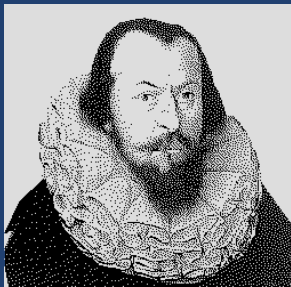


*CRC-Project*¹
(Configurable Reconfigurable Core)

Dynamically Reconfigurable
Processor-Like Architectures



Wolfgang Rosenstiel <rosenstiel@informatik.uni-tuebingen.de>

University of Tübingen

Wilhelm Schickard Institute for Computer Science

Department of Computer Engineering

¹ Funded by Deutsche Forschungsgemeinschaft (DFG)



Outline

- Introduction
- CRC Model
- Execution Schemes
- NEC DRP
- Ray Casting Example
- Temporal-Spatial Voltage Scaling
- Application-Domain Specific Architectures
- Conclusions



Introduction



- Traditional deployment of reconfigurable architectures (*FPGAs*):

change the application over time

- Newly developed devices (*processor-like reconfigurable architectures*):

reconfiguration within one clock cycle

- reconfiguration is keeping pace with the execution
- re-use of the device's components within a single application.

Commercial Examples:

- NEC: Dynamically Reconfigurable Processor (DRP)
- Intel: Reconfigurable Communications Architecture (RCA)
- IPFlex: DAP/DNA Architecture



Introduction



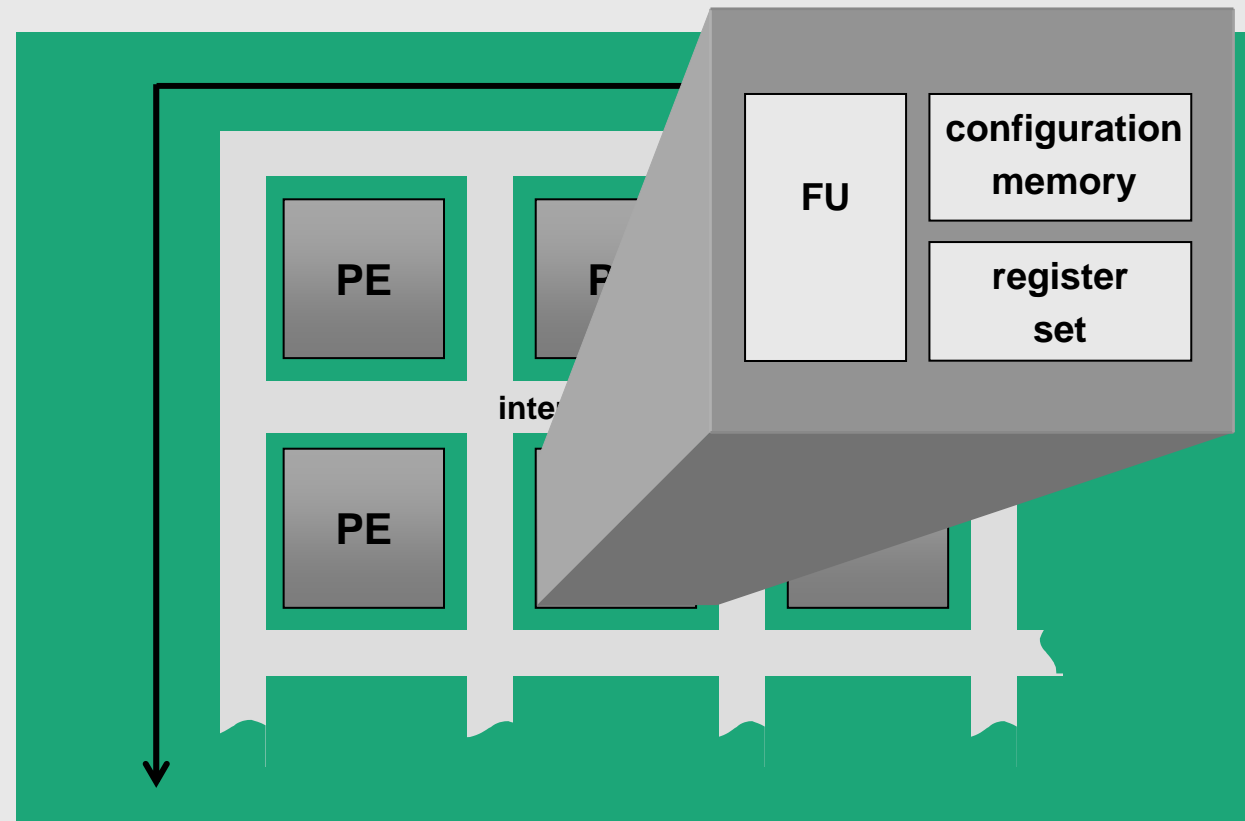
- **How to map C code onto processor-like reconfigurable architectures?**
- **What features are required for relevant applications ?**
 - number of registers, contexts, ... ?
 - interconnect network ?
 - control of reconfiguration
 - etc.
- **Benchmarking**
 - performance ?
 - power / energy ?
 - area ?



