

FPGAs, HLS Tools & Runtime Systems

(Super)Advisors: Frederic Desprez, Francois Broquedis, Olivier Muller

Georgios Christodoulis

CORSE-LIG

gchristodoulis@gmail.com

Overview

FPGAs structure

Description

Look-Up Table

Basic Logic Element

Overview

Optimization Using HLS tools

Problem Description

Serial Version

Opt1: Inner Loop Unrolling

Opt2: Pipeline

Opt3: J-loop Unrolling

Opt4: Array Partitioning

Opt5: Maximize Resource
Use

Opt6: Improve I/O

Opt6: Block Computation

Optimizations Review

FPGA

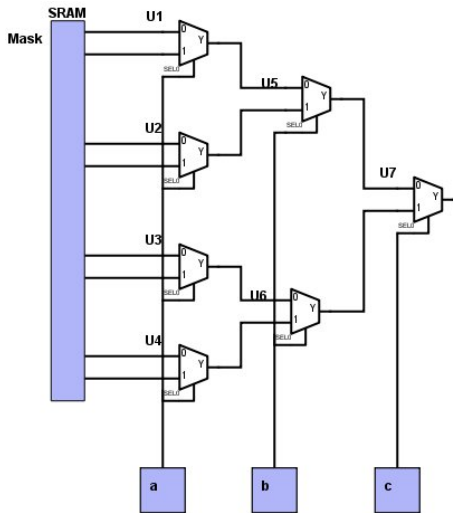
Description

A *Field Programmable Gate Array* is an integrated circuit designed to be configured by a customer or a designer after manufacturing – hence "Field-programmable".

FPGAs are semiconductor devices that are based around a matrix of configurable logic blocks (BLEs) connected via programmable interconnects.

FPGAs Structure

LUT



- ▶ It is a **table** that determines what the output is for any given input
- ▶ A **state-less** interconnection of any number of gates (no feedback loops)
- ▶ Implemented *multiplexing* a combination of SRAM bits

