Imperial College London

Exploiting Run-time Reconfiguration In Stencil Computation

Xinyu Niu, ***Qiwei Jin, Wayne Luk, Qiang Liu and Oliver Pell***

Dept. of Computing, School of Engineering, Imperial College London, UK
School of Electronic Information Engineering, Tianjin University, China
Maxeler Technologies, UK

Overview

Background: run-time reconfiguration

- Purna and Bhatia 1999
  - temporal partitioning and data flow graph scheduling
- Singhal and Bozorgzadeh 2006
  - floorplanning for reconfiguration
- Bruneel, Abouelella and Stroobandt 2009
  - mapping applications to self-reconfiguring platform
- Iskander 2010
  - accelerate design validation
- Koch and Torresen 2011
  - run-time reconfigurable sorter for large-scale problems

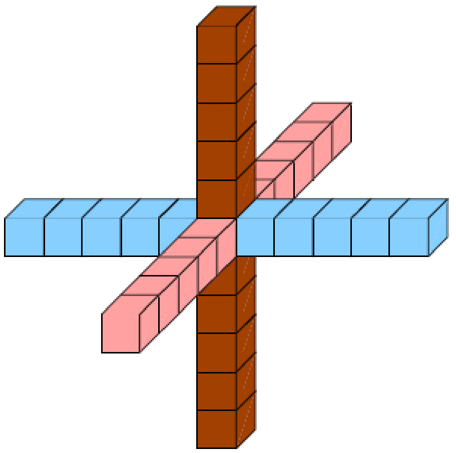
Background: run-time reconfiguration

- Purna and Bhatia 1999
  - temporal partitioning and data flow graph scheduling
- Singhal and Bozorgzadeh 2006
  - floorplanning for reconfiguration
- Bruneel, Abouelella and Stroobandt 2009
  - mapping applications to self-reconfiguring platform
- Iskander 2010
  - accelerate design validation
- Koch and Torresen 2011
  - run-time reconfigurable sorter for large-scale problems
- Our focus: automate reconfigurable stencil computation

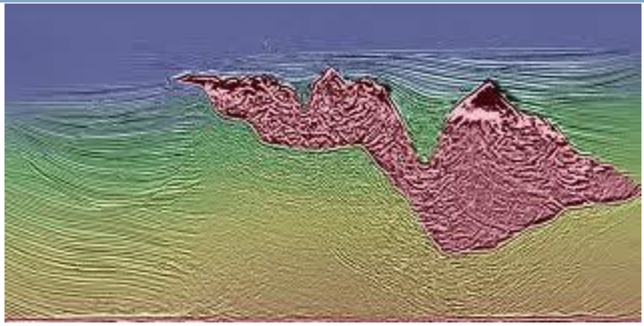
Contributions

1. identify reconfiguration opportunities
   - partitioning algorithm: generate configurations based on variations in time dimension
2. analyse runtime benefits for stencil computation
   - analytical model: optimise generated partitions, to fully utilise available resources
3. evaluate run-time solutions
   - Scheduling algorithm: evaluate run-time benefits and overhead, to provide optimal run-time solution

Stencil Computation: Reverse Time Migration (RTM)



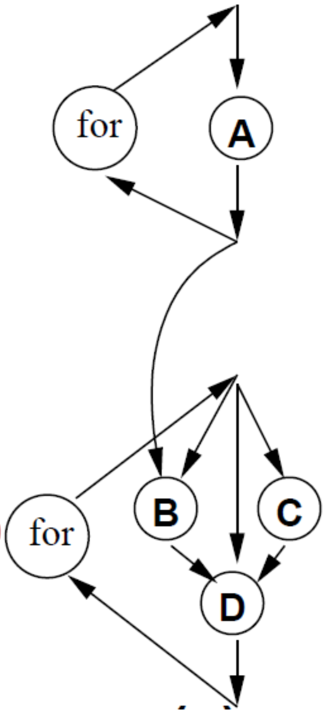
$$\frac{\partial^2 p}{\partial t^2} = v^2 \nabla^2 p + s(t)$$