CRC Model

Configurable Reconfigurable Core

Modifiable model for processor-like reconfigurable architectures

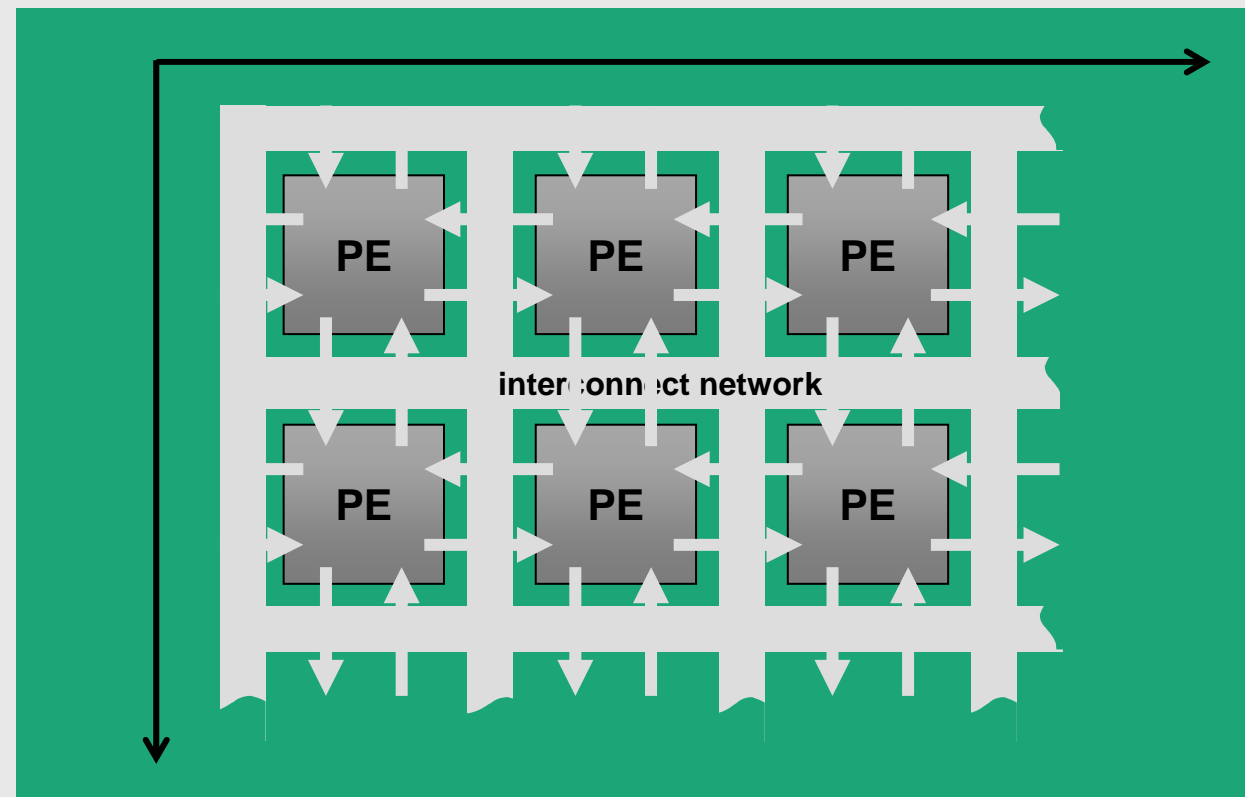




CRC Model



Synthesizable architecture instances

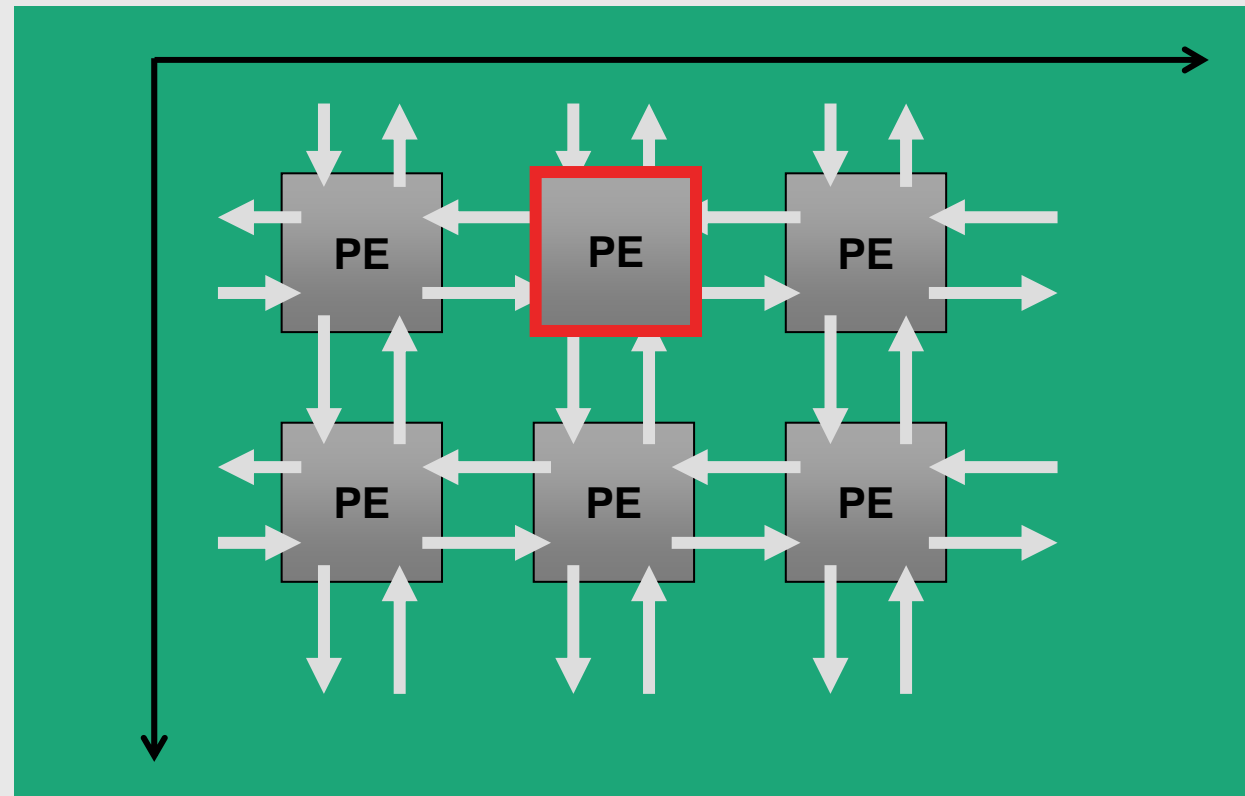




CRC Model

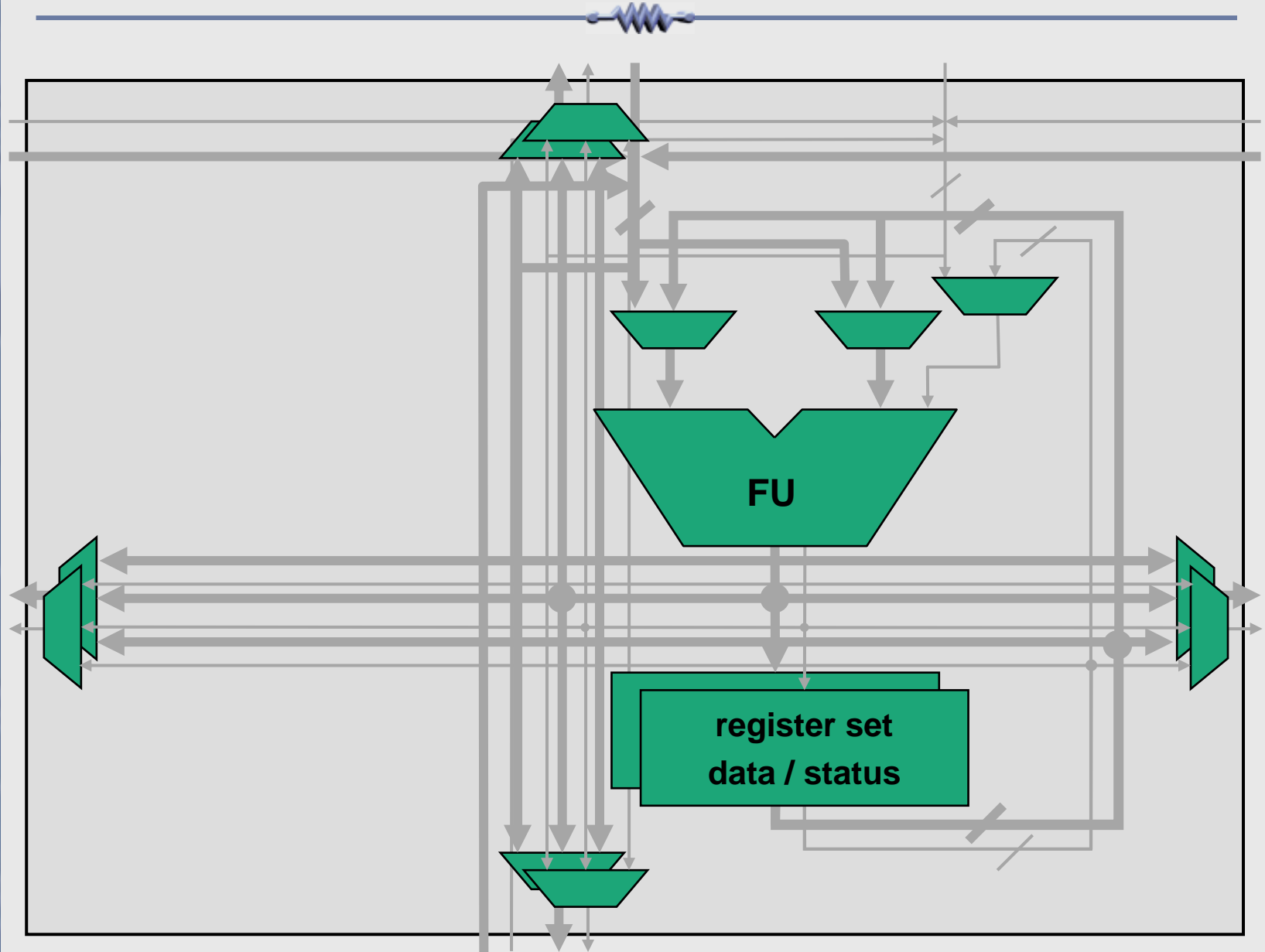


Synthesizable architecture instances



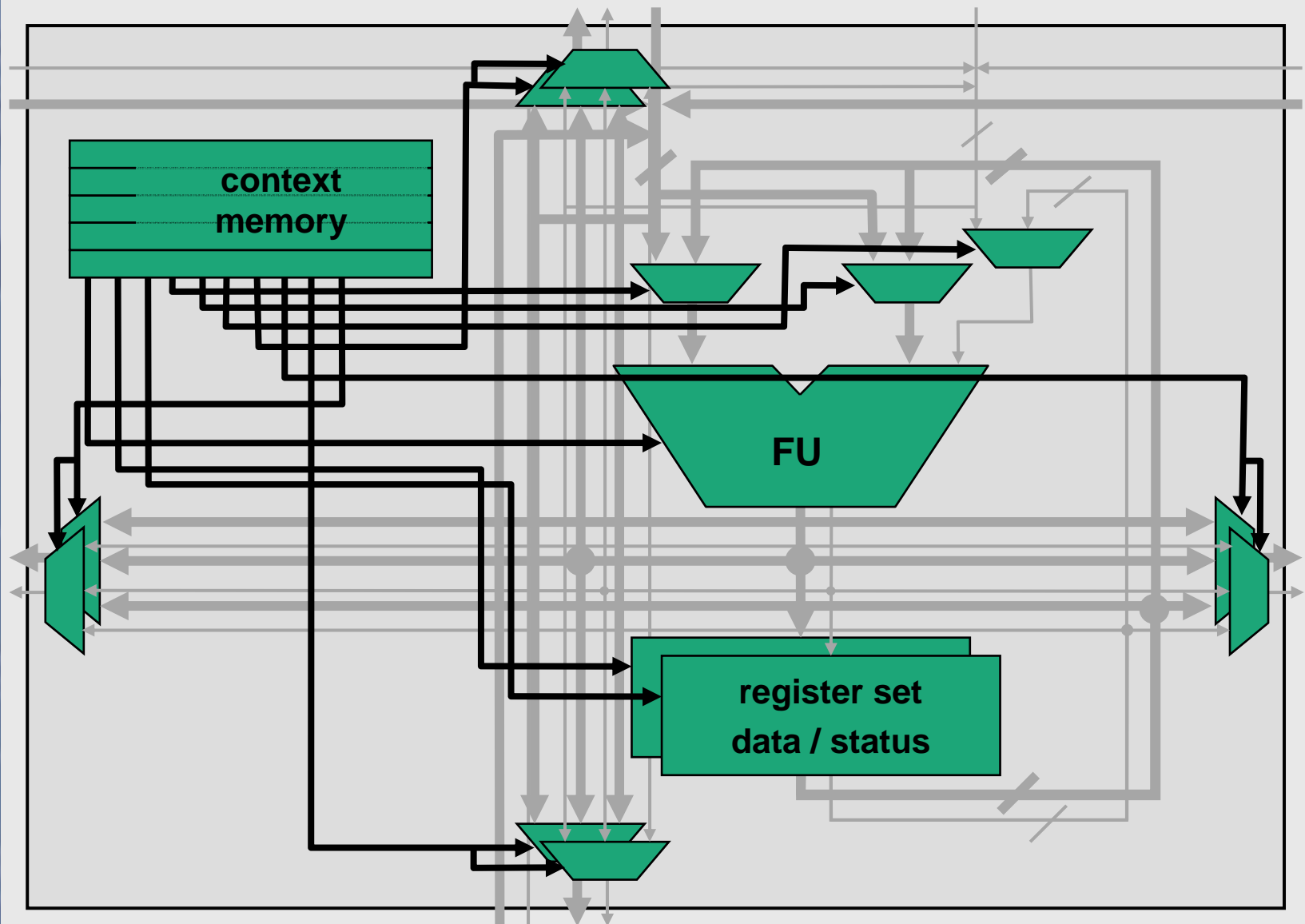


PE of the Architecture Instances



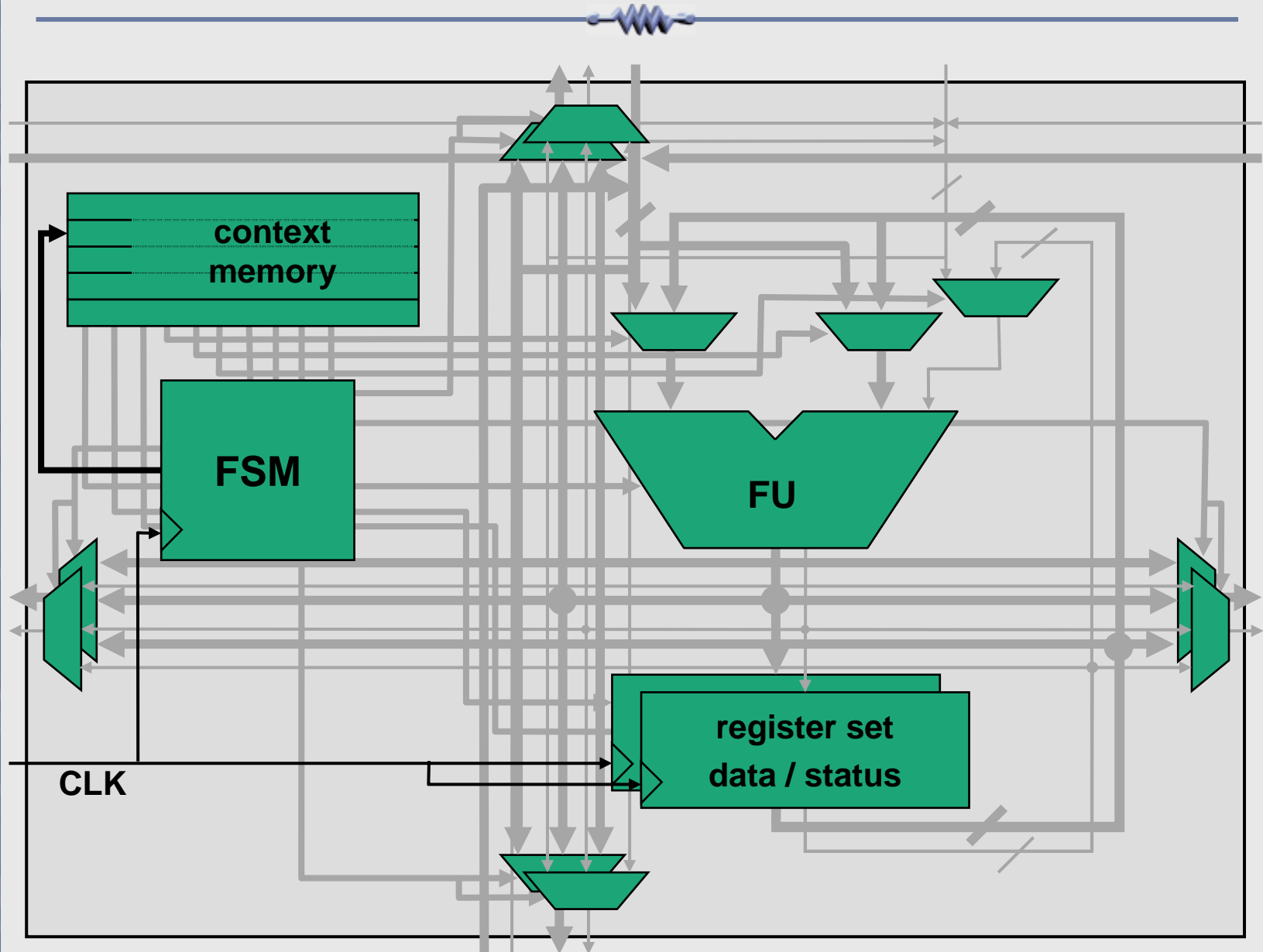


Reconfiguration of a PE



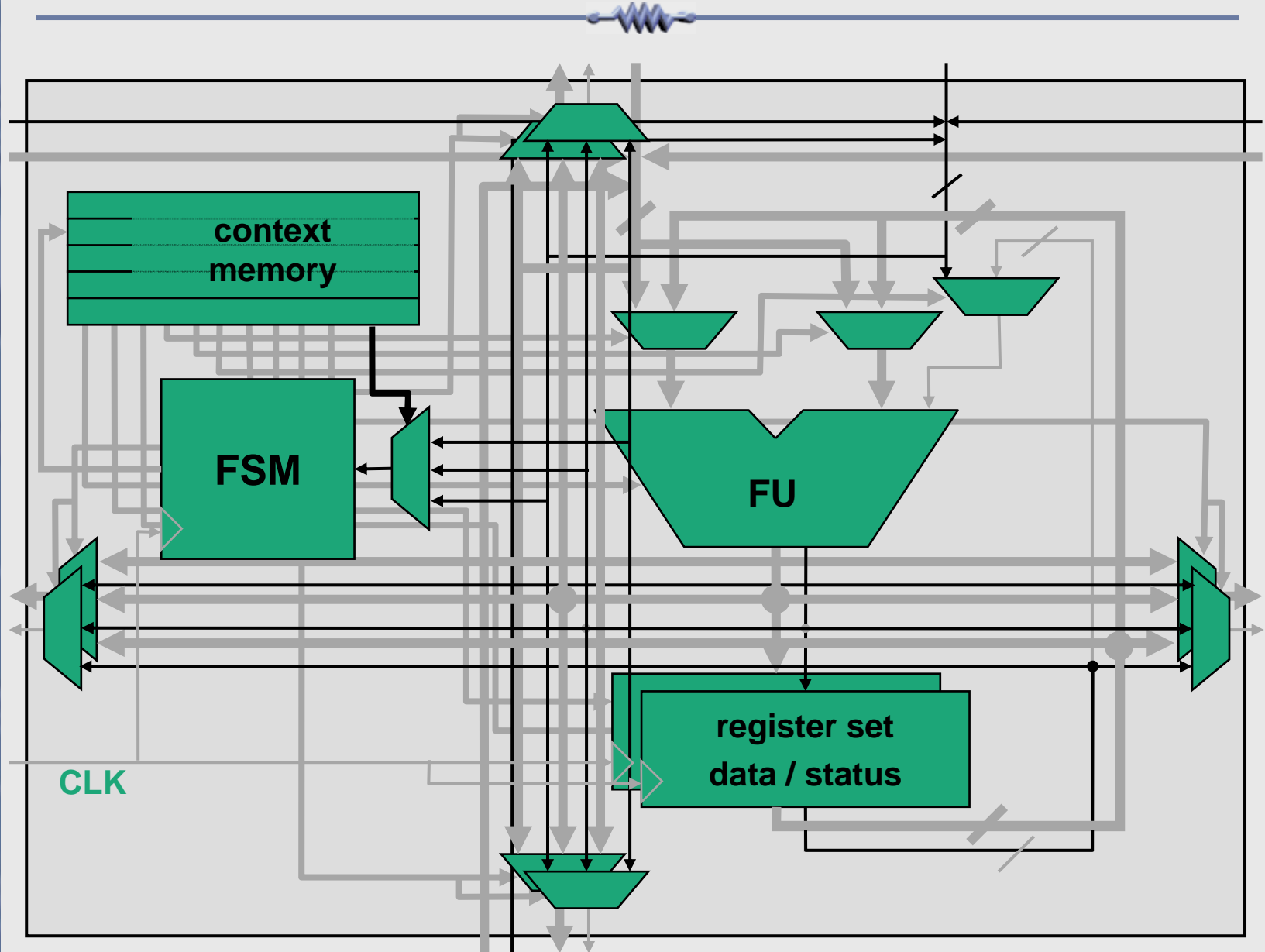


Reconfiguration of a PE





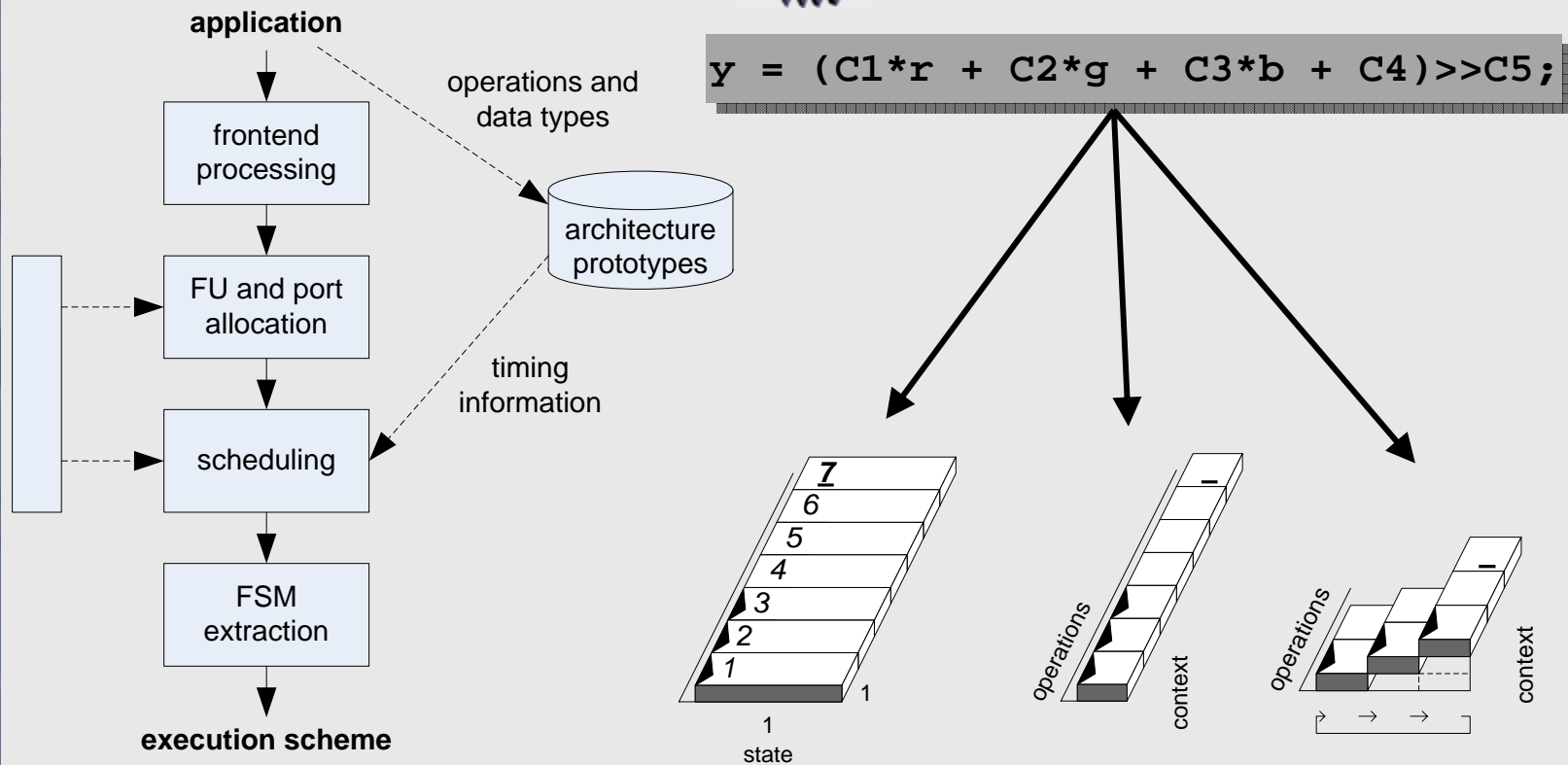
Reconfiguration of a PE



- Introduction
- CRC Model
- Execution Schemes
- NEC DRP
- Ray Casting Example
- Temporal-Spatial Voltage Scaling
- Application-Domain Specific Architectures
- Conclusions



Execution Schemes

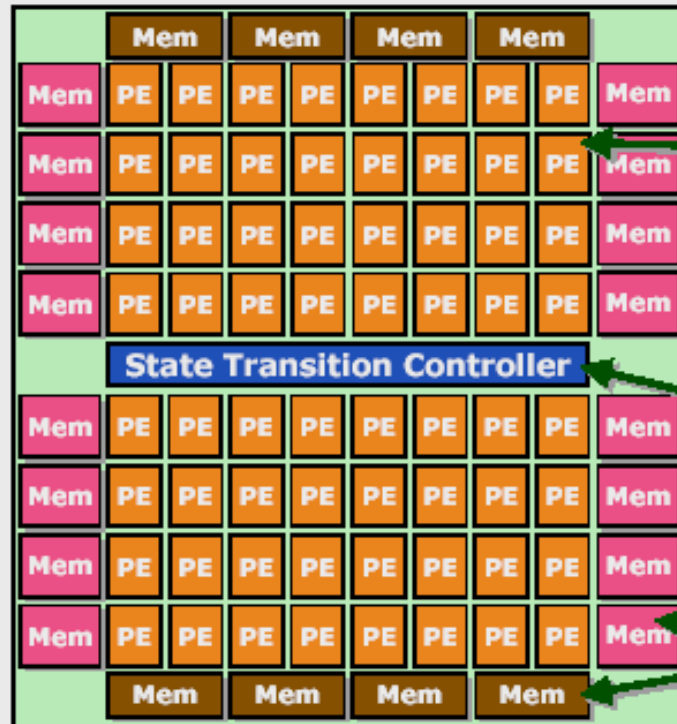


application	frontend	DIR_f	samples/s	latency	inports	FUs	states	contexts	FSM
RGB2Y	SSA	$(3/1) * 77\text{MHz}$	77,000,000	13 ns	3	7	1	1	none
RGB2Y	SSA	$(3/1) * 200\text{MHz}$	200,000,000	20 ns	3	7	1	1	none
RGB2Y	SSA	$(3/3) * 200\text{ MHz}$	66,666,667	30 ns	1	3	3	3	1 Medv.
HPGS	SSA	$(9/18) * 200\text{MHz}$	11,111,111	30 ns	1	1	18	18	1 Medv.
HPGS	none	$(9/18) * 200\text{ MHz}$	11,111,111	90 ns	1	1	18	11	1 Moore
resampling	n/a	$(8/1) * 200\text{ MHz}$	$\leq 200,000,000$	$\geq 60\text{ ns}$	8	28	2	2	1 Medv.
resampling	n/a	$(8/1) * 200\text{ MHz}$	200,000,000	60 ns	8	28	2	2	12 Medv.



NEC DRP architecture

• DRP Core (*Tile*)



- ❖ Array of byte-oriented processing elements
- ❖ Fully programmable inter-PE wiring resources

- ❖ A simple sequencer

- ❖ Array of configurable data memories

[NEC 2002]

• Implementation DRP-1: 8 *Tiles*



NEC DRP