Figure: 3 stages of 2x1 MUX

FPGAs Structure

LUT Example

$$y = (a + b) \cdot c$$

a	b	c	y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

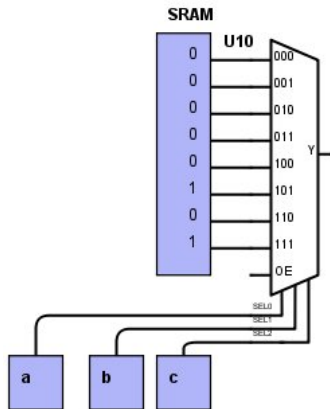


Figure: $y = (a + b) \cdot c$

FPGAs structure

BLE

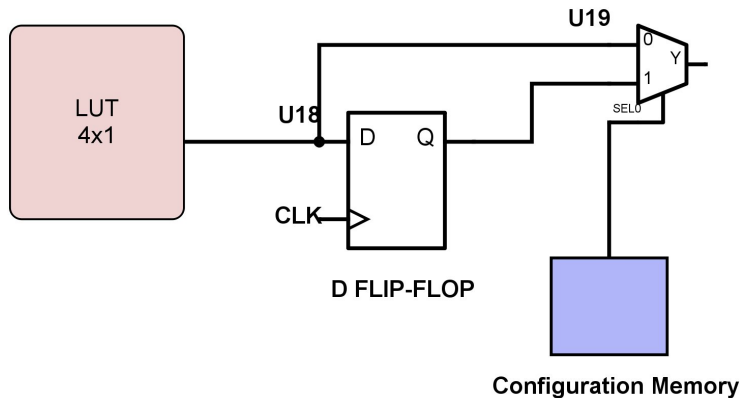


Figure: Basic Logic Element

FPGAs structure

Overview

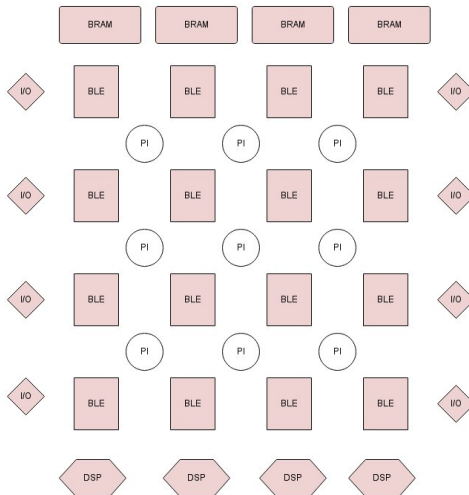


Figure: FPGAs Complete Overview

Problem Description

Matrix Multiplication

$$C = A * B$$

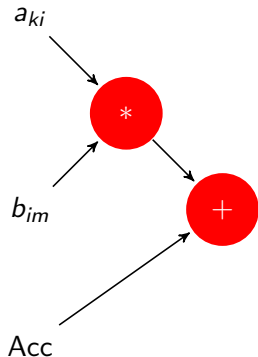
$$c_{ij} = \sum_{k=1}^n a_{ik} b_{kj}$$

$$\begin{vmatrix} c_{11} & \dots & c_{1n} \\ \vdots & c_{km} & \vdots \\ c_{n1} & \dots & c_{nn} \end{vmatrix} \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ \vdots & \vdots & & \vdots \\ a_{k1} & a_{k2} & \dots & a_{kn} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix} \begin{vmatrix} b_{11} & \dots & b_{1m} & \dots & b_{1n} \\ b_{21} & \dots & b_{2m} & \dots & b_{2n} \\ \vdots & & \vdots & & \vdots \\ b_{n1} & \dots & b_{nm} & \dots & b_{nn} \end{vmatrix}$$

No Directives

$$c_{ij} = \sum_{k=1}^n a_{ik} b_{kj}$$

$$c_{km} = a_{k1}b_{1m} + a_{k2}b_{2m} + \cdots + a_{kn}b_{nm}$$

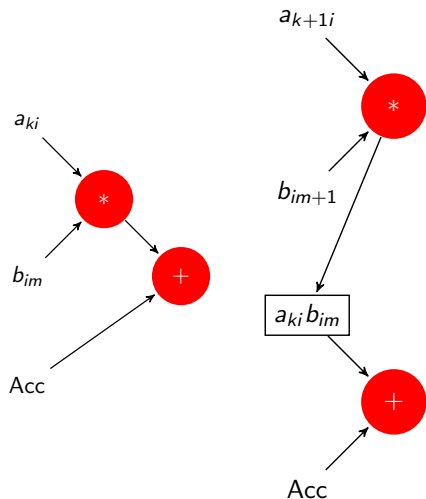


- ▶ $2n$ operations \forall element
 - ▶ n^2 elements
- $\Rightarrow 2n^3$ operations

Inner Loop Unrolling

Opt1: Sum Mul Overlapping

Our reference time interval is defined by the slowest operation which is the multiplication.

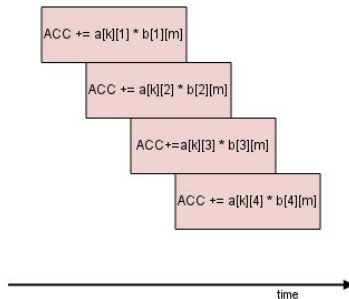
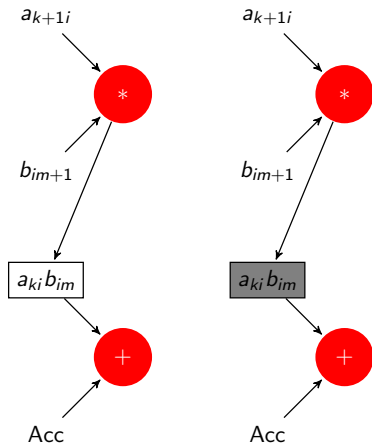


- Complete overlapping between Multiplication and addition

Pipeline

Initiation Interval is called the number of cycles between two new iterations.

