

# Hardware platforms and architectures for public-key cryptography on constrained devices

Nele Mentens  
nele.mentens@kuleuven.be

Kayaks & Dreadnoughts in a Sea of Crypto

September 15-16, 2016, Brussels

## Outline

- Introduction
- Datapath optimization
- Data storage optimization
- Control unit optimization
- Summary

- Introduction
  - Implementation platforms
  - ECC algorithms
  - ECC implementation
- Datapath optimization
- Data storage optimization
- Control unit optimization
- Summary

Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

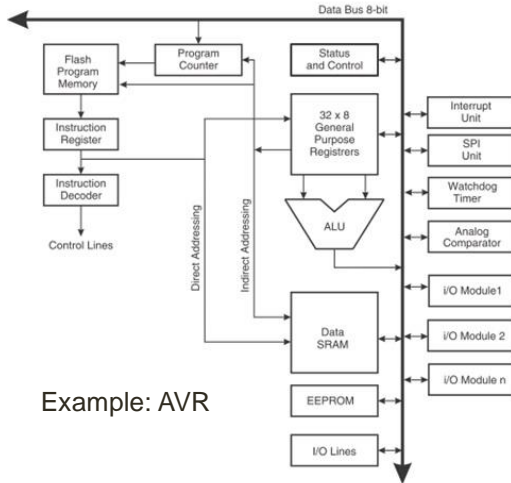
## Introduction Implementation platforms

- microprocessor
- FPGA = Field-Programmable Gate Array
- ASIC = Application-Specific Integrated Circuit

# Introduction

## Implementation platforms – microprocessor

### architecture



Example: AVR

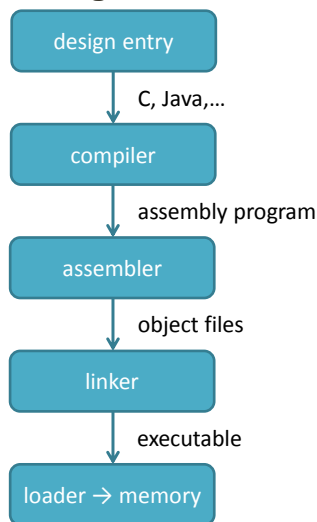
The CPU is the heart of a microprocessor and contains a.o.:

- ALU (Arithmetic Logic Unit)
- register file
- program memory

# Introduction

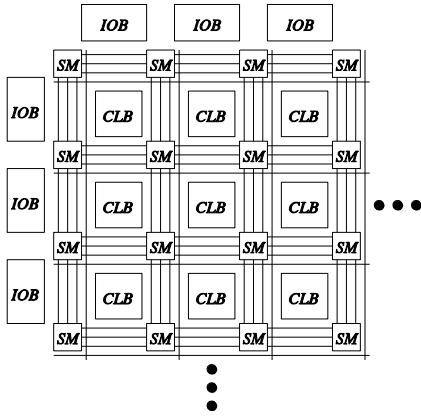
## Implementation platforms – microprocessor

### design flow



- The hardware architecture of a microprocessor is fixed
- The code describes what should be executed on the fixed hardware
- The instructions end up in the program memory

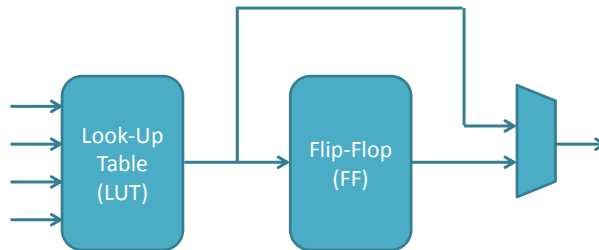
**architecture**



**Basic components:**

- CLB = Configurable Logic Block
  - CLBs consist of slices.
  - Slices consist of
    - Look-Up Tables (LUTs),
    - Multiplexers,
    - Flip-Flops (FFs),
    - Carry logic.
- SM = Switch Matrix
- IOB = Input/Output Block

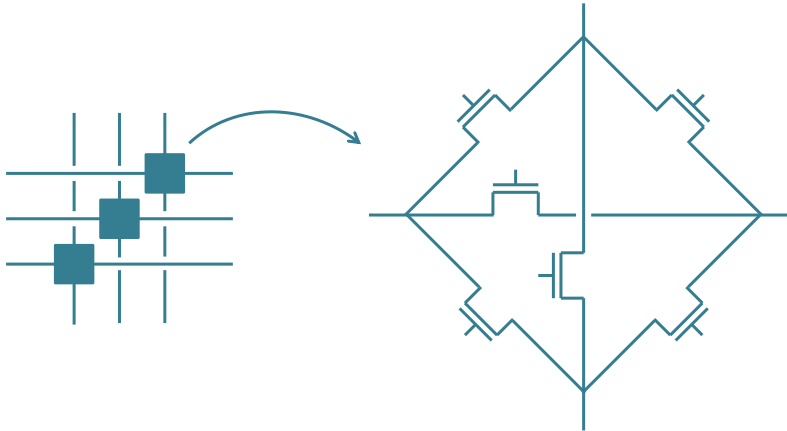
**basic content of a slice (excluding carry-logic)**



# Introduction

## Implementation platforms – FPGA

basic principle of a switch matrix



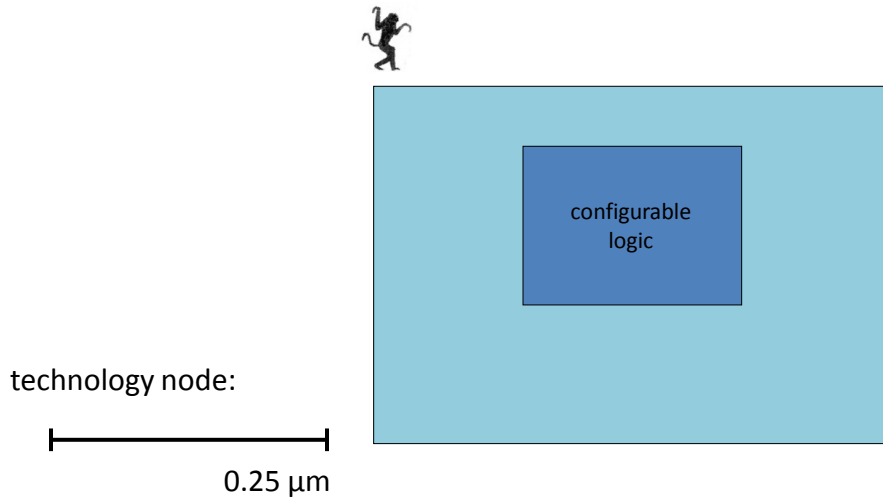
Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

6/45

# Introduction

## Implementation platforms – FPGA

1991: XC4000



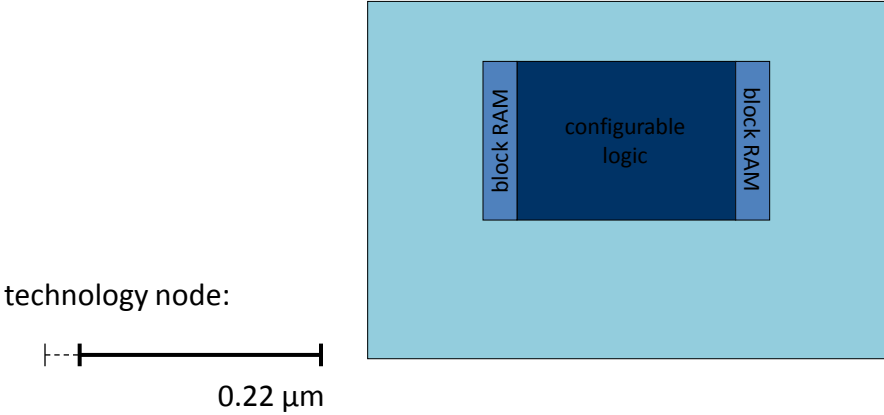
Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