26 operations executed in 1 context:

- 29 MHz clock speed
- 26 PEs

```

process drp()
{
DOZERIF_TYPE din, dout; // ???????

unsigned char  in0, in1, in2, in3, in4, in5, in6, in7;
unsigned char  out0, out1, out2, out3, out4, out5, out6, out7;
unsigned char  t1, t2, t3, t4, t5, t6, t7, t8, t9, t10, t11, t12, t13, t
SyncDozer();

din = DataIn(); // ???
in0 = doz2uchar(din,0); // ?
in1 = doz2uchar(din,1); // ?
in2 = doz2uchar(din,2);
in3 = doz2uchar(din,3);
in4 = doz2uchar(din,4);
in5 = doz2uchar(din,5);
in6 = doz2uchar(din,6);
in7 = doz2uchar(din,7);

t1 = in0 + in1;
t2 = in5 + in7;
t3 = t1 + in2;
t4 = t3 + in4;
t5 = t4 + t2;
t6 = t3 + t5;
t7 = t5 + t2;
t8 = t3 + t6;
t9 = t7 + t2;
t10 = t6 + t5;
t11 = t1 + t8;
t12 = t9 + in7;
out4 = t7 + t10;
t13 = t11 + t6;
t14 = t7 + t12;
out0 = t12 + in7;
t16 = t1 + t11;
t17 = t16 + in0;
t18 = t13 + in3;
t19 = t14 + in6;
    
```

RTL Variables	C/BDL Variable
inValid	inValid
inData	inData
outIntr	outIntr
outRequest	outRequest
outData	outData
outDataEnd	outDataEnd
outValid	outValid

Introduction

CRC Model

Execution
Schemes

NEC DRP

Ray Casting
Example

Temporal-Spatial
Voltage Scaling

Application-
Domain Specific
Architectures

Conclusions



NEC DRP

Placement and routing:

RAP [cvs_tag: RAP_20051122] [built: Nov 22 2005] <<< Opend DesignFile : net/project_D/project.dsn >>> <@helga> <2>

File(F) Edit(E) Arrange(G) View(V) Mode(M) Tools(T) Options(P) Report(R) Help(H)

Circuit project_D Mode SELECT

(Info) DesignLoad :: Context(1) Loading.....
(Info) DesignLoad :: Context(2) Loading.....
(Info) Net Status Check :: Context(0) Process(0) Done.
(Info) Net Status Check :: Context(1) Process(0) Done.
(Info) Net Status Check :: Context(2) Process(0) Done.