Example: Oil & Gas Application

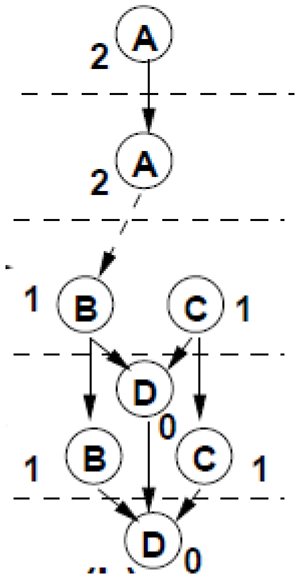
```
for t = 1 to tmax:
   for i = 0 to X*Y*Z-1:
      l = convolve(curr[i], stencil)
      next := 2 * curr[i] - prev[i] + vv[i] * l
      next[i] := next + source[i]
      apply_boundary_condition(curr, next)
      swap_buffers(prev, curr, next)
```

$$\begin{tikzcd}
 & \boxed{\text{Forward propagation}} \arrow[dr] & \\
\boxed{\text{Forward propagation}} \arrow[ur] \arrow[dr] & & \text{correlation} \\
 & \boxed{\text{Backward propagation}} \arrow[ur] &
\end{tikzcd}$$



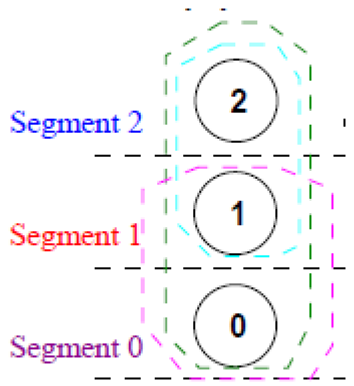
1. Partitioning algorithm: functions to segments

- segments
  - functions in same data dependency level combined as a segment
  - segment variations express algorithm variations in time dimension



Partitioning algorithm: segments to configurations

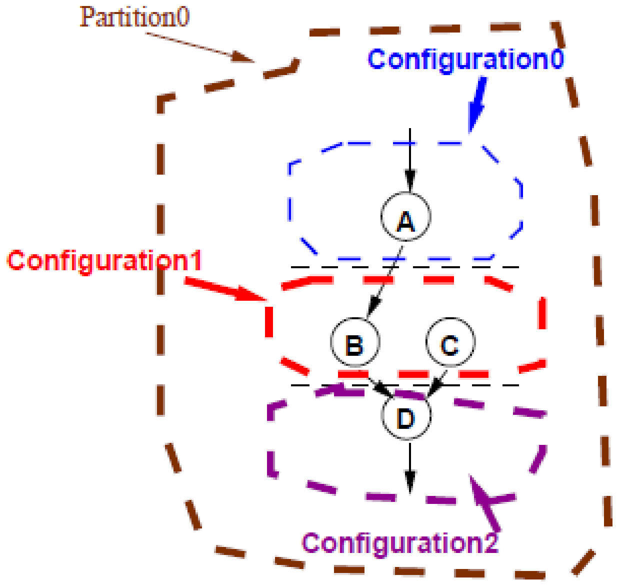
- segments
  - functions in same data dependency level cor
    segment
  - segment variations express algorithm variati
    dimension