7/45

# Introduction

## Implementation platforms – FPGA

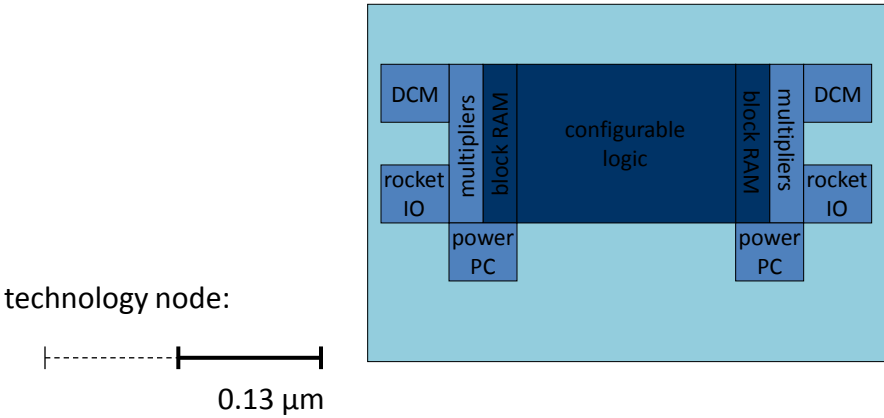
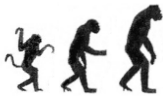
1991: XC4000  
 1998: Virtex



# Introduction

## Implementation platforms – FPGA

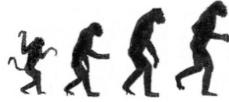
1991: XC4000  
 1998: Virtex  
 2002: Virtex-II Pro



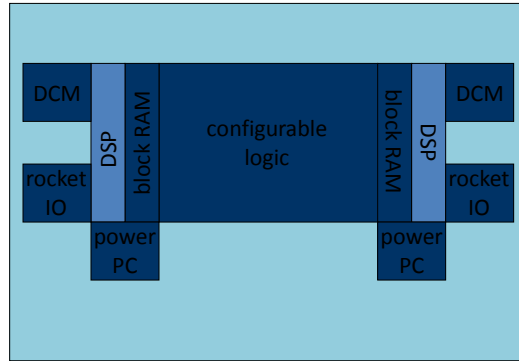
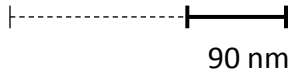
# Introduction

## Implementation platforms – FPGA

- 1991: XC4000
- 1998: Virtex
- 2002: Virtex-II Pro
- 2004: Virtex-4



technology node:



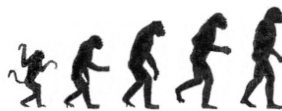
Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

7/45

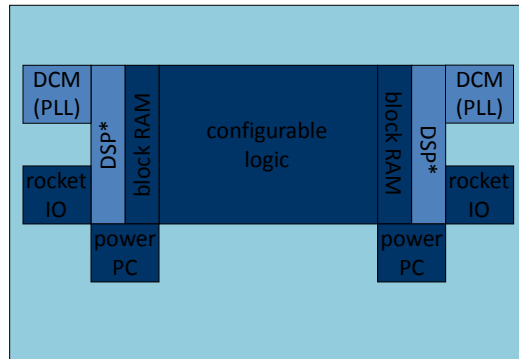
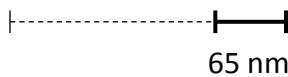
# Introduction

## Implementation platforms – FPGA

- 1991: XC4000
- 1998: Virtex
- 2002: Virtex-II Pro
- 2004: Virtex-4
- 2006: Virtex-5



technology node:



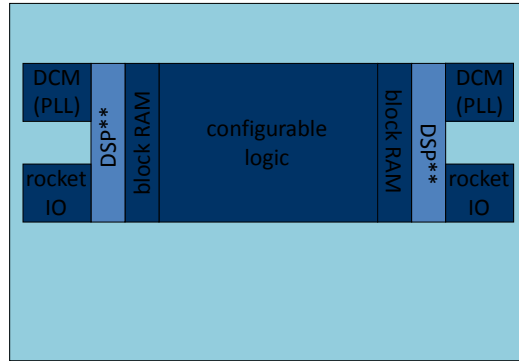
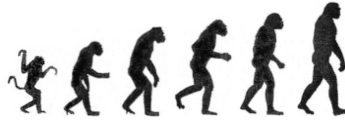
Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

7/45

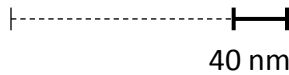
# Introduction

## Implementation platforms – FPGA

- 1991: XC4000
- 1998: Virtex
- 2002: Virtex-II Pro
- 2004: Virtex-4
- 2006: Virtex-5
- 2009: Virtex-6



technology node:



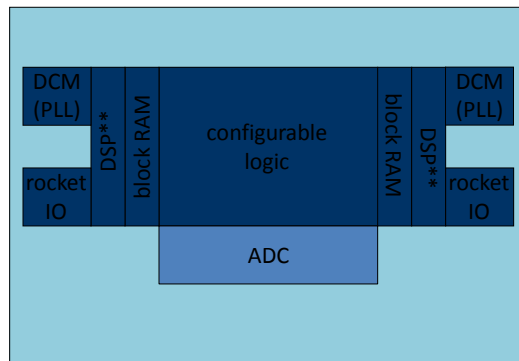
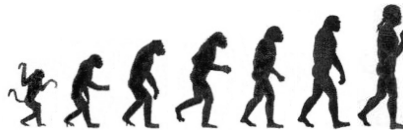
Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

7/45

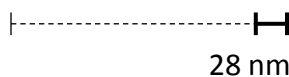
# Introduction

## Implementation platforms – FPGA

- 1991: XC4000
- 1998: Virtex
- 2002: Virtex-II Pro
- 2004: Virtex-4
- 2006: Virtex-5
- 2009: Virtex-6
- 2010: Virtex-7



technology node:



Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

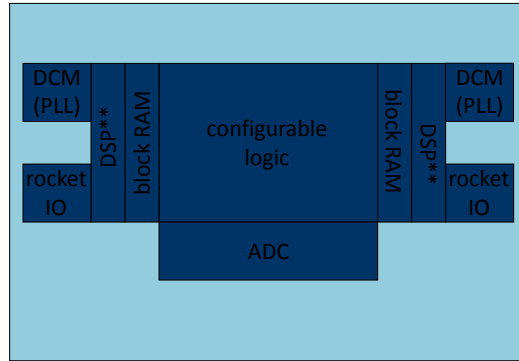
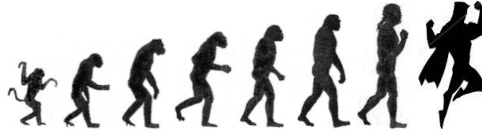
7/45



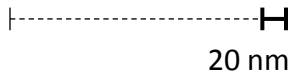
# Introduction

## Implementation platforms – FPGA

- 1991: XC4000
- 1998: Virtex
- 2002: Virtex-II Pro
- 2004: Virtex-4
- 2006: Virtex-5
- 2009: Virtex-6
- 2010: Virtex-7
- 2013: Virtex UltraScale



technology node:



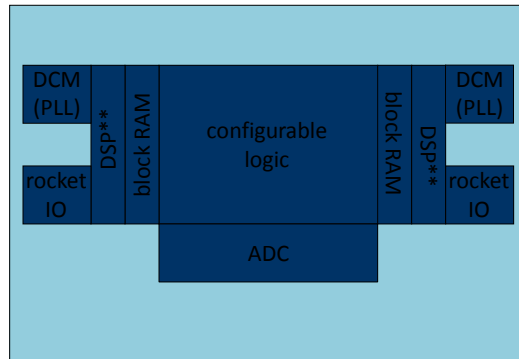
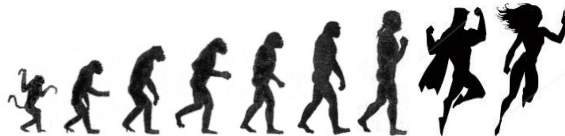
Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

7/45

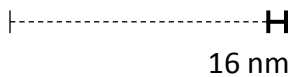
# Introduction

## Implementation platforms – FPGA

- 1991: XC4000
- 1998: Virtex
- 2002: Virtex-II Pro
- 2004: Virtex-4
- 2006: Virtex-5
- 2009: Virtex-6
- 2010: Virtex-7
- 2013: Virtex UltraScale
- 2015: Virtex UltraScale+



technology node:



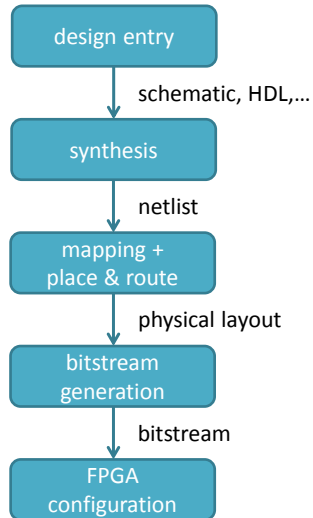
Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

7/45

# Introduction

## Implementation platforms – FPGA

### design flow



- The hardware architecture of an FPGA is configurable
- The code describes the hardware that we need
- The bitstream ends up in the configuration memory
- The area is measured in terms of occupied LUTs, flip-flops, dedicated hardware blocks

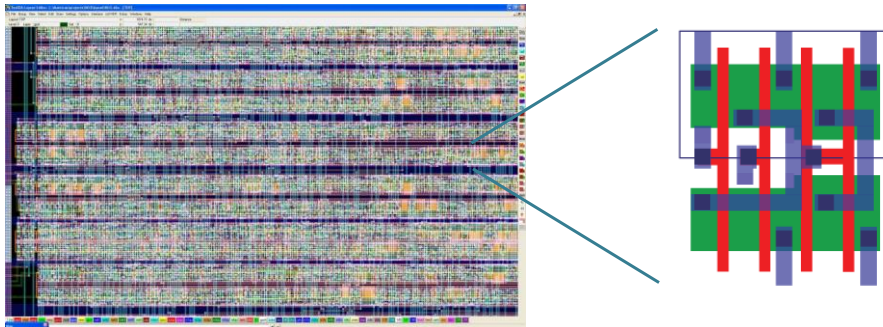
Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

8/45

# Introduction

## Implementation platforms – ASIC

### architecture



### Basic components:

- Standard cells from a standard cell library
  - Logic cells and sequential cells

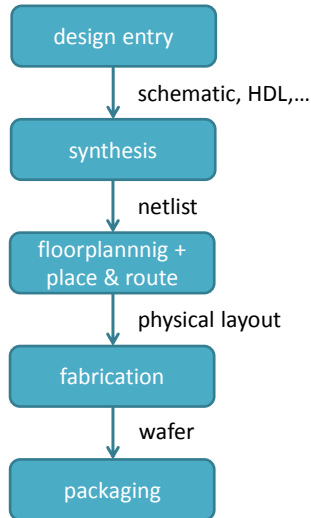
Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

9/45

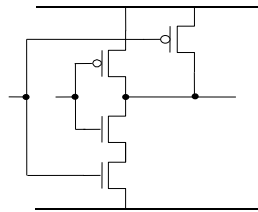
# Introduction

## Implementation platforms – ASIC

### design flow



- The hardware architecture of an ASIC is fixed
- The code describes the hardware that we need
- The GDSII file contains the physical information that goes to the foundry
- The area is measured in terms of the number of equivalent NAND gates (Gate Equivalent = GE)



Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

10/45

# Introduction

## Implementation platforms – comparison

HW		HW-SW			SW
ASIC	FPGA	Domain specific	DSP	VLIW	General purpose
Area efficiency					
High					Low
Performance/Energy unit					
Low					High
Programmability					

Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

11/45

# Introduction

## Implementation platforms – comparison

- FPGA advantages over ASIC
  - faster time-to-market
  - smaller Non-Recurring Engineering (NRE) cost
  - programmable in the field
- ASIC advantages over FPGA
  - better performance
  - lower cost for high volumes

# Introduction

## ECC algorithms

- Elliptic Curve Cryptography (ECC) is the preferred approach for public-key cryptography on constrained devices
  - Smaller key lengths than other public-key algorithms
- Typical algorithm for point multiplication in ECC:
  - Evaluate one or more (original or recoded) bits of the scalar
  - Execute specific sequences of modular operations
  - Update intermediate variables

# Introduction

## ECC algorithms – example

- Left-to-right binary point multiplication

in:  $P \in G$ , positive integer  $n = (n_{l-1} \dots n_0)$ ,  $n_{l-1} = 1$   
 out:  $[n]P \in G$

$R \leftarrow P$   
 for  $i = l-2$  to  $0$  do  
      $R \leftarrow [2]R$   
     if  $n_i = 1$  then  $R \leftarrow R+P$   
      $i \leftarrow i-1$   
 Return  $R$

# Introduction

## ECC algorithms – example

Short Weierstrass curve:  $y^2 = x^3 + a*x + b$

Modified Jacobian coordinates:  $x = X/Z_2$ ;  $y = Y/Z_3$ ;  $T = a*Z_4$

$(X_3, Y_3, Z_3, T_3) = (X_1, Y_1, Z_1, T_1) + (X_2, Y_2, Z_2, T_2)$

$ZZ_1 = Z_1^2$

$ZZ_2 = Z_2^2$

$U_1 = X_1 * ZZ_2$

$U_2 = X_2 * ZZ_1$

$S_1 = Y_1 * Z_2 * ZZ_2$

$S_2 = Y_2 * Z_1 * ZZ_1$

$H = U_2 - U_1$

$I = (2 * H)^2$

$J = H * I$

$r = 2 * (S_2 - S_1)$

$V = U_1 * I$

$X^3 = r^2 - J - 2 * V$

$Y^3 = r * (V - X^3) - 2 * S_1 * J$

$Z_3 = ((Z_1 + Z_2)^2 - ZZ_1 - ZZ_2) * H$

$ZZ_3 = Z_3^2$

$T_3 = a * ZZ_3^2$

**P3 ← P1+P2**

**point addition**

$(X_3, Y_3, Z_3, T_3) = [2](X_1, Y_1, Z_1, T_1)$

$XX = X_1^2$

$A = 2 * Y_1^2$

$AA = A^2$

$U = 2 * AA$

$S = (X_1 + A)^2 - XX - AA$

$M = 3 * XX + T_1$

$X_3 = M^2 - 2 * S$

$Y_3 = M * (S - X_3) - U$

$Z_3 = 2 * Y_1 * Z_1$

$T_3 = 2 * U * T_1$

**P3 ← [2]P1**

**point doubling**

## Introduction

### ECC implementation – microprocessor

- Typical algorithm for point multiplication in Elliptic Curve Cryptography (ECC):
  - Evaluate one or more (original or recoded) bits of the scalar → **LSL, BREQ**
  - Execute specific sequences of modular operations → **ADD, ADC, SUB, SBC, MUL**
  - Update intermediate variables

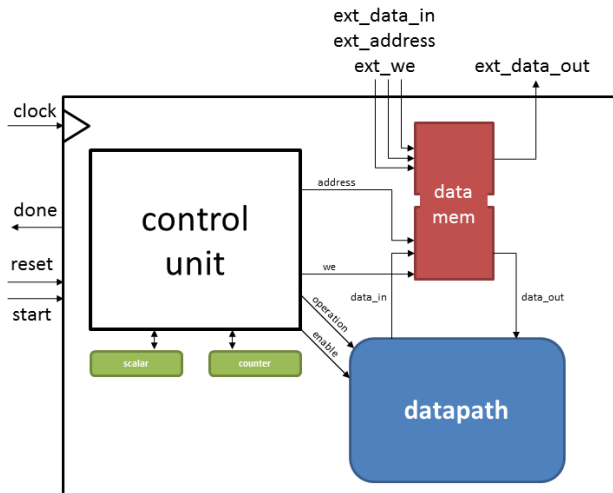
## Introduction

### ECC implementation – hardware architecture

- Typical algorithm for point multiplication in Elliptic Curve Cryptography (ECC):
  - Evaluate one or more (original or recoded) bits of the scalar → **control unit + shift register + counter**
  - Execute specific sequences of modular operations → **control unit + datapath + data storage**
  - Update intermediate variables → **data storage**