NEC DRP

26 operations executed in 11 contexts:

- 134 MHz clock speed
- 18 PEs (5 ALUs)

The screenshot shows the Musketeer workspace with the following components:

- Project Tree:** C Level Simulation, C/BDL Source Code (c, _bd), project.c, dozer_if.h, C Testbench Code (c), Input Data.
- Code Editor:** project.c with assembly-like code:


```

16: in1 = doz2uchar(din,6); // ?
17: in2 = doz2uchar(din,5);
18: in3 = doz2uchar(din,4);
19: in4 = doz2uchar(din,3);
20: in5 = doz2uchar(din,2);

36: out4 = t7 + t10;
37: t13 = t11 + t6;
38: t14 = t7 + t12;
39: out0 = t12 + in7;
40: t16 = t1 + t11;
41: t17 = t16 + in0;
42: t18 = t13 + in3;
43: t19 = t14 + in6;
44: out7 = t12 + out0;
45: out3 = t18 + in3;
46: out6 = t19 + in6;
47: out1 = t17 + t11;
48: out2 = t16 + out3;
49: out5 = t14 + out6;

51: dout = uchar2doz(out0, out1, out2, out3,
52: DataOut(dout); // ???
53:
            
```
- Timing Diagram:** Waveform showing data flow over 13 clock cycles.
- Place & Router Table:**

ALL	ALU	DMU	DELAY	MEMORY	WIRE			
PE	ALU	DMU	FFU	RFU	MX	AL	MEMORY	WIRE
#	ALU	DMU	FFU	RFU	MX	AL	WME	hME
0	0	2	0	2	2	4	0	0
1	0	0	0	0	0	-	0	0
2	1	1	8	0	8	10	0	0
3	1	1	9	0	9	11	0	0
4	2	1	12	0	12	15	0	0
5	1	1	13	0	13	15	0	0
6	2	1	14	0	14	17	0	0
7	2	1	16	0	16	19	0	0
8	2	1	16	0	16	19	0	0
9	2	1	16	0	16	19	0	0
10	3	1	17	0	17	21	0	0
11	5	1	18	0	18	24	0	0
12	5	2	10	0	10	17	0	0
13	14	31	2	151	191	0	0	0
- Configuration Panel:** Frequency: 134 MHz, PE: 18, VMEM: 11, HMEM: 11.
- RTL Variables Table:**

RTL Variables	C/BDL Variables	01	02	03	04	05	06	07	08	09	10	11	12	13
RG_drp_t9	t9													
RG_drp_t8	t8													
RG_drp_t7	t7													
RG_drp_t6	t6													
RG_drp_out0	out0													
RG_drp_t5	t5													
RG_drp_t10	t10													
RG_drp_t4	t4													

Introduction

CRC Model

Execution
Schemes

NEC DRP

Ray Casting
Example

Temporal-Spatial
Voltage Scaling

Application-
Domain Specific
Architectures

Conclusions



NEC DRP

software pipelining
26 operations executed in 4 contexts:

- 129 MHz clock speed
- 15 PEs (8 ALUs)

Place & Router : table : multicontext_pipeline.ide

ALL	ALU	DMU	DELAY	MEMORY	WIRE CON	REG				
PE	ALU	DMU	FFU	RFU	MX	AL	MEMORY	UNF		
#	ALU	DMU	FFU	RFU	MX	AL	vME	hME	MC	MUL
0	0	2	0	2	2	4	0	0	0	0
1	0	1	0	0	1	1	0	0	0	0
2	8	1	9	14	14	32	0	0	0	0
3	7	1	15	6	15	29	0	0	0	0
4	6	1	15	6	15	28	0	0	0	0
5	5	2	11	9	11	27	0	0	0	0
6	4	2	11	9	11	27	0	0	0	0
7	3	2	15	14	15	32	0	0	0	0
8	2	8	19	26	58	121	0	0	0	0

RTL Variables

RTL Variables	C/BDL Variables	01	02	03	04	05	06
t11	t11	H	H	R	W	R	
t12	t12	H	R		W	R	
t13	t13	H	R			W	
t14	t14	H	R	H	R	W	
t16	t16	H	R	H	R	W	
t17	t17		W	R			
t18	t18		W	R			
t19	t19		W	R			

Introduction

CRC Model

Execution
Schemes

NEC DRP

Ray Casting
Example

Temporal-Spatial
Voltage Scaling

Application-
Domain Specific
Architectures

Conclusions



NEC DRP



Placement and routing context 1:

RAP [cvs_tag: RAP_20051122] [built: Nov 22 2005] <<< Opend DesignFile : net/project_D/project.dsn >>> <@helga>

File(F) Edit(E) Arrange(G) View(V) Mode(M) Tools(T) Options(P) Report(R) Help(H)

Circuit project_D Mode SELECT

(Info) Net Status Check :: Context(2) Process(0) Done.
 (Info) Net Status Check :: Context(3) Process(0) Done.
 (Info) Net Status Check :: Context(4) Process(0) Done.
 (Info) Net Status Check :: Context(5) Process(0) Done.
 (Info) I_RAP03001 | <DesignOpen> :: Loaded Design File [net/project_D/project.dsn]

Introduction

CRC Model

Execution
Schemes

NEC DRP

Ray Casting
Example

Temporal-Spatial
Voltage Scaling

Application-
Domain Specific
Architectures

Conclusions



NEC DRP



Placement and routing context 2:

RAP [cvs_tag: RAP_20051122] [built: Nov 22 2005] <<< Opend DesignFile : net/project_D/project.dsn >>> <@helga>

File(F) Edit(E) Arrange(G) View(V) Mode(M) Tools(T) Options(P) Report(R) Help(H) Session

Circuit project_D Mode SELECT

```

(Info) Net Status Check :: Context( 2) Process(0) Done.
(Info) Net Status Check :: Context( 3) Process(0) Done.
(Info) Net Status Check :: Context( 4) Process(0) Done.
(Info) Net Status Check :: Context( 5) Process(0) Done.
(Info) I_RAP03001 | <DesignOpen> :: Loaded Design File [net/project_D/project.dsn]
    
```

Introduction

CRC Model

Execution
Schemes

NEC DRP

Ray Casting
Example

Temporal-Spatial
Voltage Scaling

Application-
Domain Specific
Architectures

Conclusions



NEC DRP



Placement and routing context 3:

The screenshot shows the RAP software interface. The main window displays a PCB layout grid with various components placed on it. The components are represented by small icons and labels, such as 'U1', 'U2', 'U3', 'U4', 'U5', 'U6', 'U7', 'U8', 'U9', 'U10', 'U11', 'U12', 'U13', 'U14', 'U15', 'U16', 'U17', 'U18', 'U19', 'U20', 'U21', 'U22', 'U23', 'U24', 'U25', 'U26', 'U27', 'U28', 'U29', 'U30', 'U31', 'U32', 'U33', 'U34', 'U35', 'U36', 'U37', 'U38', 'U39', 'U40', 'U41', 'U42', 'U43', 'U44', 'U45', 'U46', 'U47', 'U48', 'U49', 'U50', 'U51', 'U52', 'U53', 'U54', 'U55', 'U56', 'U57', 'U58', 'U59', 'U60', 'U61', 'U62', 'U63', 'U64', 'U65', 'U66', 'U67', 'U68', 'U69', 'U70', 'U71', 'U72', 'U73', 'U74', 'U75', 'U76', 'U77', 'U78', 'U79', 'U80', 'U81', 'U82', 'U83', 'U84', 'U85', 'U86', 'U87', 'U88', 'U89', 'U90', 'U91', 'U92', 'U93', 'U94', 'U95', 'U96', 'U97', 'U98', 'U99', 'U100'. The grid is overlaid with a yellow and black pattern. The console window at the bottom shows the following text:

```
(Info) Net Status Check :: Context( 2) Process(0) Done.
(Info) Net Status Check :: Context( 3) Process(0) Done.
(Info) Net Status Check :: Context( 4) Process(0) Done.
(Info) Net Status Check :: Context( 5) Process(0) Done.
(Info) I_RAP03001 | <DesignOpen> :: Loaded Design File [net/project_D/project.dsn]
```

Introduction

CRC Model

Execution
Schemes

NEC DRP

Ray Casting
Example

Temporal-Spatial
Voltage Scaling

Application-
Domain Specific
Architectures

Conclusions



NEC DRP

Placement and routing context 4:

RAP [cvs_tag: RAP_20051122] [built: Nov 22 2005] <<< Opend DesignFile : net/project_D/project.dsn >>> <@helga>

File(F) Edit(E) Arrange(G) View(V) Mode(M) Tools(T) Options(P) Report(R) Help(H)

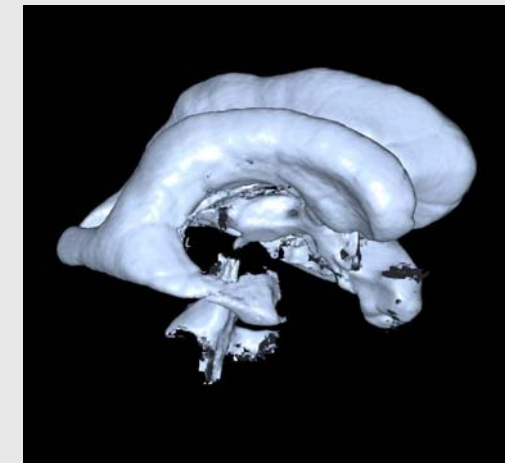
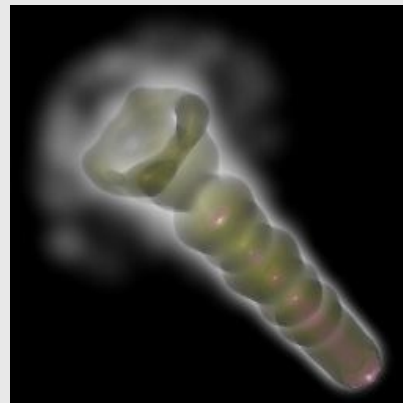
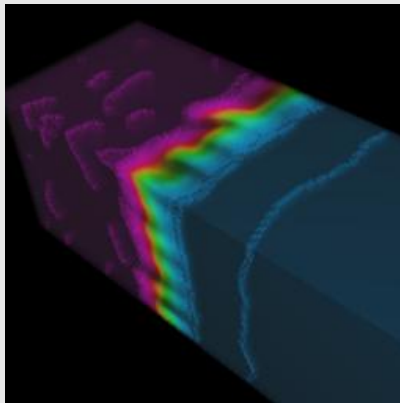
Circuit project_D Mode SELECT

(Info) Net Status Check :: Context(2) Process(0) Done.
 (Info) Net Status Check :: Context(3) Process(0) Done.
 (Info) Net Status Check :: Context(4) Process(0) Done.
 (Info) Net Status Check :: Context(5) Process(0) Done.
 (Info) I_RAP03001 | <DesignOpen> :: Loaded Design File [net/project_D/project.dsn]



Example Application from Visual Computing

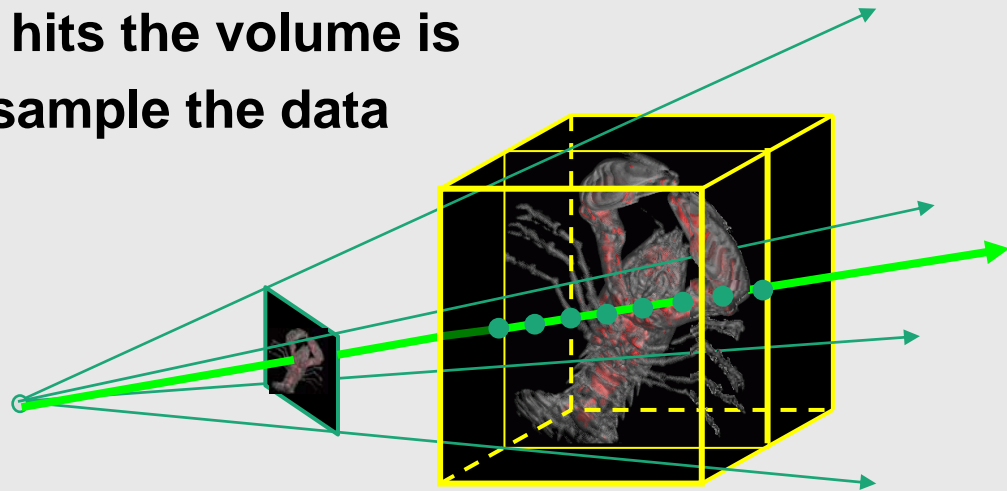
- Visual representation of scientific and medical application data





Ray Casting

- **Direct volume rendering (DVR): a visual computing method to display 3D volumetric data on a 2D screen**
- **A popular algorithm for volume rendering is ray casting:**
 - **the viewing rays from the eye position through the image plane are tested for intersection with the volume**
 - **each ray that hits the volume is traversed to sample the data**





Ray Casting



- **Limiting factors: computational power and memory bandwidth**
- **Current solutions for real-time performance:**

ASIC

- not flexible enough for extensions
- high NRE costs

CPU with multimedia extensions (SSE2, 3DNow)

- prone to fail for even slight changes in processor or memory architecture
- tedious to code

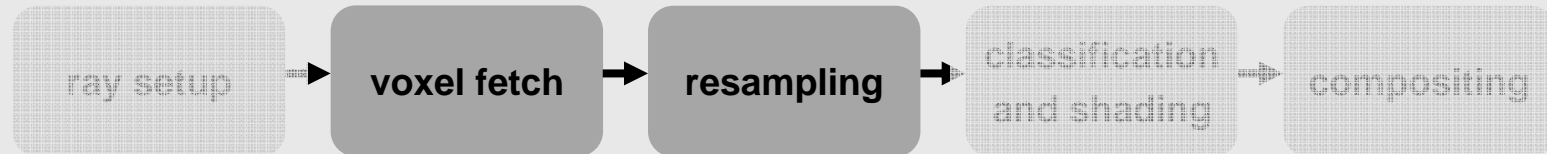
FPGA

- not suitable for low-power requirements



Ray Casting on the CRC Model

Ray casting pipeline



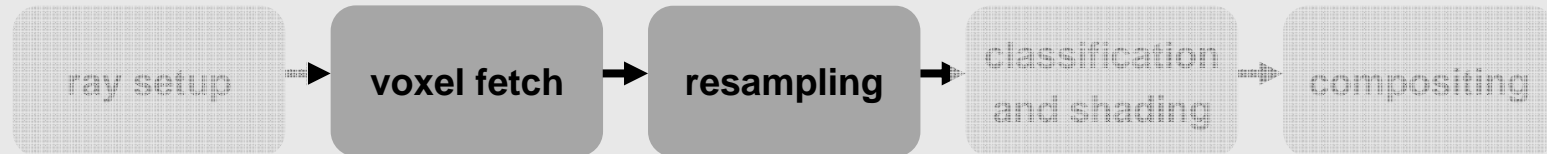
**voxel fetch:
memory bandwidth
limited**