In this case it is indicated by the time that the addition register is occupied.



J-loop unrolling

$a_{k1} \quad a_{k2} \quad \dots \quad a_{kn}$

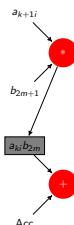
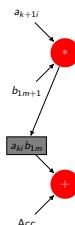
- ▶ Extra Hardware
- ▶ Real Parallelism (Superscalar CPU architectures)
- ▶ Expected scaling: $\times n$
- ▶ Actual scaling: $\times 2$

b_{11}
 b_{21}
 \vdots
 b_{n1}

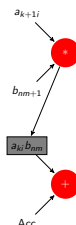
b_{12}
 b_{22}
 \vdots
 b_{n2}

\dots
 \dots
 \dots

b_{1n}
 b_{2n}
 \vdots
 b_{nn}



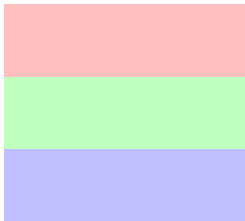
\dots



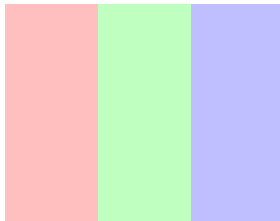
Memory Bounds: Dual Channeled memory \implies only 2 concurrent operations.

Row / Col Partitioning

A



B



- Distributing arrays into different BRAMs increases scaling proportionally to hardware addition

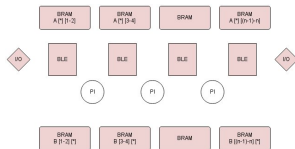


Figure: Distribute the array into multiple BRAMS

Maximize Resource Use

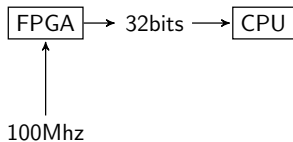
Maximize DSP use

- ▶ Since we have succeeded $II=1$ we increase n in order to maximize DSP usage
- ▶ In our case $n=32 \implies 72\%$ usage while $n=43 \implies 98\%$ usage

Maximize BRAM use

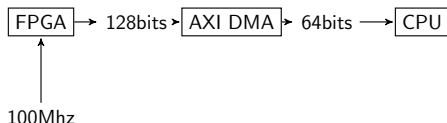
- ▶ Increasing n increases BRAM usage too

Improve I/O



- ▶ $100Mhz \times 32bits = 400MB/s$
- ▶ So the default communication speed is 400MB/s

Bottomline: We increased our communication bandwidth from 400MB/s to 1.2GB/s



- ▶ $100Mhz \times 128bits = 1.6GB/s$
- ▶ AXI DMA's limit is 1.2GB/s at 64bits channel width

Block Computation

$$c_{ij} = \sum_{k=0}^{m-1} A_{ik} B_{kj}$$

CPU FPGA

- ▶ Multiplications on FPGA
- ▶ Additions on CPU

This optimization is suitable much larger matrixes than the previous, more significantly when n exceeds 2000 entries.

Optimizations Review

Naive Implementation 0.019GFLOPS

- ▶ Inner Loop Unrolling 0.035 GFLOPS
- ▶ Pipeline Inner Loop 0.044 GFLOPS
- ▶ Unrol Loop J 0.188 GFLOPS
- ▶ Row/Col Partitioning 0.33 GFLOPS
- ▶ Maximize DSPs 1.247 GFLOPS
- ▶ Maximize Resource Usage 2.886 GFLOPS
- ▶ Improve I/O 3.573 GFLOPS
- ▶ Block Computation 4.107 GFLOPS
- ▶ Optimize Communication 4.695 GFLOPS

Thank You!