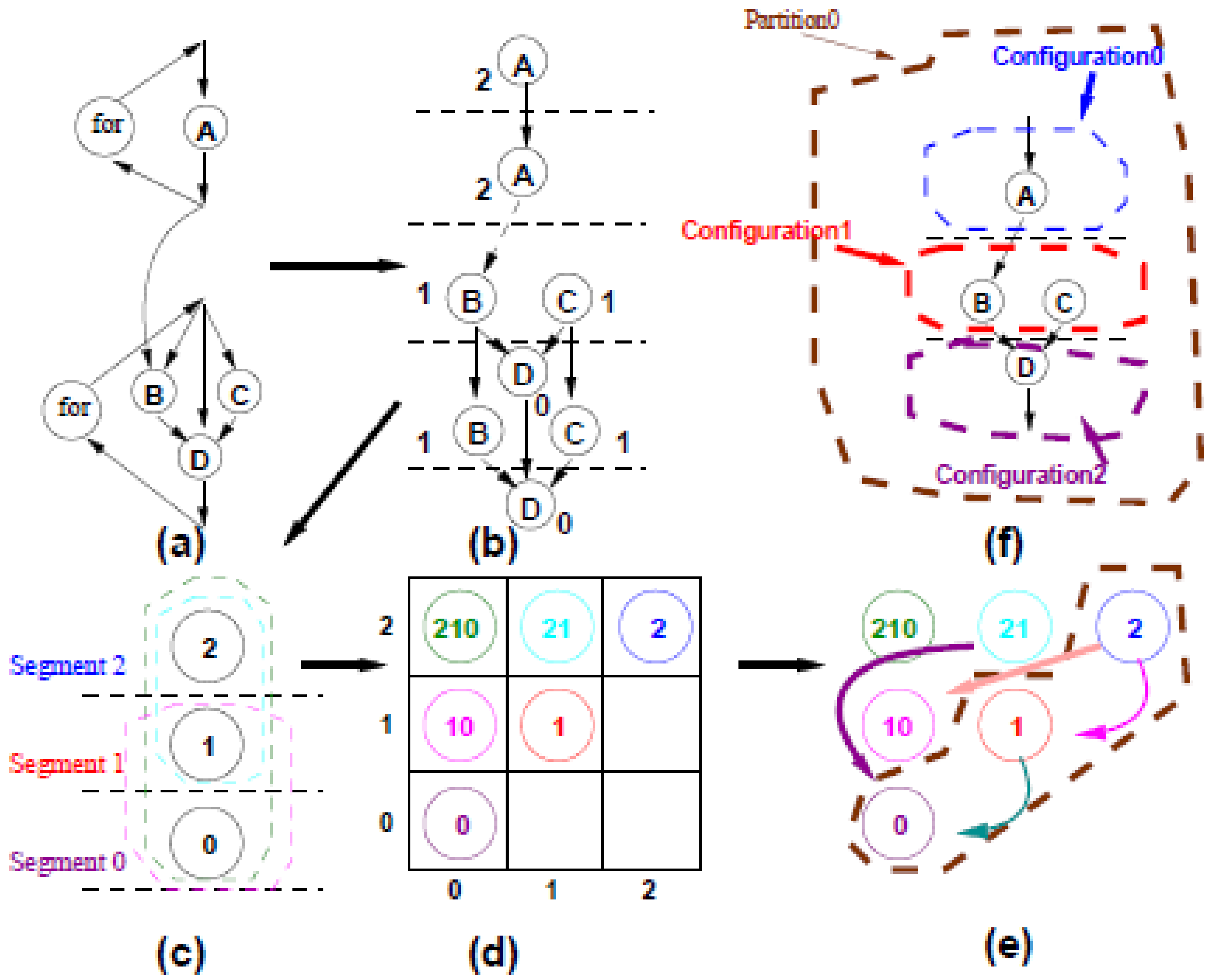
- configurations
  - one or more segments are combined as a valid configuration to be executed
  - optimised configurations utilise run-time benefits