**resampling:
most computation
intensive stage**



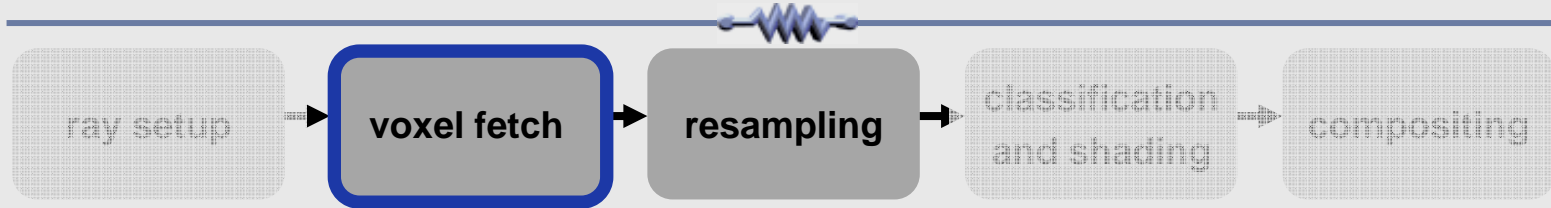
Ray Casting on the CRC Model

Ray casting pipeline





Ray Casting on the CRC Model



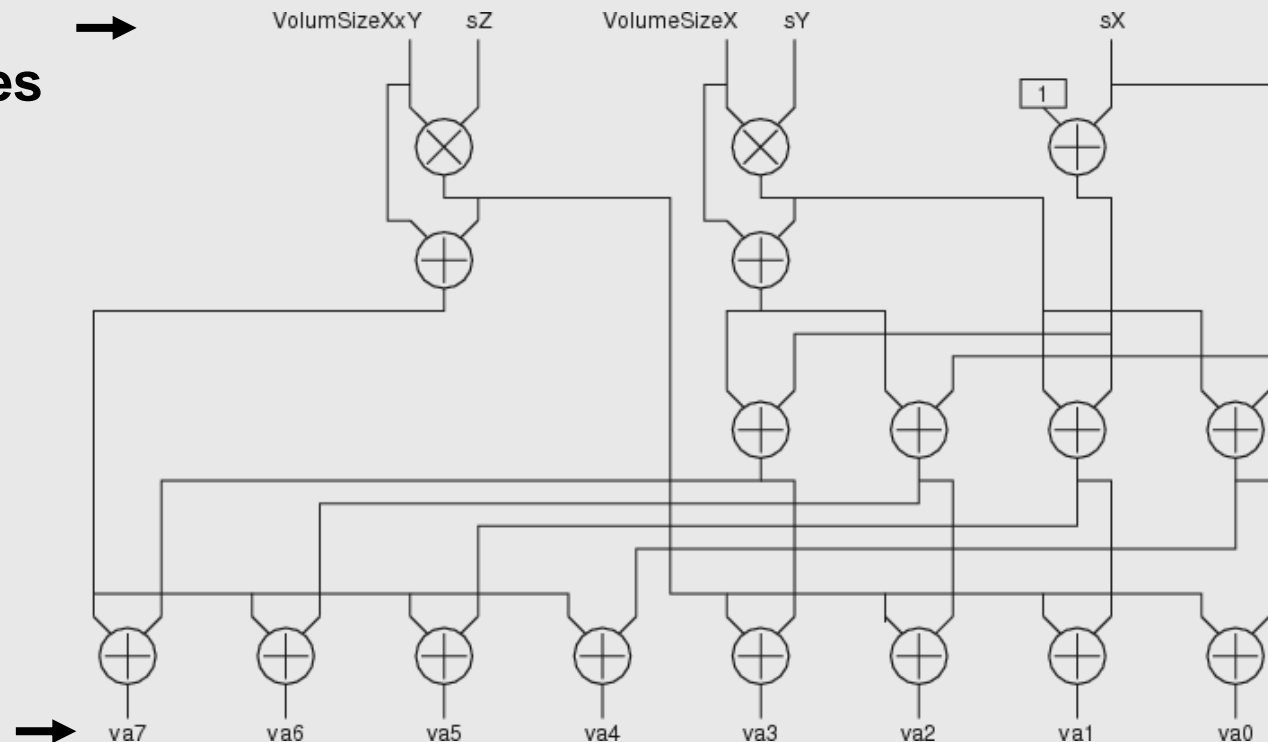
Optimized data flow graph:

