FPGAs, HLS Tools & Runtime Systems

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

Georgios Christodoulis

CORSE–LIG

gchristodoulis@gmail.com

Overview

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

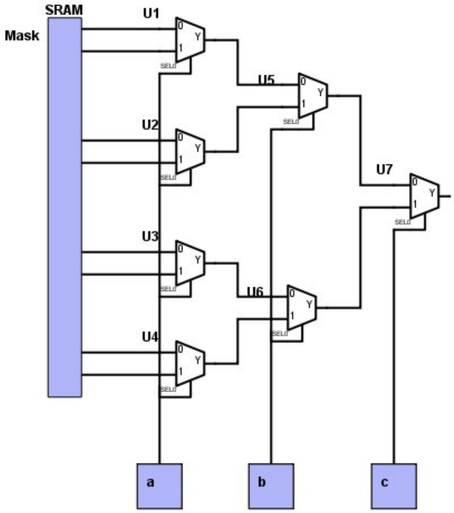


Figure: 3 stages of 2x1 MUX

- It is a table that ditermines 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

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

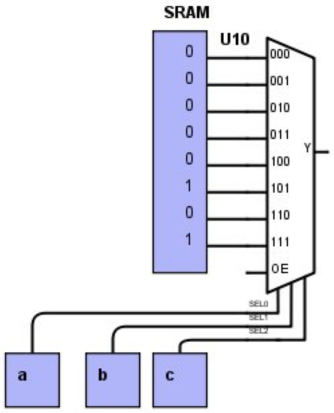


Figure: $y = (a + b) \cdot c$

FPGAs structure

BLE

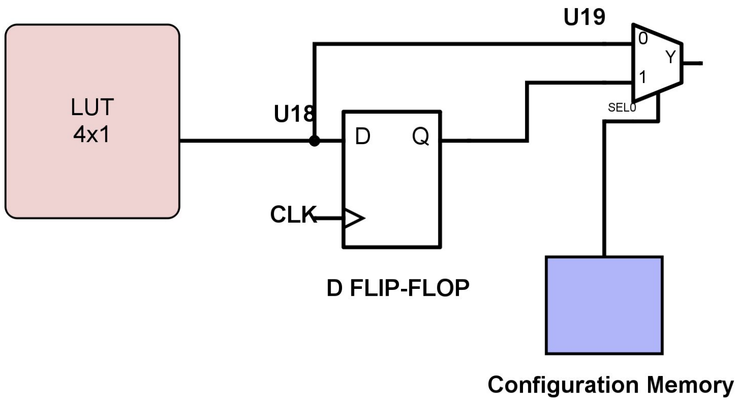


Figure: Basic Logic Element

FPGAs structure
Overview

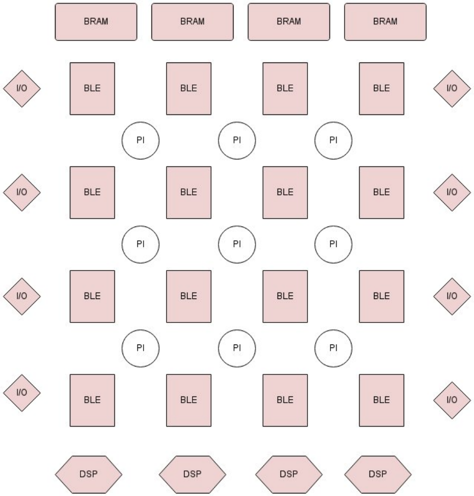


Figure: FPGAs Complete Overview

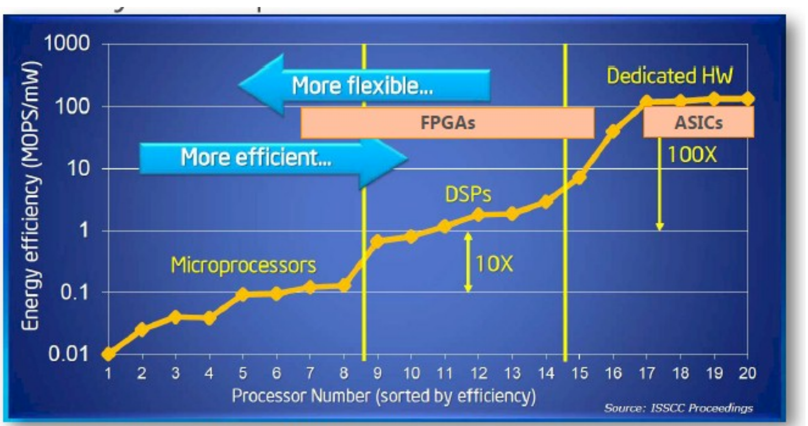
De-Motivation

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

 ENTITY lab4b_tb IS
 END lab4b_tb;

ARCHITECTURE behavior OF lab4b_tb IS

   signal Clk:
std_logic     :=  '0';
—— no reset
   signal Start:
std_logic     :=  '0';
—— no reset
   signal Din:
INTEGER       :=  0;
—— no reset

   signal Done : std_logic;
   signal Dout : INTEGER;

   constant Clk_period : time := 10 ns;

BEGIN
```

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

entity DCT_beh is
    port (
        Clk :
in std_logic;
        Start :
in std_logic;
        Din :
in INTEGER;
        Done :
out std_logic;
        Dout :
out INTEGER
      );

 end DCT_beh;

architecture behavioral of DCT_beh is
    type RF is array ( 0 to 7, 0 to 7 ) of INTEGER;
    signal OutBlock:
RF;
    signal InBlock:
RF;
    signal internal_Done:
std_logic := '0';