Partitioning algorithm: partitions

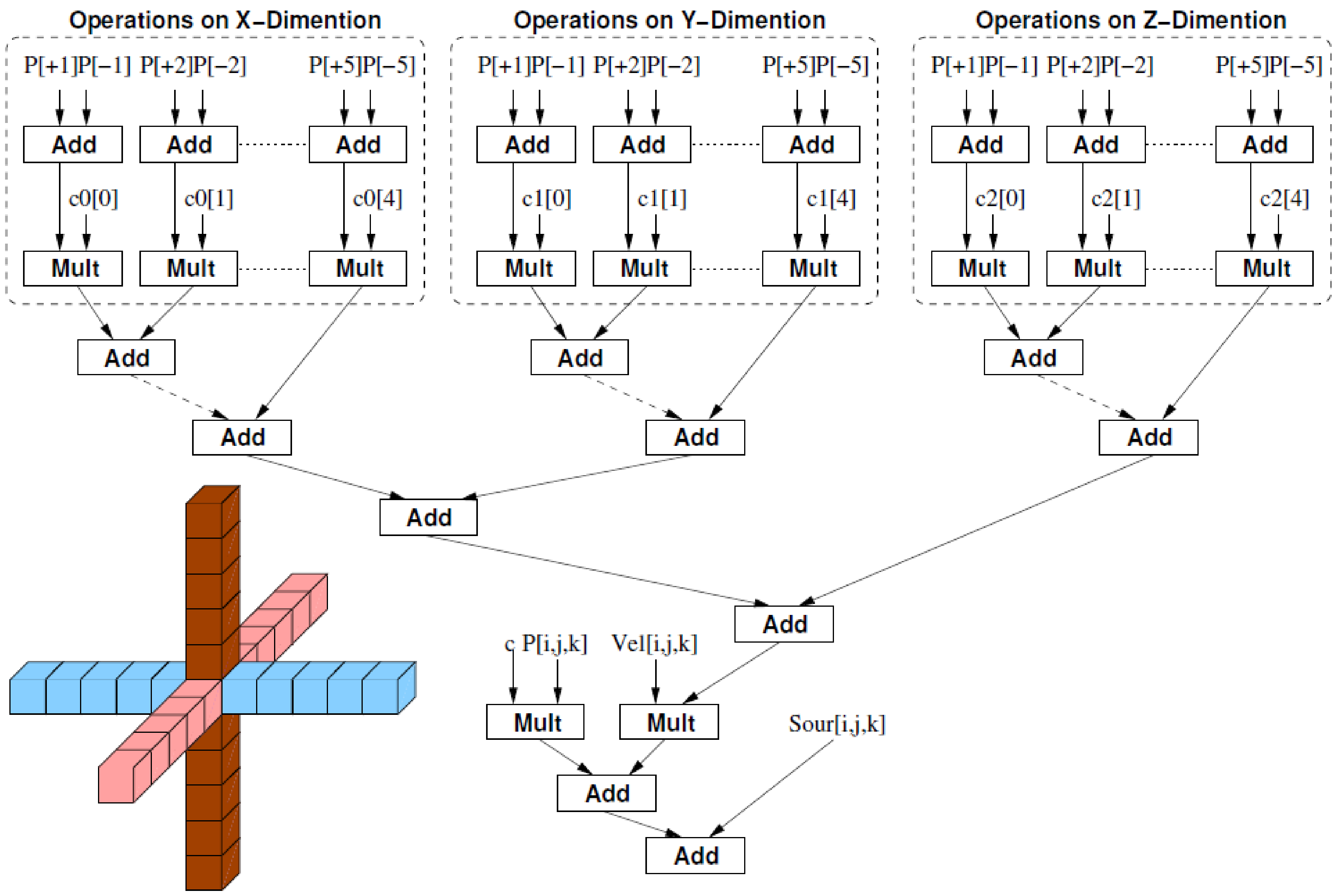
- segments
  - functions in same data dependency level combined as a segment
  - segment variations express algorith dimension
- configurations
  - one or more segments are combine configuration to be executed
  - optimised configurations utilise run-
- partitions
  - one or more configurations are coordinated to accomplish application tasks
  - optimal partition balances run-time benefits and overhead