3 sample coordinates

* 2
+ 15

8 memory addresses

to fetch a 2 x 2 x 2 neighborhood of voxels





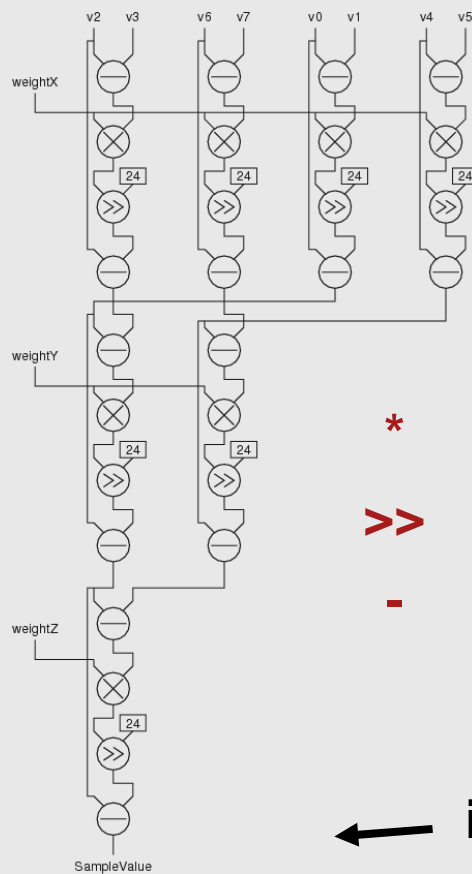
Ray Casting on the CRC Model



trilinear interpolation

2 alternative filter kernels

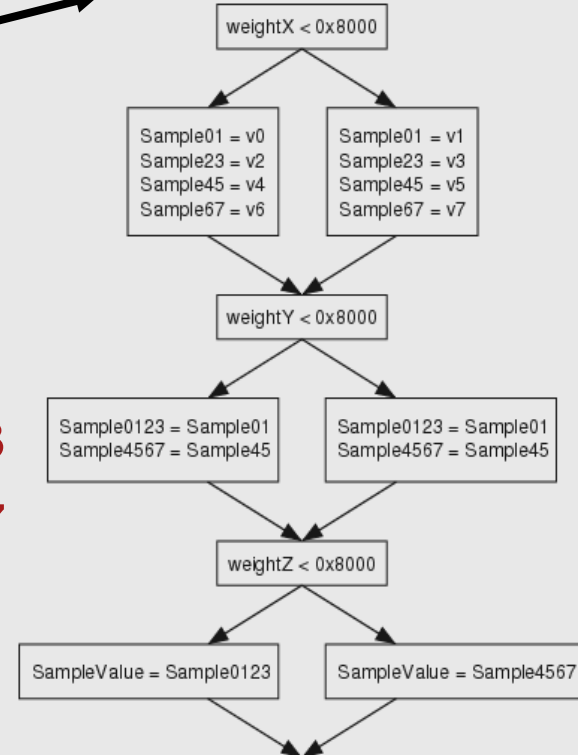
nearest neighbor interpolation



8 voxels

*** 7**
>> 7
- 14

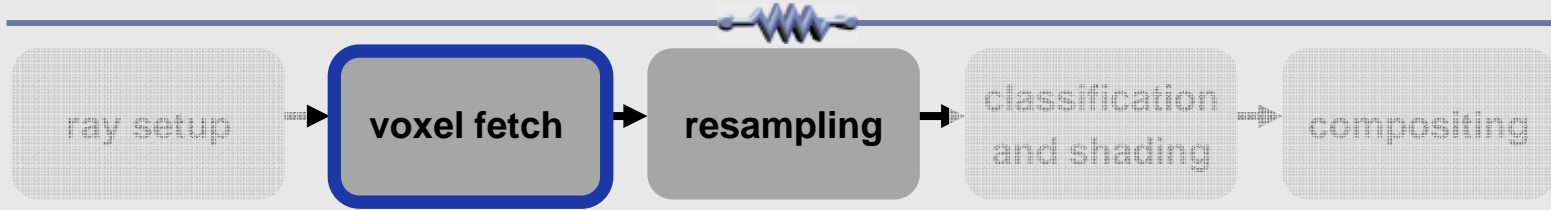
< 3
SEL 7



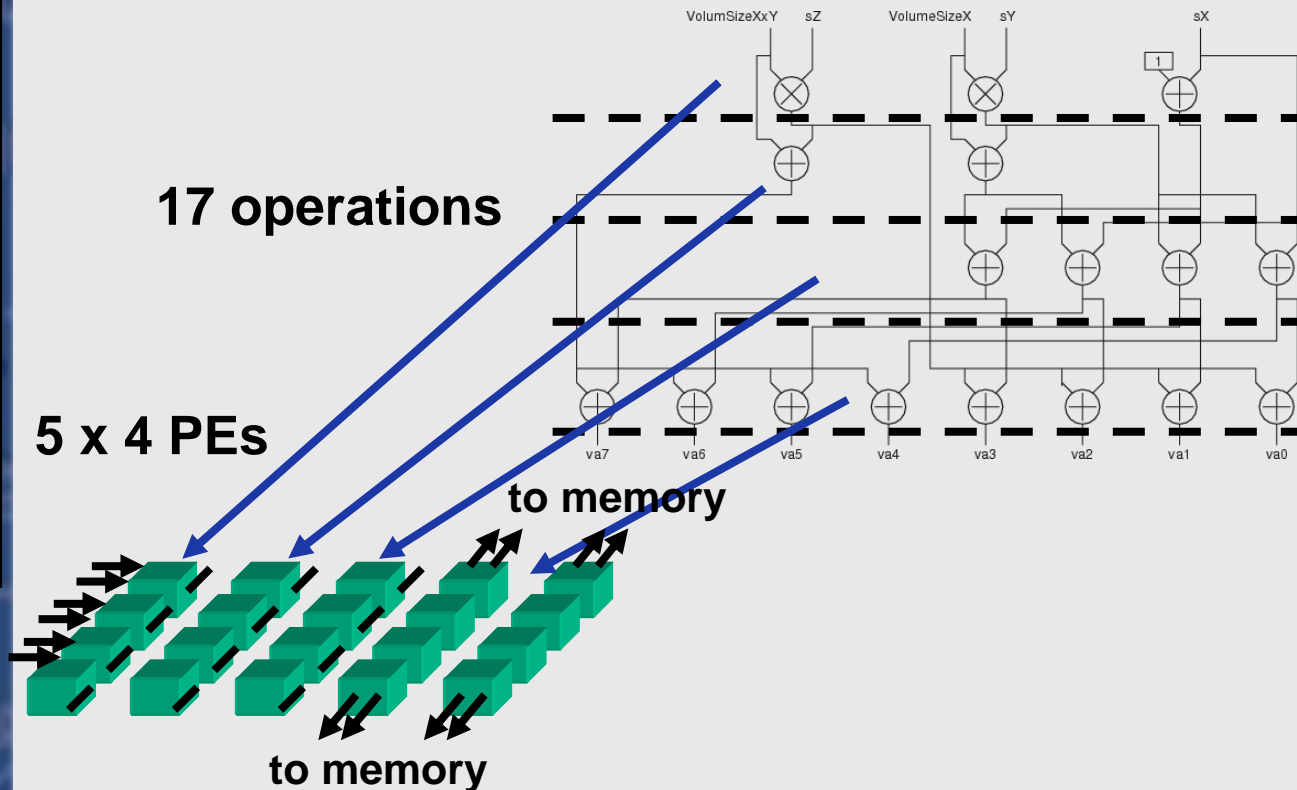
interpolated sample value



Architecture for High Throughput

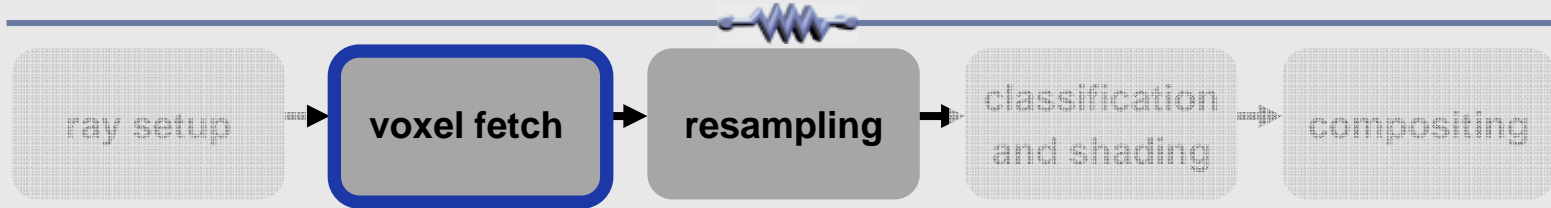


Pipeline is subdivided into (super-)pipeline stages to achieve high throughput

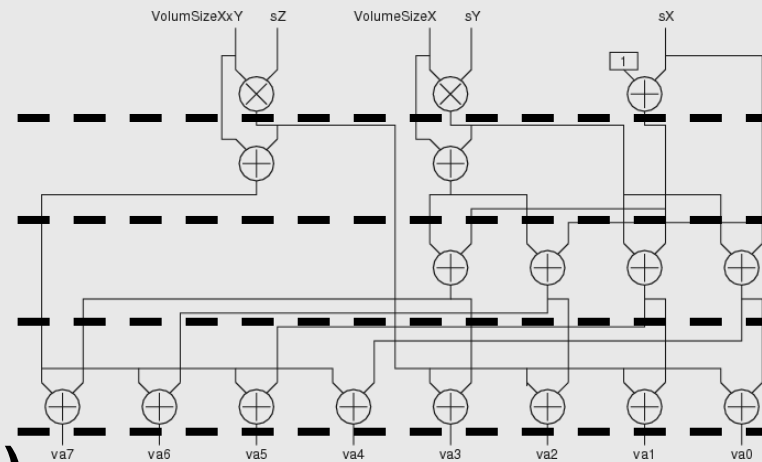




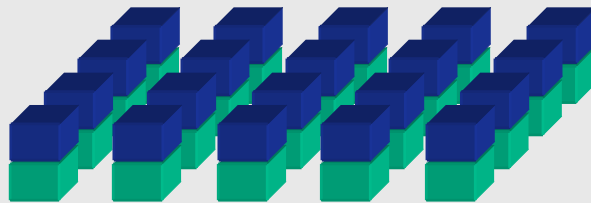
Architecture for High Throughput



Pipeline is subdivided into (super-)pipeline stages to achieve high throughput



**1 context
(no reconfiguration)**





Architecture for High Throughput

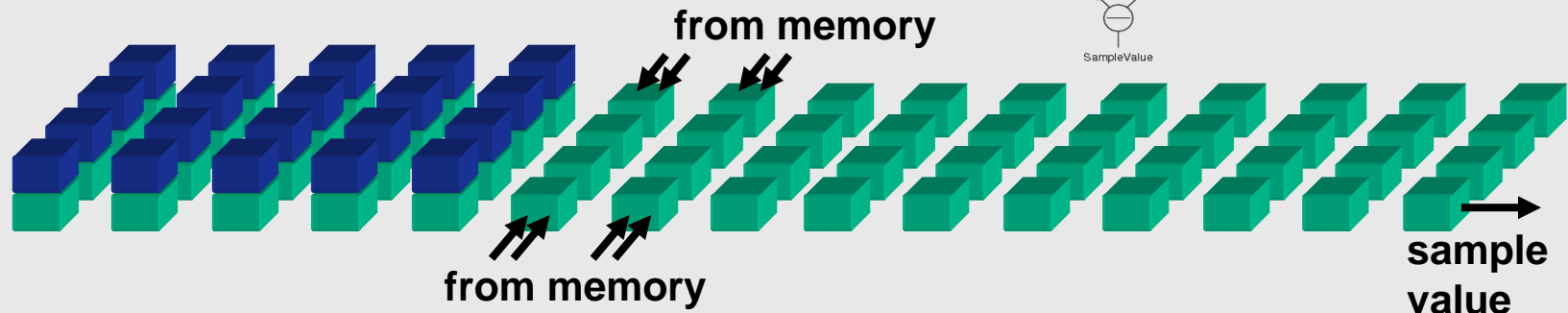
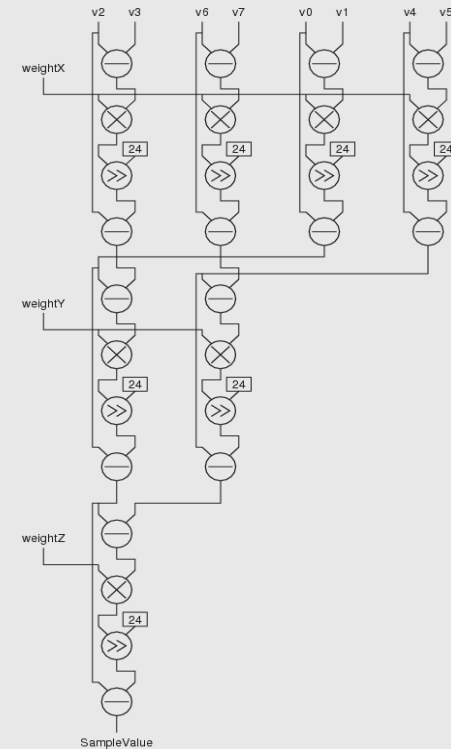


Filter kernel for resampling:

trilinear interpolation

28 operations

10 x 4 PEs



Introduction

CRC Model

Execution
Schemes

NEC DRP

Ray Casting
Example

Temporal-Spatial
Voltage Scaling

Application-
Domain Specific
Architectures

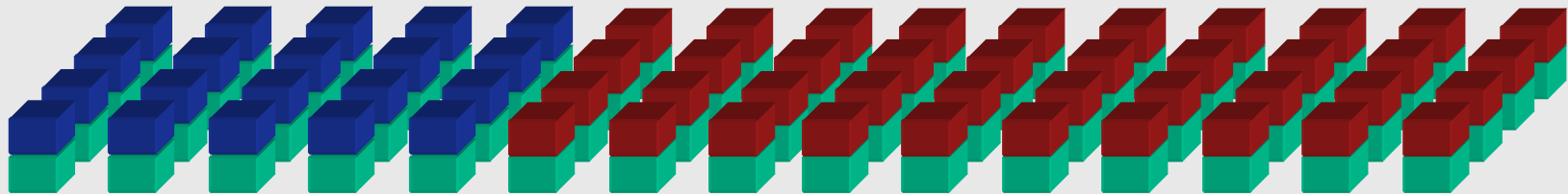
Conclusions



Architecture for High Throughput



trilinear interpolation





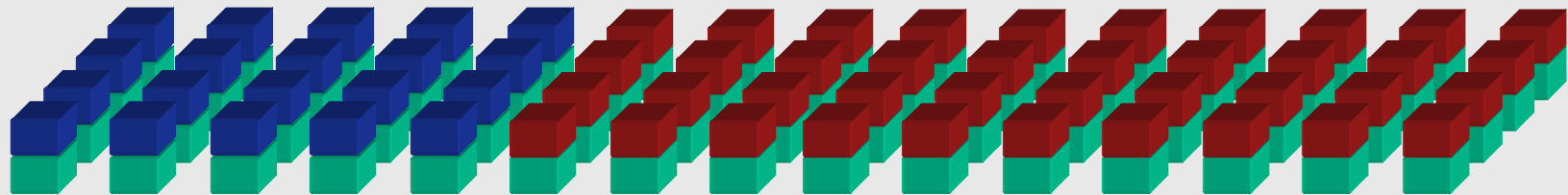
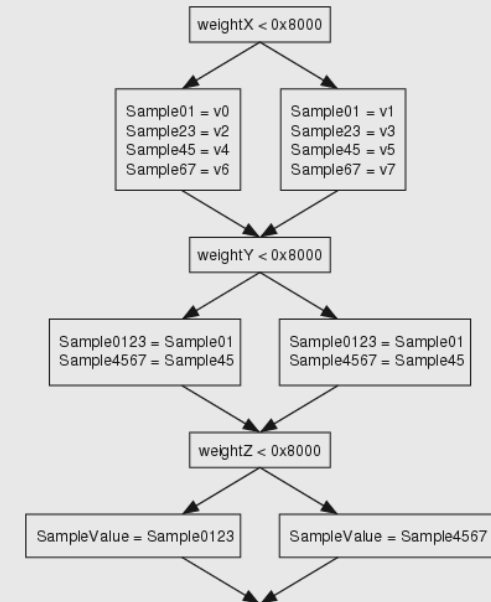
Architecture for High Throughput



Alternative filter kernel:

nearest neighbor interpolation

10 operations





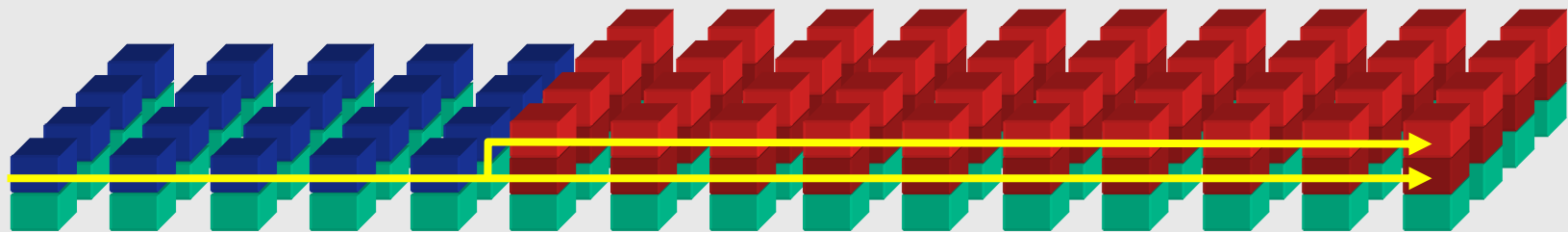
Architecture for High Throughput



nearest neighbor interpolation

trilinear interpolation

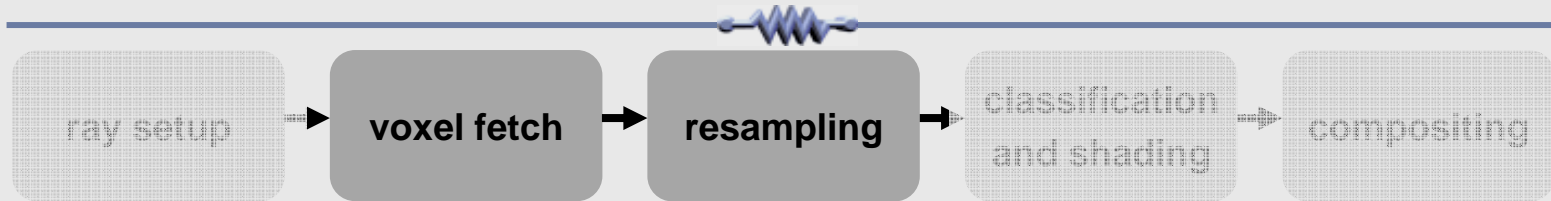
Filter kernel can be selected dynamically