# Introduction

## ECC implementation – hardware architecture

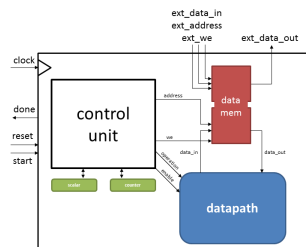


Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

18/45

## Outline

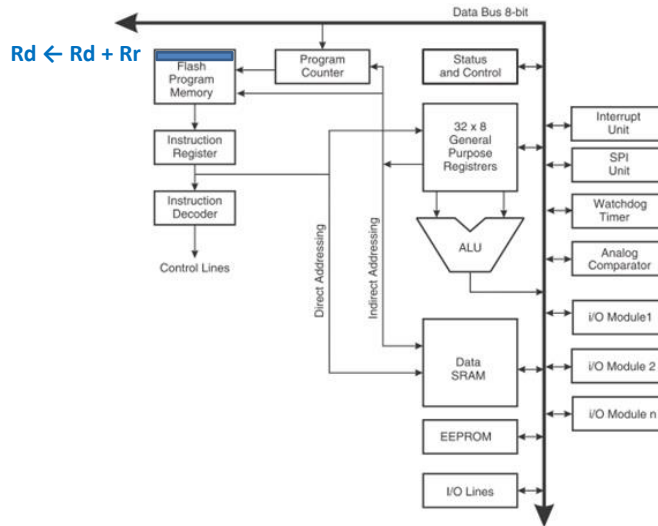
- Introduction
- Datapath optimization
  - 256-bit addition
  - 256-bit multiplication
- Data storage optimization
- Control unit optimization
- Summary



Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

# Datapath optimization

## 256-bit addition – microprocessor

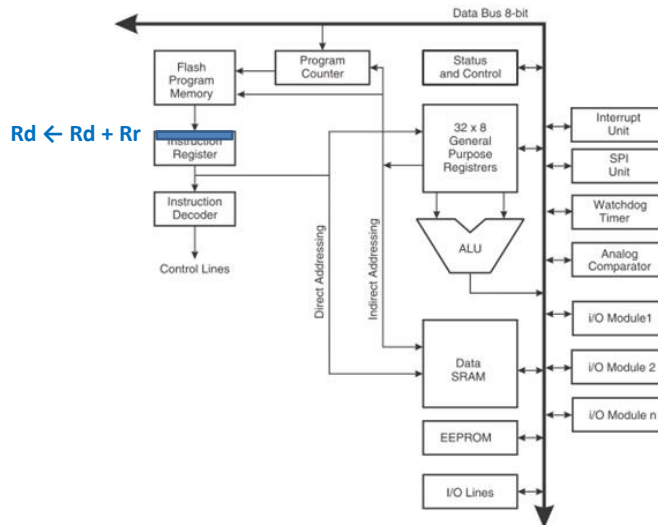


Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

19/45

# Datapath optimization

## 256-bit addition – microprocessor



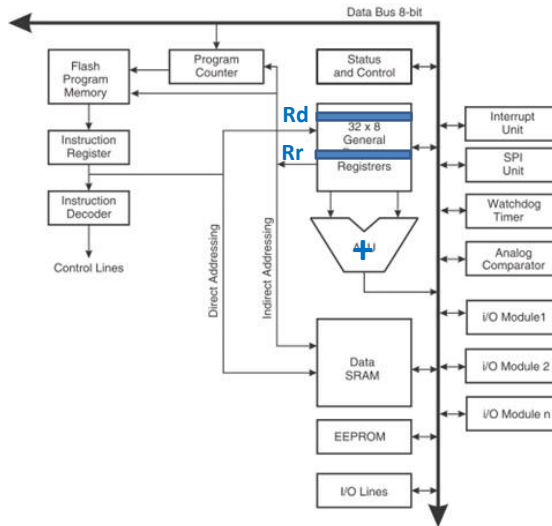
Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