Dynamic partitioning: algorithm

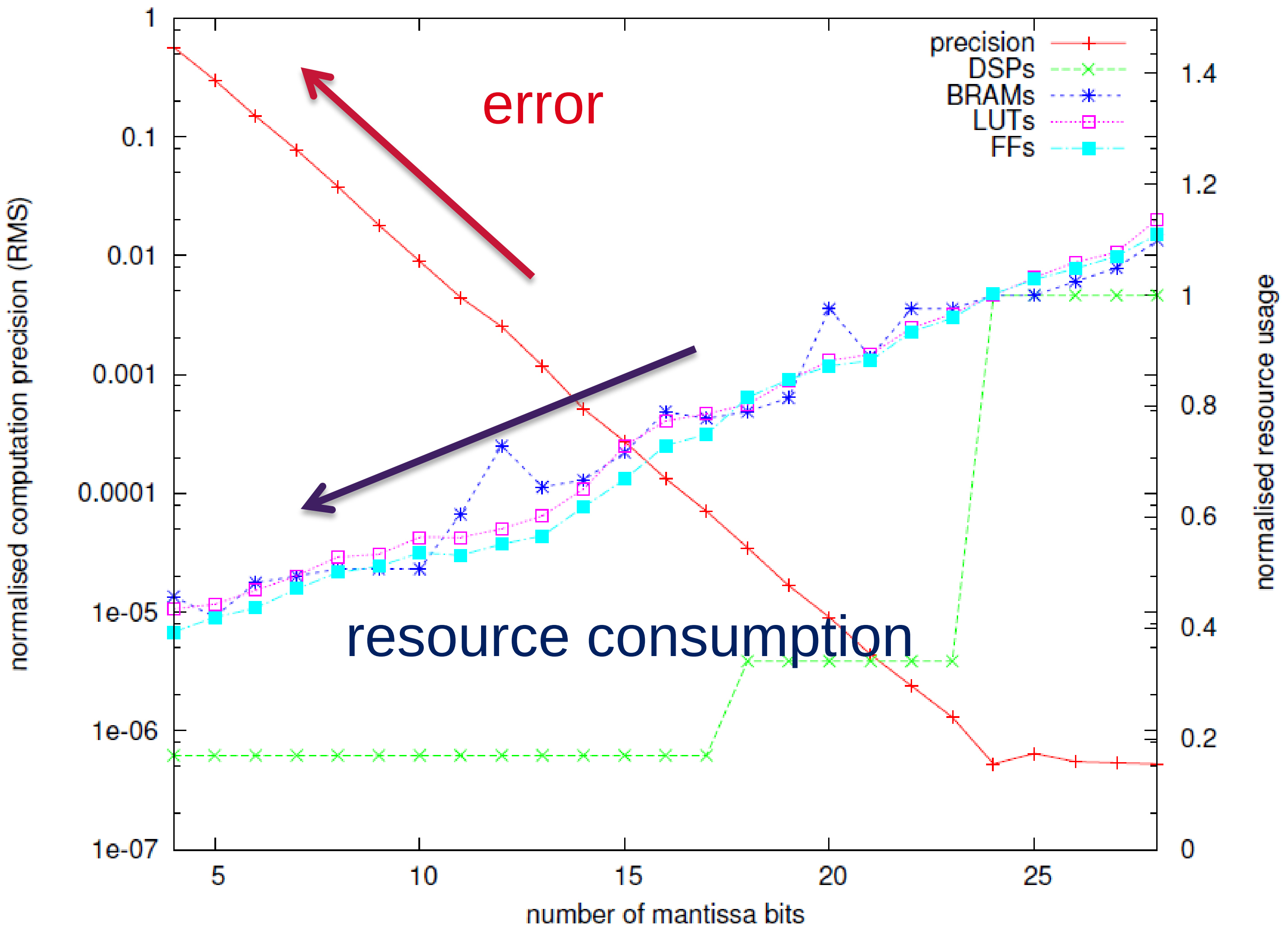


2. Analytical model: data-paths


Operations on X–Dimention
Operations on Y–Dimention
Operations on Z–Dimention
P[+1]P[−1] P[+2]P[−2] P[+5]P[−5]
Add
Mult
c0[0] c0[1] c0[4]
c1[0] c1[1] c1[4]
c2[0] c2[1] c2[4]
c P[i,j,k]
Vel[i,j,k]
Sour[i,j,k]