selection of filter kernel

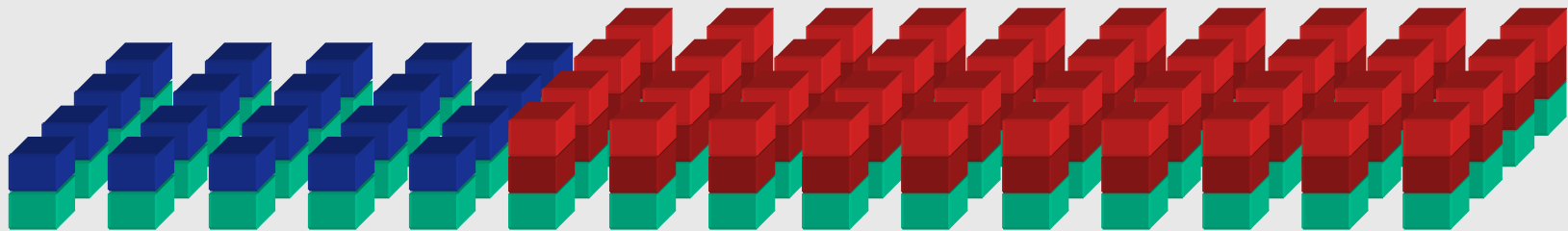


Architecture for High Throughput



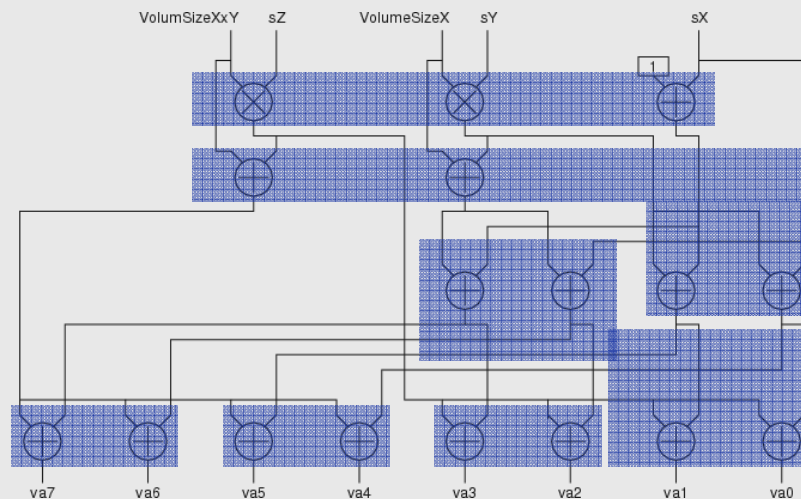
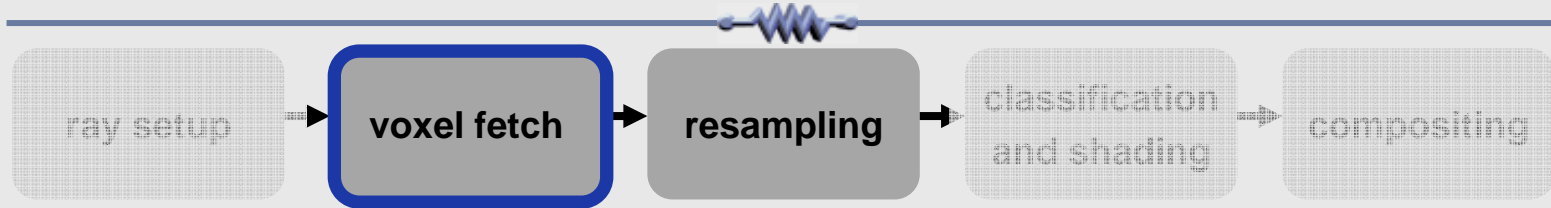
Super-pipelined mapping:

- 15 x 4 PEs
- 2 contexts and 2 states
- 8 memory blocks
- 1 clock cycle per sample





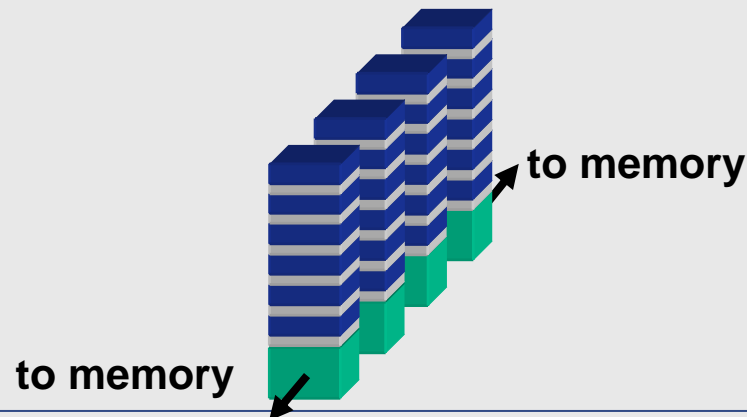
Architecture for Low Area



Sequential execution using 4 PEs

- 6 contexts
- 6 states
- 2 memory blocks

Similar to a VLIW processor





Architecture for Low Area



Multi-context mapping:

- 2 x 4 PEs
- 22 contexts and 22 states
- 2 memory blocks
- 15 clock cycles per sample



nearest neighbor interpolation:

**6 contexts
+ 1 context for
synchronization**

trilinear interpolation

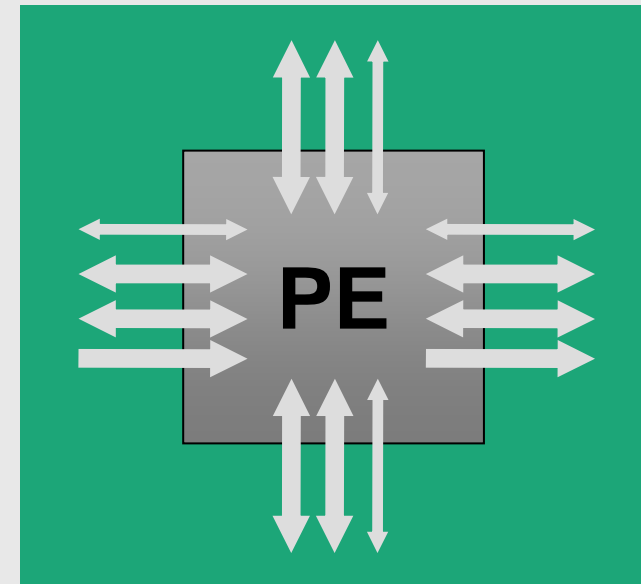
15 contexts



PEs for Ray Casting

- **32-bit wide arithmetic operations**
- **16-bit x 16-bit multipliers**
- **7 data registers**
- **3 status registers**
- **4 states, 4 contexts for high throughput**
- **24 states, 24 contexts for low area**

- **interconnect network:**





Benchmark Results

	area	clock speed	samples per second	power dissipation	energy per sample
CRC 4 contexts (super-pipelined)	60 PEs (5.85 mm ²)	163.4 MHz	163 400 000	372 mW	2277 pJ
CRC 24 contexts (multi-context)	8 PEs (1.29 mm ²)	162.1 MHz	10 800 000	74,7 mW	6913 pJ
Virtex-II 1000 (super-pipelined)	341 Slices 9 18x18 mult.	140 MHz	140 000 000	983 mW	7014 pJ

CRC model: 130 nm standard cell technology

Virtex-II: 150 nm technology (120 nm transistors)



Slack Time









Voltage [V]

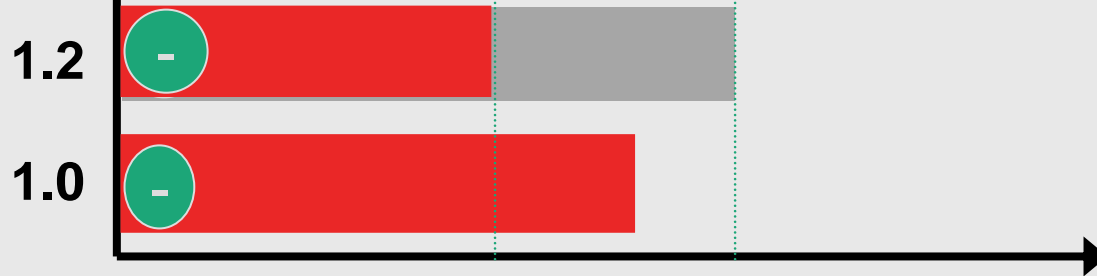
1.2

1.0

Slack

Delay [t]

	 1.0 V	 1.2 V
	5.45 ns	4.06 ns
	3.48 ns	2.68 ns
	2.29 ns	1.38 ns
	2.80 ns	2.10 ns
P	2.32 mW	3.71 mW



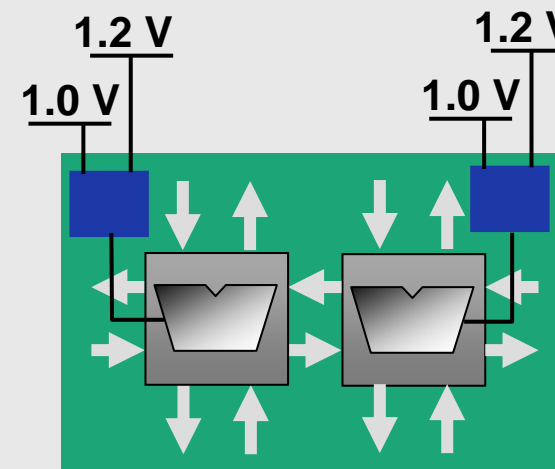
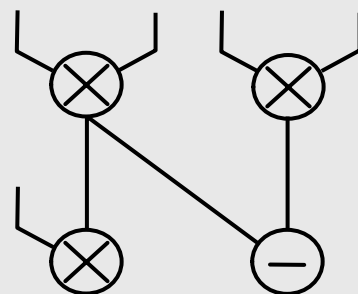


Temporal-Spatial Voltage Scaling on the CRC-Model



	 1.0 V	 1.2 V
	5.45 ns	4.06 ns
	3.48 ns	2.68 ns
P	2.32 mW	3.71 mW


**timing
constraint
4.5 ns**



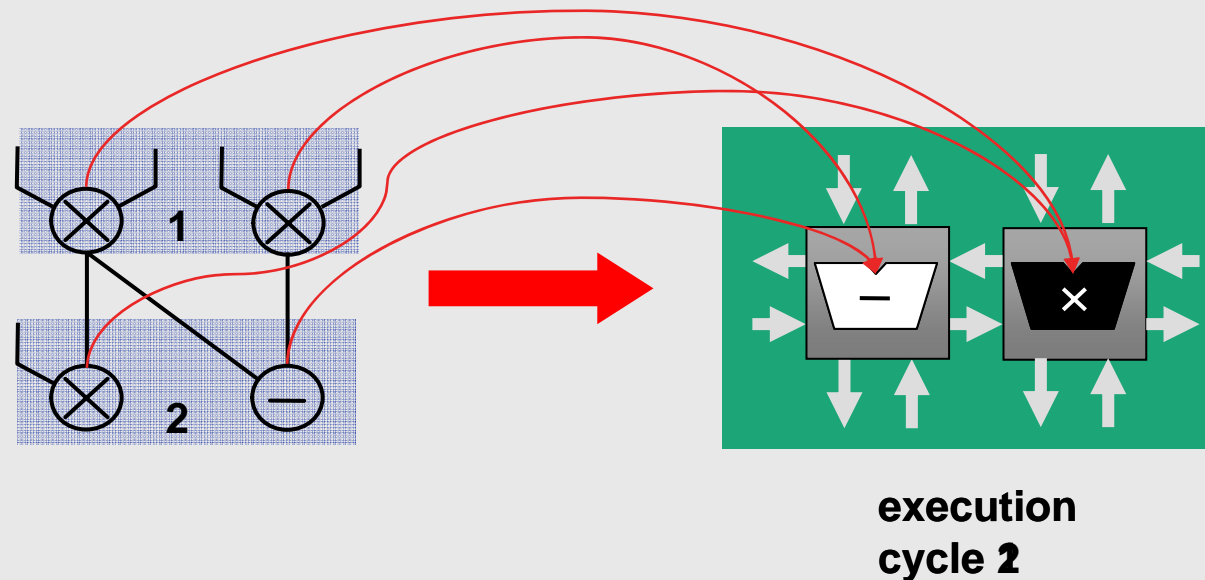


Temporal-Spatial Voltage Scaling on the CRC-Model



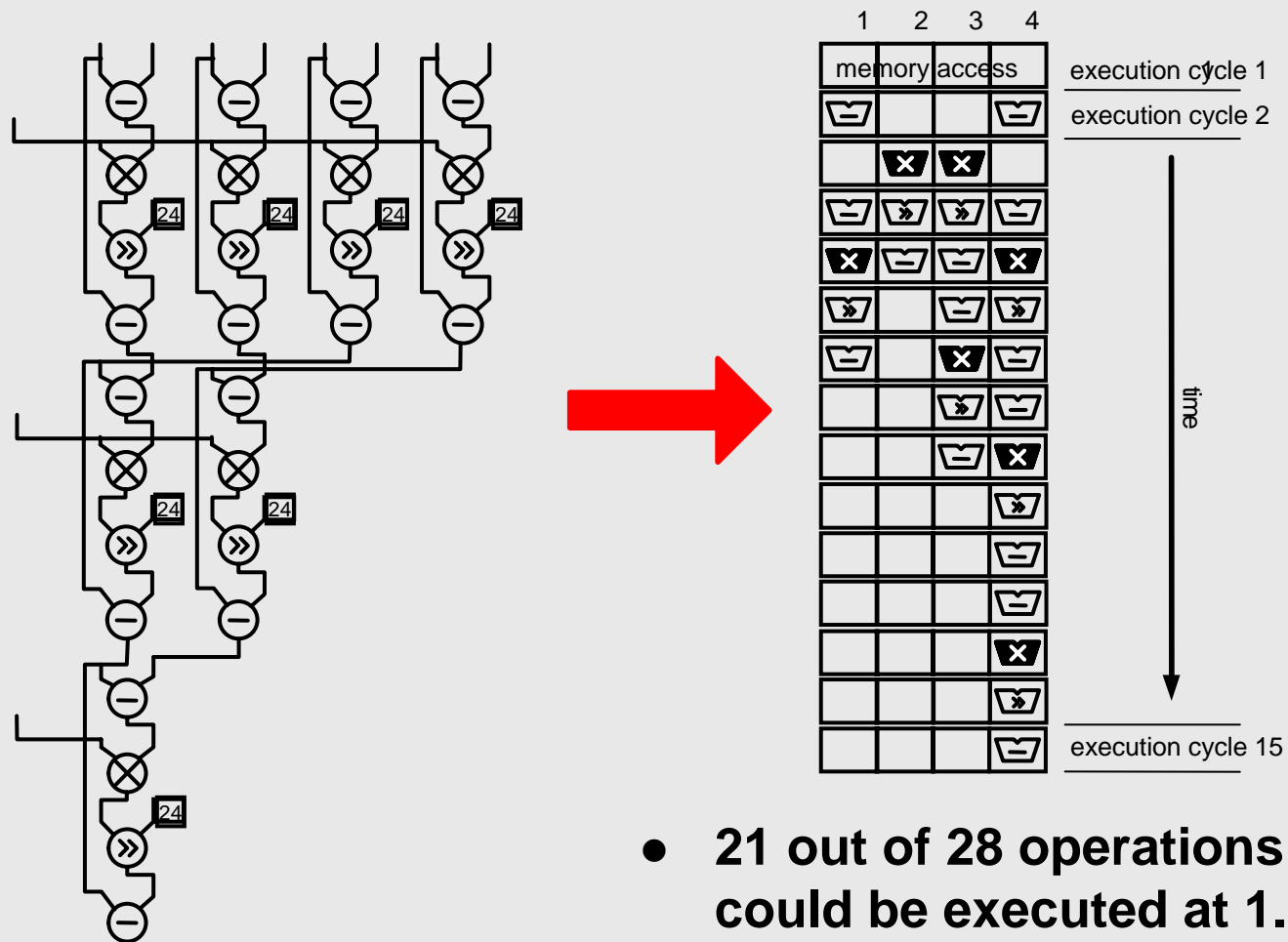
	 1.0 V	 1.2 V
	5.45 ns	4.06 ns
	3.48 ns	2.68 ns
P	2.32 mW	3.71 mW