— no reset
    signal Input_Ready:
std_logic := '0';
— no reset
    signal done_detected:
std_logic := '0';
— no reset
    signal input_rdy_detected:
std_logic := '0';
— no reset
    signal last_out:
std_logic := '0';
— no reset

begin
INPUT_DATA:
    process
    begin
        wait until Start = '1';
        —Read Input Data
        for i in 0 to 7 loop
            for j in 0 to 7 loop
                wait until Clk = '1' and clk'event;
                InBlock(i,j) <= Din;
                if i=7 and j=7 then
                    Input_Ready <= '1', '0' after 11 ns;
                end if;
            end loop;
        end loop;
    end process;

WAIT_FOR_InBlock:
    process
    begin
        wait until clk = '1' and clk'event;
```

Motivation

Energy Efficiency



Source: Bob Broderson, Berkeley Wireless group

More Motivation

Acceleration

Application	CPU only	FPGA co-processing	Factor
Hough Processing	12 mins Pentium 4 @3Ghz	2sec 20Mhz	$\times$370
Spacial Statistics	3,397 hours Pentium 4 @2.8Ghz	36 hours	$\times$96
Black-Scholes (Financial)	2.3 Mexperiments/sec Pentium 4 @2.8Ghz	299 Mexperiments/sec	$\times$130
Smith Waterman SS	6461 CPU sec Opteron	100 sec	$\times$64
Prewitt Edge Detection	327 MCC 1CPU @1GHz	131 KCC FPGA @0.33MHz	$\times$83
Monte Carlo Radiative Heat Transfer	60ns CPU CPU @3GHz	6.12ns	$\times$10
BJM (5M paths) (Financial)	6300 sec Pentium 4 @1.5GHz	242 sec 61MHz	$\times$26

Problem Description

Matrix Multiplication

$$C = A * B$$

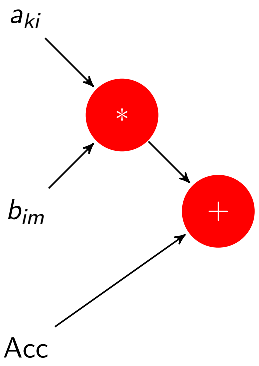
$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