19/45



# Datapath optimization

## 256-bit addition – microprocessor

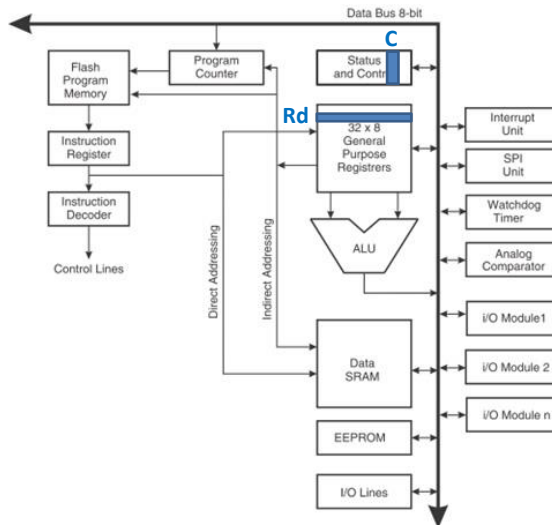


Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

19/45

# Datapath optimization

## 256-bit addition – microprocessor

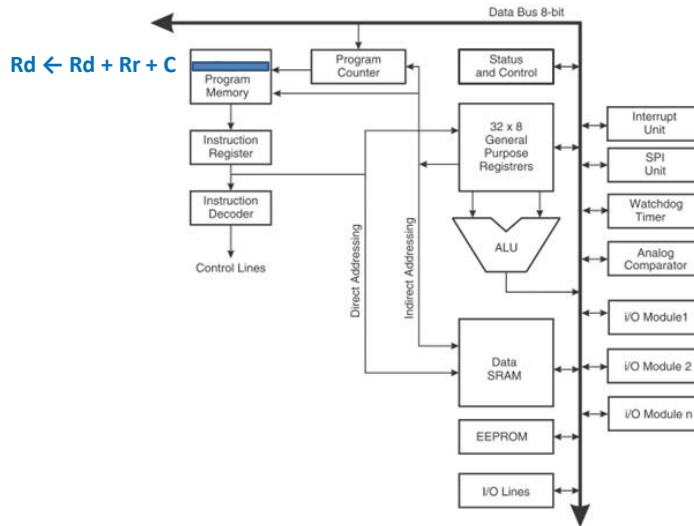


Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

19/45

# Datapath optimization

## 256-bit addition – microprocessor

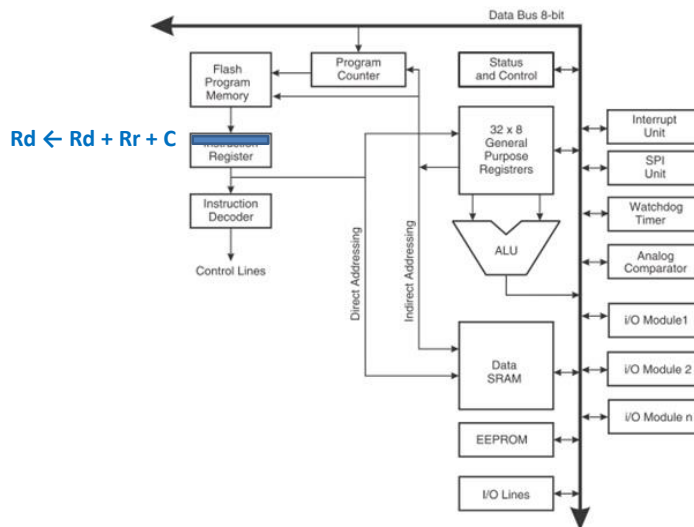


Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

19/45

# Datapath optimization

## 256-bit addition – microprocessor

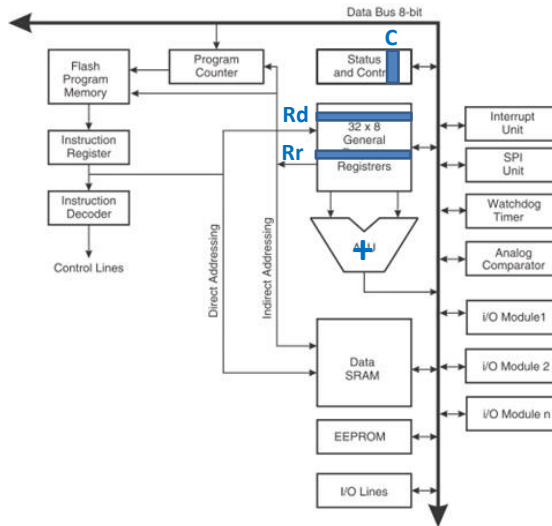


Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

19/45

# Datapath optimization

## 256-bit addition – microprocessor

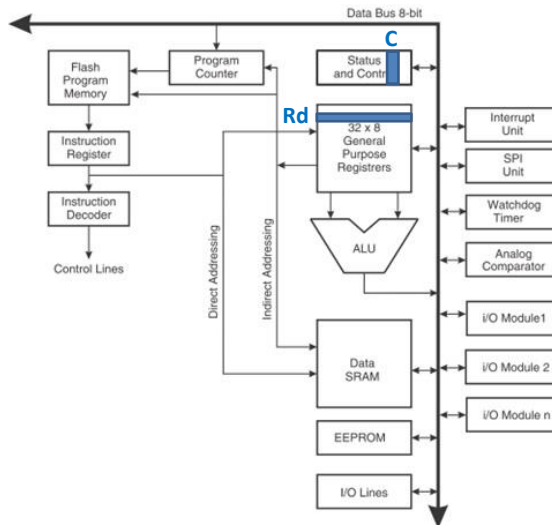


Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

19/45

# Datapath optimization

## 256-bit addition – microprocessor

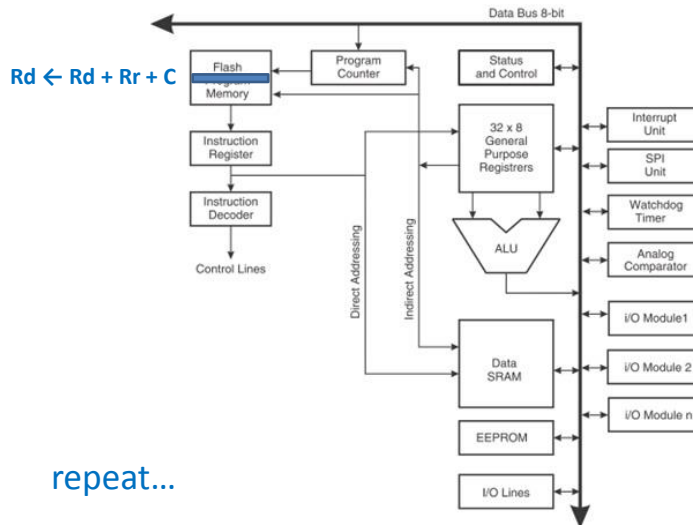


Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

19/45

# Datapath optimization

## 256-bit addition – microprocessor



Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

19/45

# Datapath optimization

## 256-bit addition – hardware architecture

- most straightforward: ripple-carry adder
- parallel ... digit-serial ... bit-serial

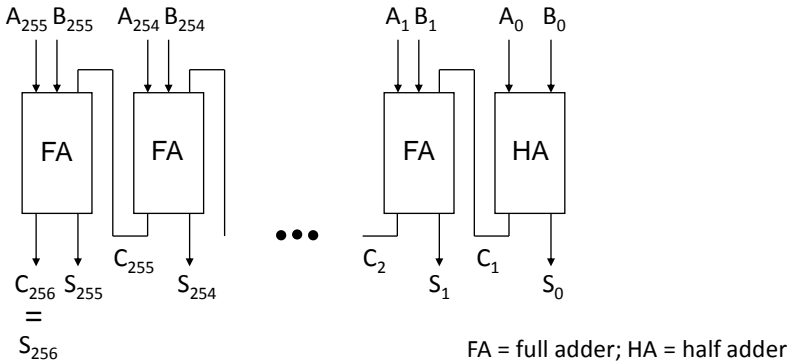
Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

20/45

# Datapath optimization

## 256-bit addition – hardware architecture

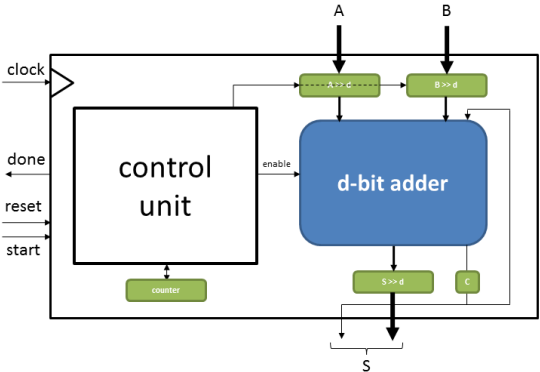
- most straightforward: ripple-carry adder
- **parallel ... digit-serial ... bit-serial**



# Datapath optimization

## 256-bit addition – hardware architecture

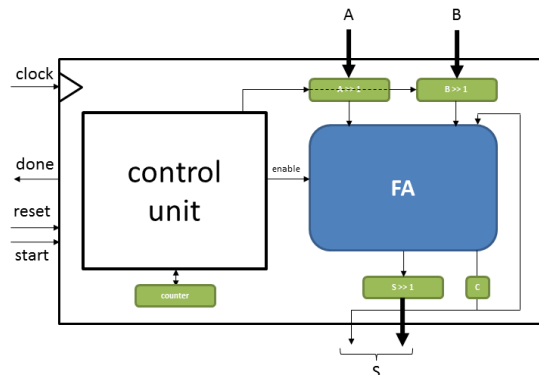
- most straightforward: ripple-carry adder
- parallel ... **digit-serial ... bit-serial**



## Datapath optimization

### 256-bit addition – hardware architecture

- most straightforward: ripple-carry adder
- parallel ... digit-serial ... **bit-serial**



Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

23/45

## Datapath optimization

### 256-bit addition – hardware architecture

- most straightforward: ripple-carry adder
- parallel ... **digit-serial** ... **bit-serial**
  - less logic for adding digits/bits
  - overhead in control logic, counter, input and output shift registers, carry register

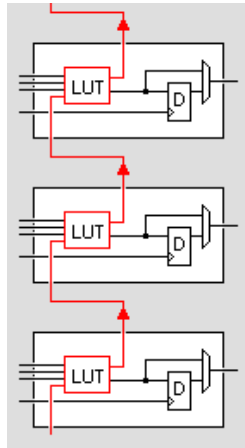
Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

24/45

## Datapath optimization

### 256-bit addition – hardware architecture

#### FPGA: fast carry-chains for ripple-carry addition



(just write  $s \leq a + b$  in VHDL)

(source: <http://www.fpga4fun.com>)

Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

25/45

## Datapath optimization

### 256-bit addition – hardware architecture

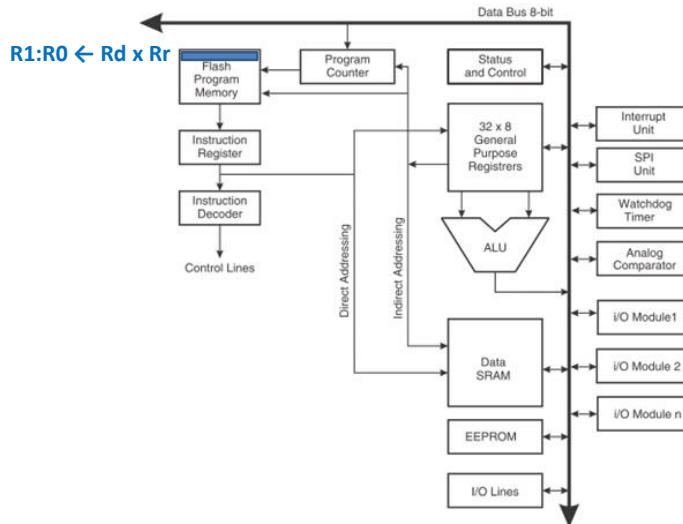
- ASIC:
  - ripple-carry adder using HA and FA standard cells
  - faster architectures:
    - carry look-ahead, carry-select, conditional sum,...

Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

26/45

# Datapath optimization

## 256-bit multiplication – microprocessor

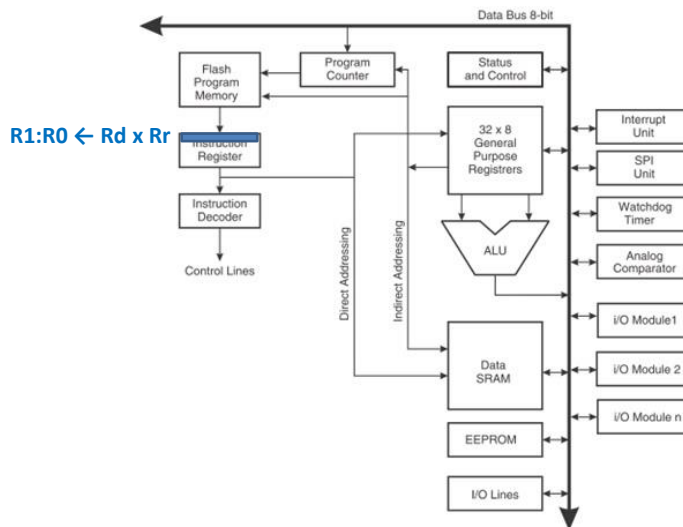


Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

27/45

# Datapath optimization

## 256-bit multiplication – microprocessor



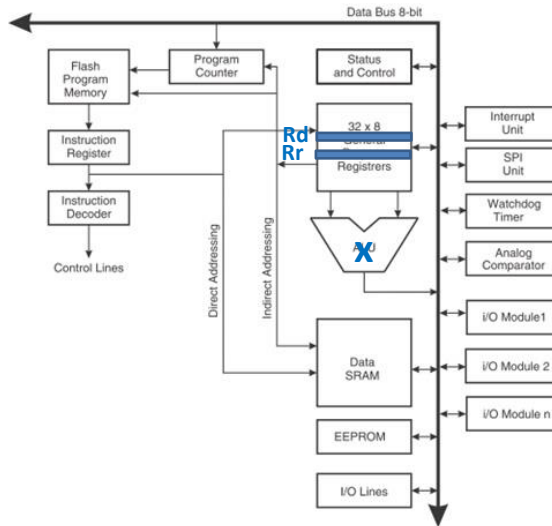
Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