**timing
constraint
4.5 ns**





Trilineare Interpolation

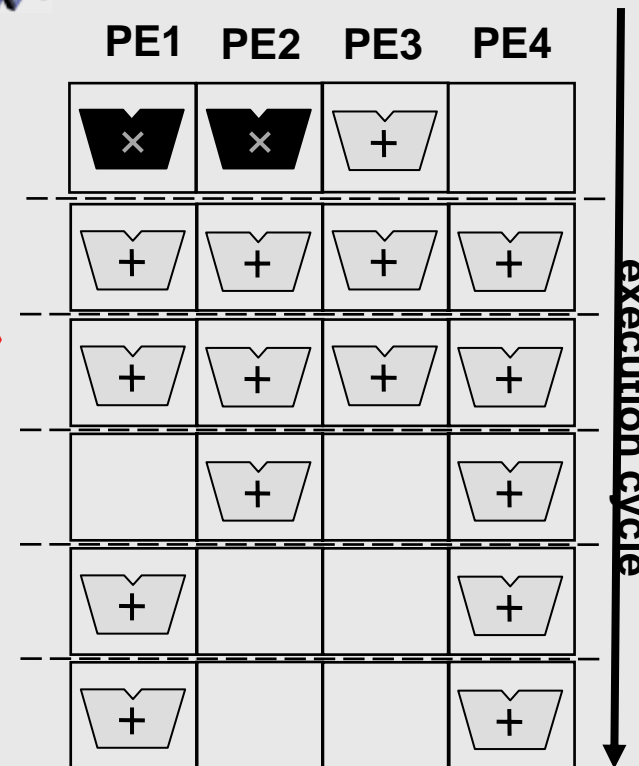
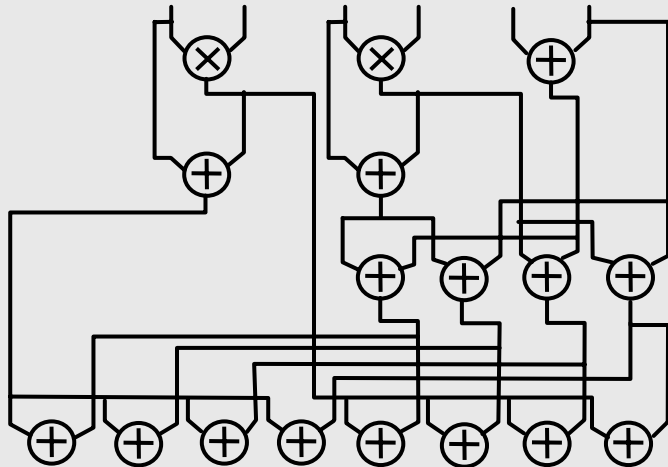


- **21 out of 28 operations could be executed at 1.0 V**
- ➔ **23% energy gain**
- ➔ **23% EDP Reduction**



Voxel Fetch

data flow graph











- **15 out of 17 operations could be executed at 1.0 V**
- **28 % energy gain**
- **28% EDP Reduction**



Application Domain: Image Processing

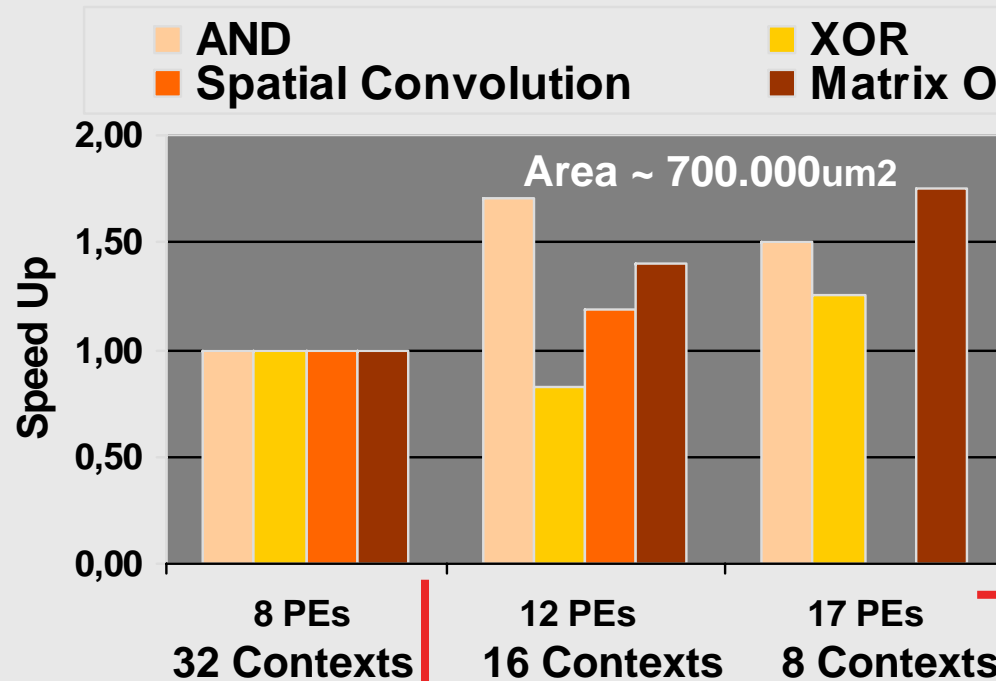


<p>Histogram Stretching</p>		
<p>Binary AND und Binary XOR</p>		<p>AND XOR</p> 
<p>Spatial Convolution 3x3 und 5x5</p>		
<p>Matrix Operations</p>		



Adequacy of the Architecture to the Domain

Which one is the most adequate architecture to the domain?



... and which is the most adequate mapping strategy?

*Loop Unrolling,
Constant folding,
Copy Propagation*

*Loop Unrolling, Constant
folding, Common Sub-
Expression Elimination*

*Loop Unrolling,
Constant folding,
Constant Propagation,
Copy Propagation*

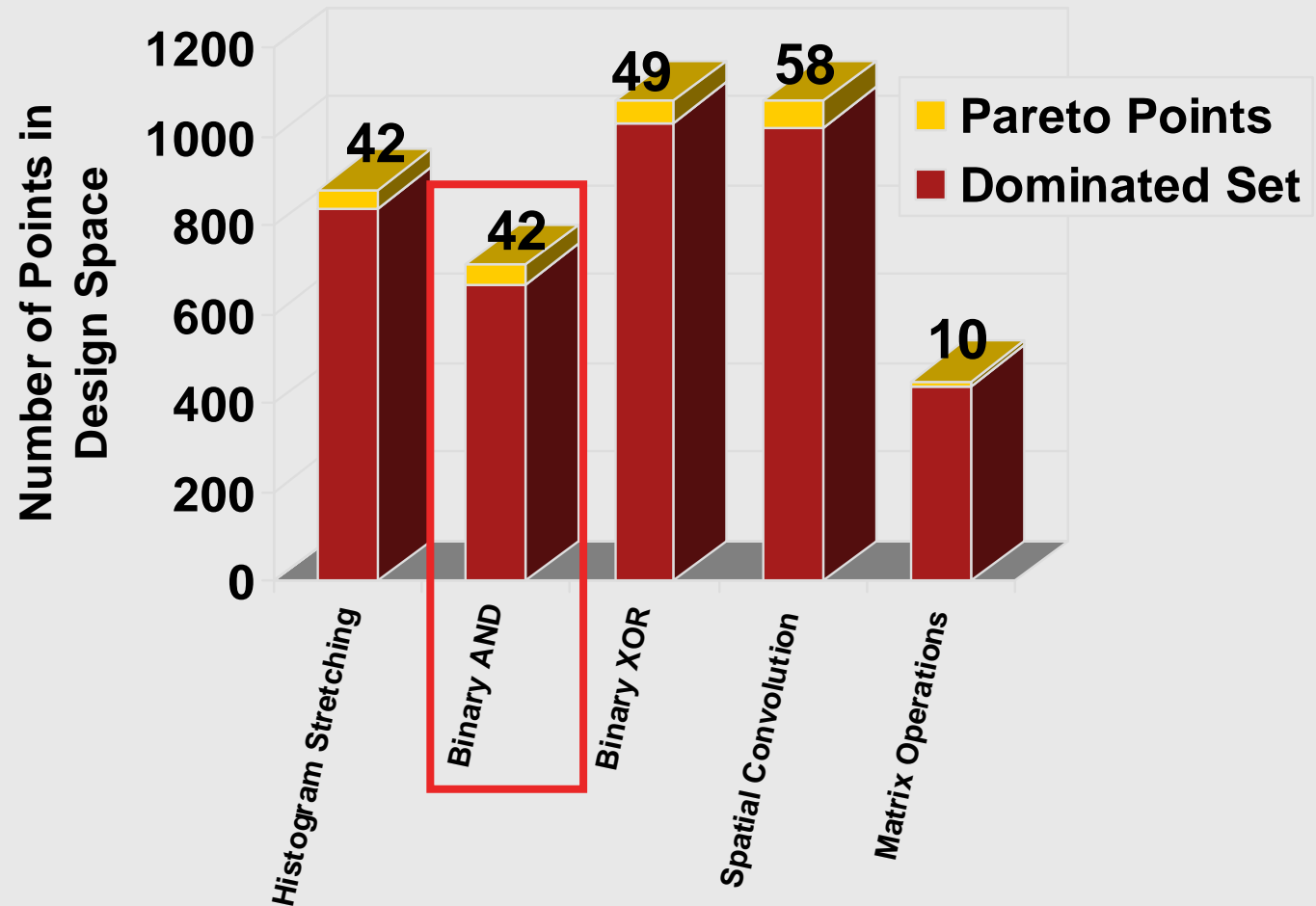


Image-Processing Domain Design Space

Application	Transformations	number PEs	Design space analysis (# pareto points)
Histogram Stretching	<ul style="list-style-type: none"> • Dead Code Elimination • Tree height reduction 	CRC Model	882
Binary AND und Binary XOR	<ul style="list-style-type: none"> • Loop Unrolling (2x – 16x) • Common Sub-Expression Elimination 		711 (AND) 1080 (XOR)
Spatial Convolution 3x3 und 5x5	<ul style="list-style-type: none"> • Constant Folding <ul style="list-style-type: none"> • Copy Propagation 	2–19 PEs	1080
Matrix Operations	<ul style="list-style-type: none"> • Constant Propagation 		440



Image-Processing Domain Design Space





Conclusions



- **The benefits of reconfiguration can only be utilized if applications can be mapped efficiently:**
 - **C code must be partitioned temporally by an automated high-level compiler**
 - **the target architecture must provide the resource to execute the application efficiently**
- **Ray casting example shows considerable advantages compared to FPGAs**
- **We proposed a voltage scaling technique which satisfies the timing constraints without frequency scaling**
- **Design of Processor-Like Architectures for an Application Domain is challenging:**
 - **Resource allocation should attend evenly all applications in domain.**
 - **Set of mapping strategy should foresee appropriate use of resources for all applications.**