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

No Directives

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

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



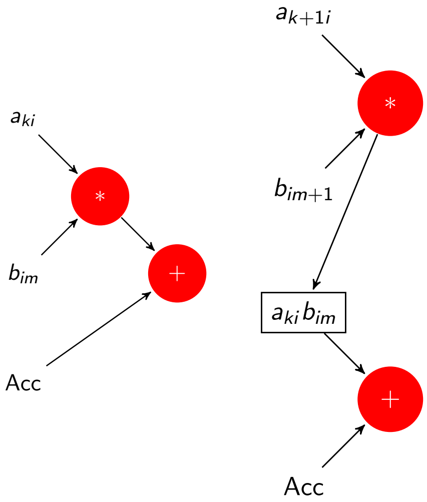
- $2n$ operations $\forall$ element
- n^2 elements

$\Rightarrow$ $2n^3$ operations

Inner Loop Unrolling

Opt1: Sum Mul Overlaping

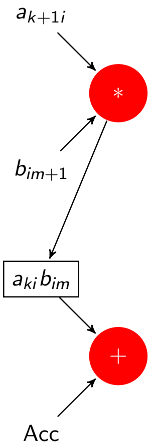
Our reference time interval is defined by the slowest operation which is the multiplication.

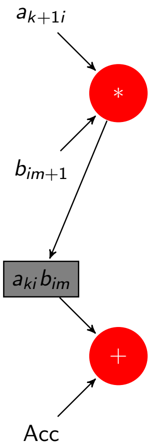


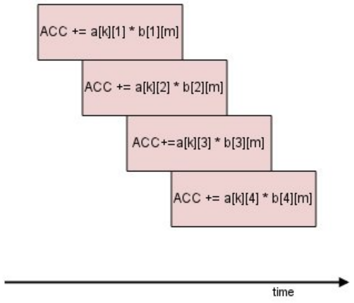
- Complete overlaping between Multiplication and addition

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.





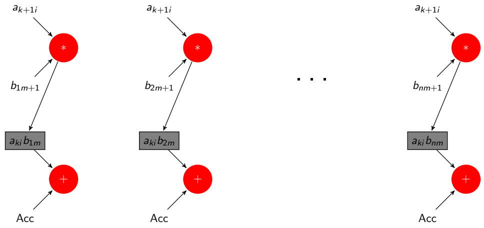


J-loop unrolling

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

$$\begin{array}{cccc} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nn} \end{array}$$

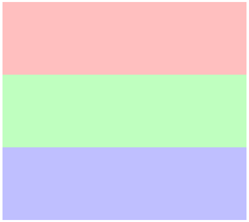
- Extra Hardware
- Real Parallelism (Superscalar CPU architectures)
- Expected scaling: $\times n$
- Actual scaling: $\times 2$



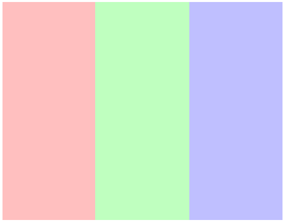
Memmory Bounds: Dual Channeled memory $\implies$ only 2 concurent operations.

Row / Col Partitioning

A



B



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

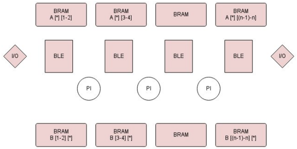


Figure: Distribute the array into multiple BRAMS

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

$$\begin{tikzcd}
\boxed{\text{FPGA}} \arrow[r] & \text{32bits} \arrow[r] & \boxed{\text{CPU}} \\
\text{100Mhz} \arrow[u]
\end{tikzcd}$$

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

$$\begin{tikzcd}
\boxed{\text{FPGA}} \arrow[r] & \text{128bits} \arrow[r] & \boxed{\text{AXI DMA}} \arrow[r] & \text{64bits} \arrow[r] & \boxed{\text{CPU}} \\
\text{100Mhz} \arrow[u]
\end{tikzcd}$$

- $100Mhz \times 128\text{bits} = 1.6\text{GB}/s$
- AXI DMAs limit is 1.2GB/s at 64bits channel width

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

Block Computation

$$c_{ij} = \underset{\substack{\uparrow \\ \text{CPU}}}{\boxed{\sum_{k=0}^{m-1}}} \quad \underset{\substack{\uparrow \\ \text{FPGA}}}{\boxed{A_{ik}B_{kj}}}$$

- Multiplications on FPGA
- Addictions 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!