27/45



# Datapath optimization

## 256-bit multiplication – microprocessor

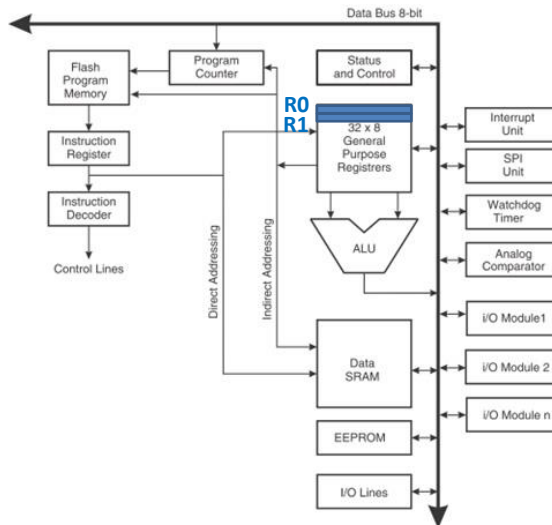


Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

27/45

# Datapath optimization

## 256-bit multiplication – microprocessor

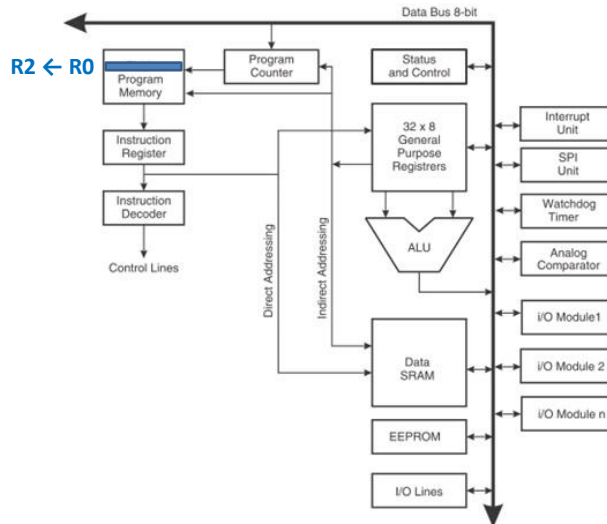


Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

27/45

# Datapath optimization

## 256-bit multiplication – microprocessor

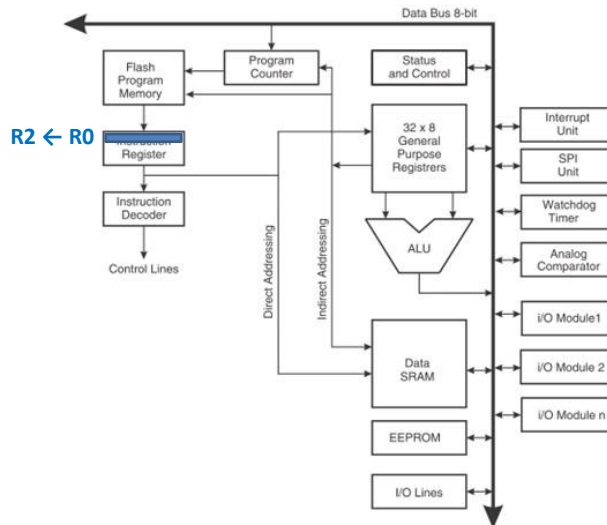


Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

27/45

# Datapath optimization

## 256-bit multiplication – microprocessor

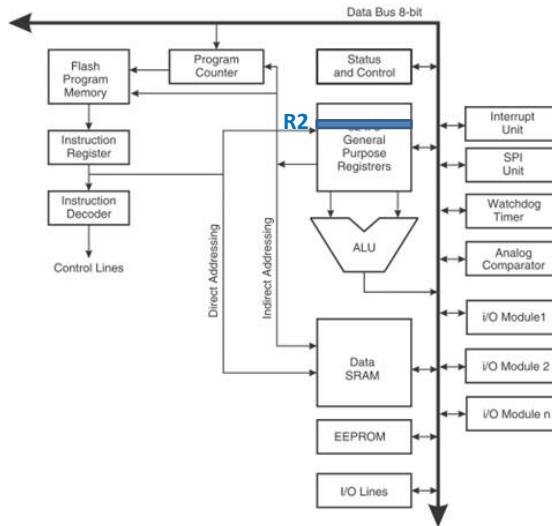


Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

27/45

# Datapath optimization

## 256-bit multiplication – microprocessor

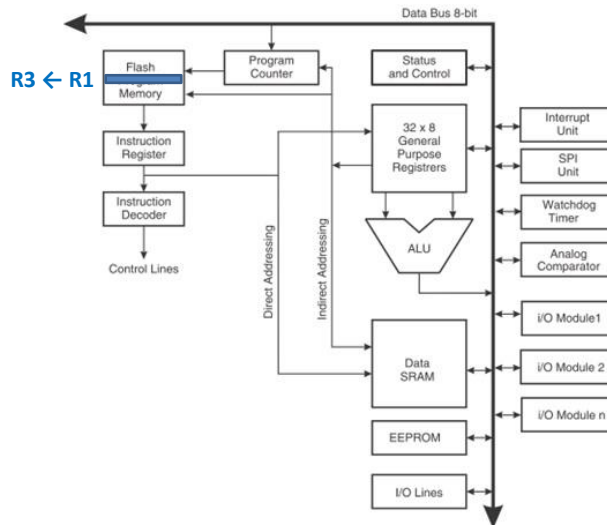


Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

27/45

# Datapath optimization

## 256-bit multiplication – microprocessor

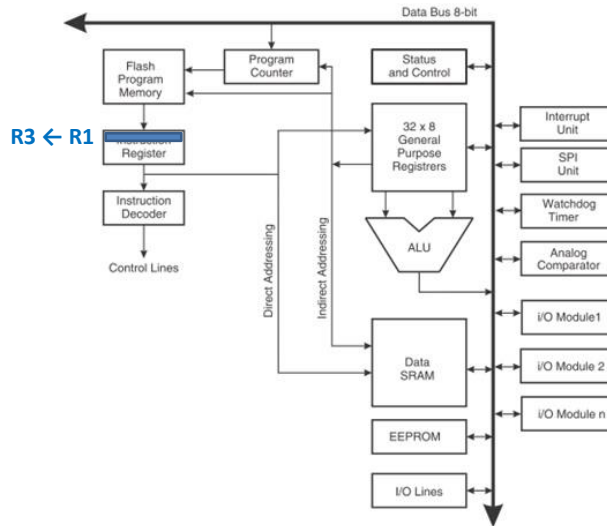


Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

27/45

# Datapath optimization

## 256-bit multiplication – microprocessor

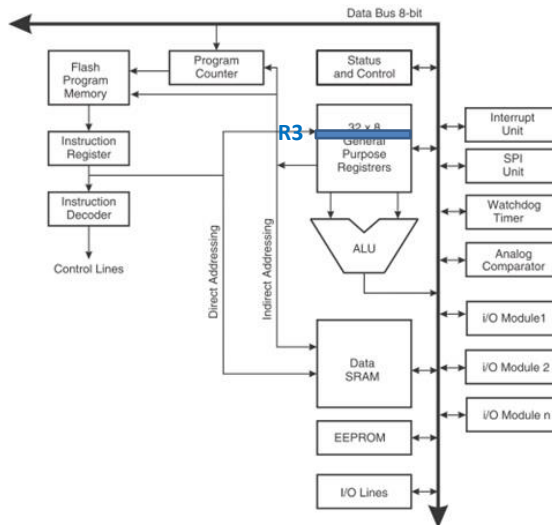


Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

27/45

# Datapath optimization

## 256-bit multiplication – microprocessor

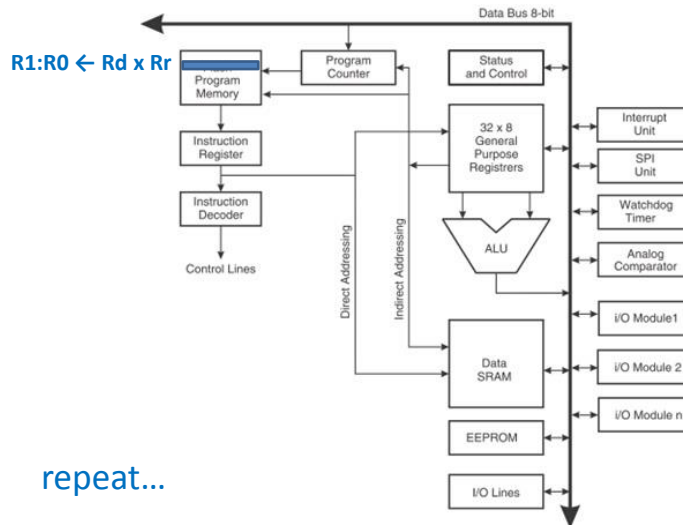


Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

27/45

# Datapath optimization

## 256-bit multiplication – microprocessor



Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

27/45

# Datapath optimization

## 256-bit multiplication – microprocessor

- Optimization:
  - rely on the compiler
  - do low-level programming (assembly)

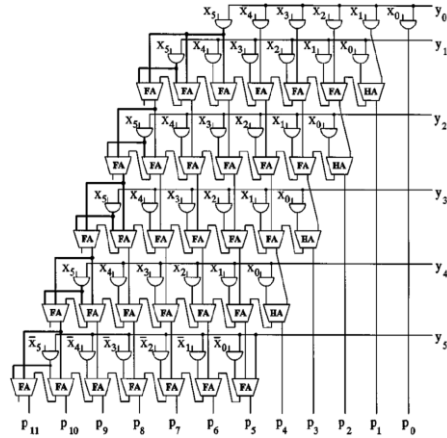
Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

28/45

# Datapath optimization

## 256-bit multiplication – hardware architecture

most straightforward: carry-save multiplier



Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

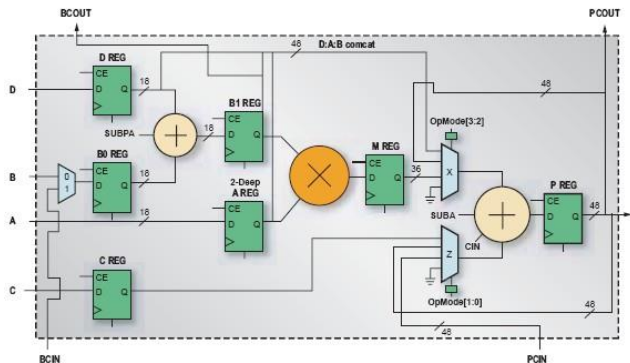
29/45

# Datapath optimization

## 256-bit multiplication – hardware architecture

FPGA: dedicated multipliers in the DSP slices

(just write  $p \leftarrow a * b$  in VHDL)



(source: <http://www.eetimes.com>)

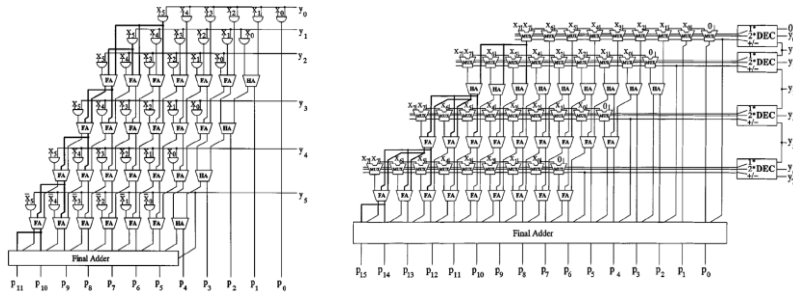
Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

30/45

# Datapath optimization

## 256-bit multiplication – hardware architecture

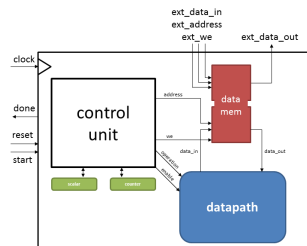
- ASIC
  - standard cells for the design of multipliers
  - faster architectures:
    - carry-save multiplier, Booth multiplier,...



Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

# Outline

- Introduction
- Datapath optimization
- Data storage optimization
  - Microprocessor
  - Memory in hardware
- Control unit optimization
- Summary

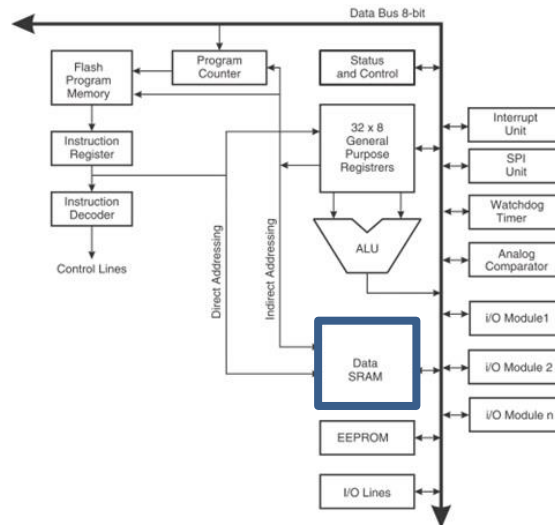


Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

# Data storage optimization

## Microprocessor

- Comply to the data memory available on the processor



Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

32/45

# Data storage optimization

## Memory in hardware

- Flip-flop
  - bit storage on a clock edge
- Embedded SRAM
  - matrix structure
  - row-based read and write

Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

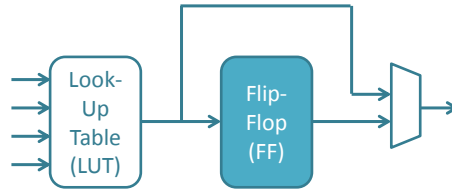
33/45



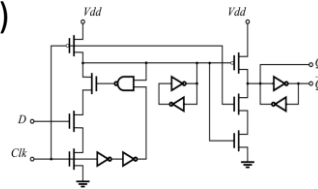
# Data storage optimization

## Memory in hardware – flip-flops

- Fixed area per bit:
  - FPGA: basic element



- ASIC: standard cell (5.5 – 12 GE)
  - Example: UltraSparc flip-flop



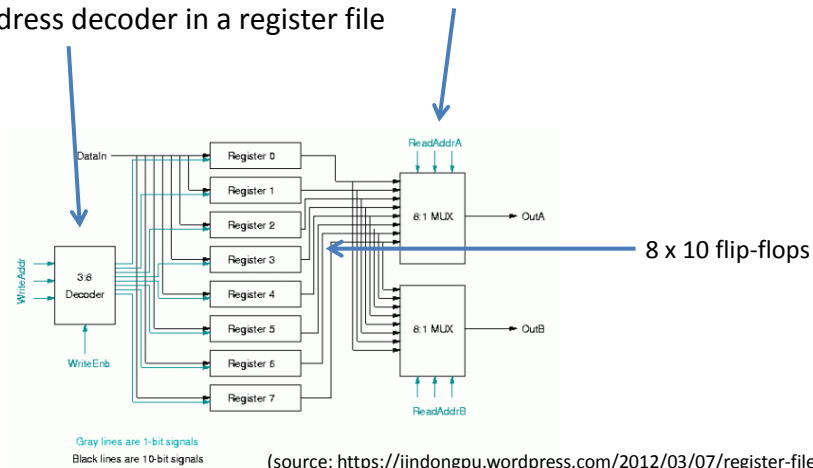
Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

