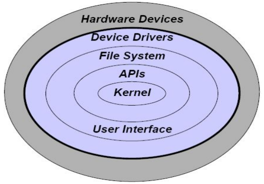
Complex coordination

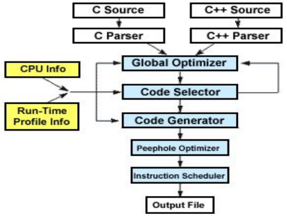


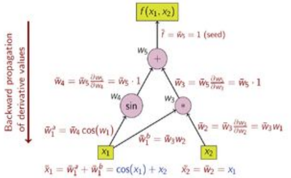


Circuit to Generate Fibonacci Series *(Fig. 13)*








$f(x_1, x_2)$
$\bar{f} = \bar{w}_5 = 1$ (seed)
w_5
+
$\bar{w}_4 = \bar{w}_5 \frac{\partial w_5}{\partial w_4} = \bar{w}_5 \cdot 1$
$\bar{w}_3 = \bar{w}_5 \frac{\partial w_5}{\partial w_3} = \bar{w}_5 \cdot 1$
w_4
sin
w_3
*
Backward propagation of derivative values
$\bar{w}_1^a = \bar{w}_4 \cos(w_1)$
$\bar{w}_1^b = \bar{w}_3 w_2$
$\bar{w}_2 = \bar{w}_3 \frac{\partial w_3}{\partial w_2} = \bar{w}_3 w_1$
x_1
x_2
$\bar{x}_1 = \bar{w}_1^a + \bar{w}_1^b = \cos(x_1) + x_2$
$\bar{x}_2 = \bar{w}_2 = x_1$

Backup after this slide