34/45

# Data storage optimization

## Memory in hardware – flip-flops

- Register file: overhead for the output selection and the address decoder in a register file



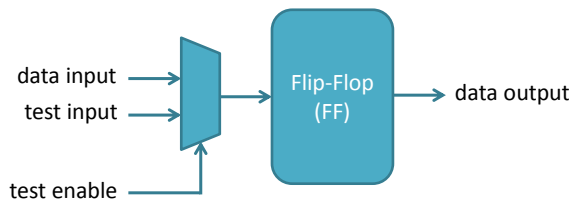
Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

35/45

# Data storage optimization

## Memory in hardware – flip-flops

- Optimizations:
  - Scan flip-flop: standard cell, designed for scan-chain testing, combining a flip-flop and a multiplexer
    - Can also be used in normal operation mode in ASICs
    - Flip-flop + multiplexer: 8 GE (open source NanGate library)
    - Scan flip-flop: 7.667 GE (open source NanGate library)



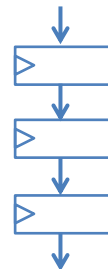
Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

36/45

# Data storage optimization

## Memory in hardware – flip-flops

- Optimizations:
  - Shift register: to avoid the address decoder
    - For both FPGA and ASIC, but not always possible
    - FPGAs have dedicated, optimized shift registers with parameterizable output selection



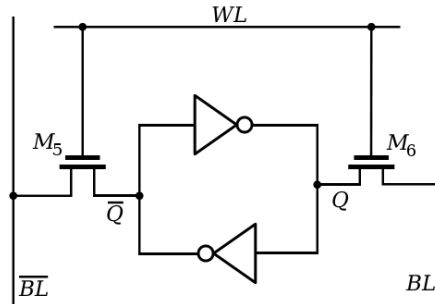
Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

37/45

# Data storage optimization

## Memory in hardware – embedded RAM

- Fixed area per bit: ~1.5 GE



(source: <https://commons.wikimedia.org>)

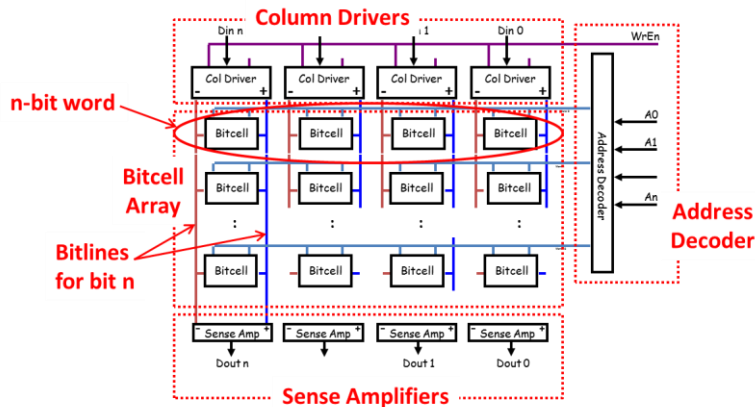
Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

38/45

# Data storage optimization

## Memory in hardware – embedded RAM

- Overhead for the address decoder
- Additional overhead for drivers and sense amplifiers per column



Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

39/45

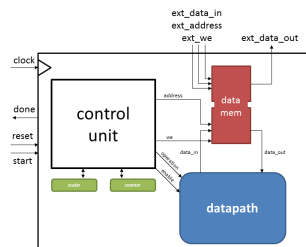
# Data storage optimization

## Memory in hardware – embedded RAM

- FPGA: embedded RAM blocks (BRAM = Block RAM)
  - Example Xilinx Spartan-6:
    - Configurable 8 Kb blocks:
      - 8k x 1, 4k x 2, 2k x 4, 1k x 8, 512 x 16, 256 x 32
      - Single-port, dual-port
- ASIC: embedded RAM using memory compilers
  - Rule of thumb: use flip-flops for memories smaller than 8x8

## Outline

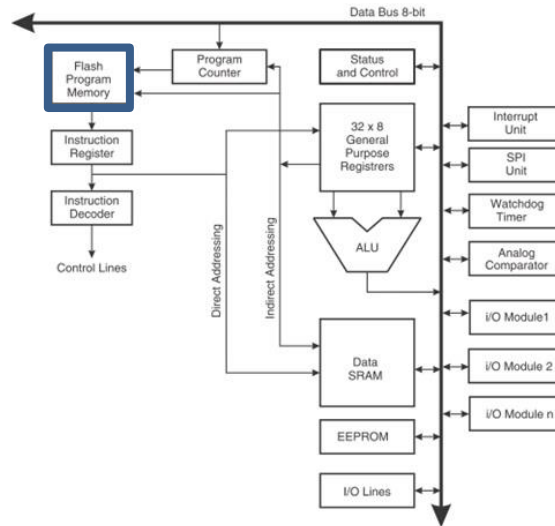
- Introduction
- Datapath optimization
- Data storage optimization
- Control unit optimization
  - Microprocessor
  - Control in hardware
- Summary



# Control unit optimization

## Microprocessor

- Comply to the program memory available on the processor
- Rely on the compiler or do low-level programming (assembly)



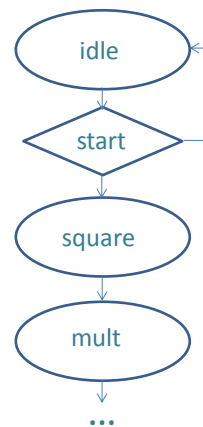
Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

41/45

# Control unit optimization

## Control in hardware – Finite State Machine (FSM)

- An FSM runs through a finite number of states
- The next state is determined based on the inputs
  - “start” in the example
  - the key bits in a scalar multiplication FSM
- The output is determined by the state (and sometimes by the inputs)

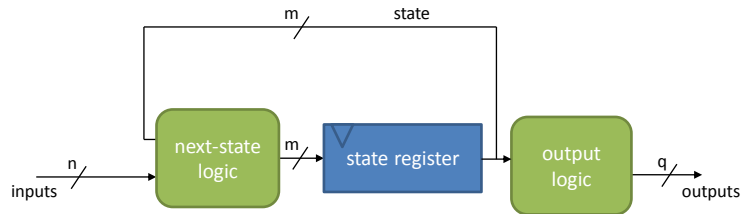


Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

42/45

# Control unit optimization

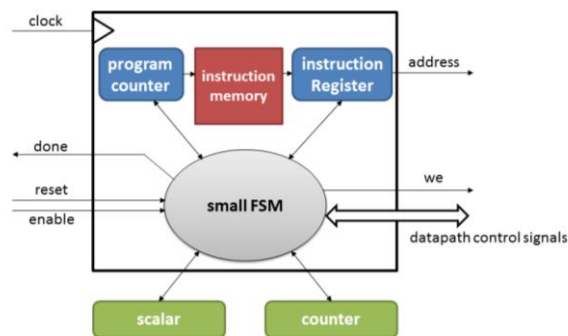
## Control in hardware – FSM in logic + flip-flops



- FPGA: using LUTs and flip-flops
- ASIC: using logic standard cells and flip-flop standard cells

# Control unit optimization

## Control in hardware – microcode approach



- FPGA: use BRAM for the instruction memory
- ASIC: use embedded RAM for the instruction memory if the size is big enough

- Introduction
- Datapath optimization
- Data storage optimization
- Control unit optimization
- Summary

Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

## Summary

- Datapath optimization
  - FPGA: maximize the use of DSP slices
  - ASIC: use optimized architectures
- Data storage optimization
  - FPGA: maximize the use of BRAM
  - ASIC: use flip-flops or embedded RAM
    - depending on the size of the memory
- Control unit optimization
  - FPGA: use FSMs in logic + FFs or microcode approach (instructions in BRAM)
    - depending on the size of the FSM
  - ASIC: use FSMs in logic + FFs or microcode approach
    - depending on the size of the FSM

Kayaks & Dreadnoughts, September 15-16, 2016, Brussels

45/45