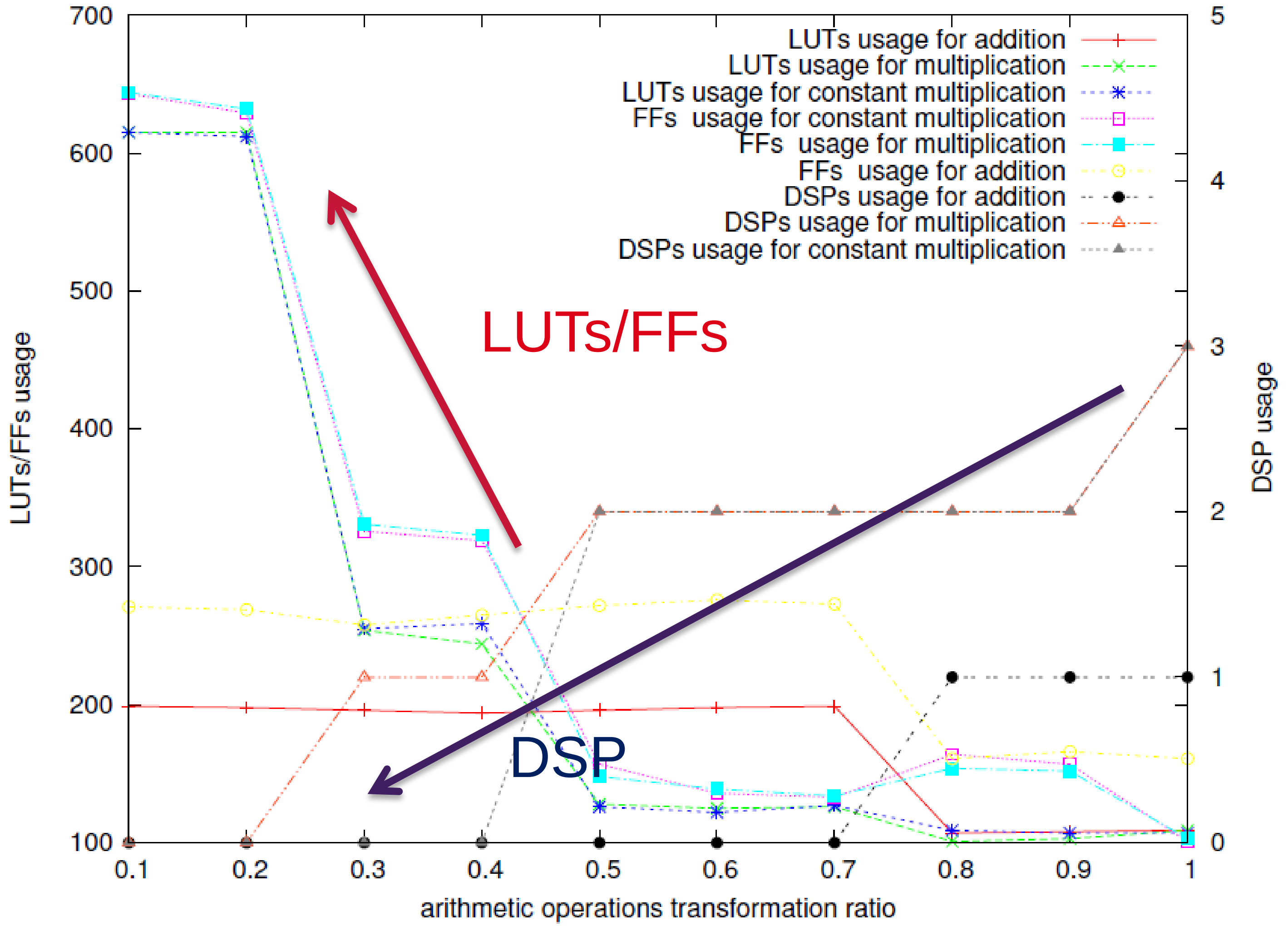
Analytical model: error and resource consumption

- fully pipelined data-paths
- bit-width optimisation, arithmetic operation transform



Analytical model: FPGA resources

- fully pipelined data-paths
- bit-width optimisation, arithmetic operation transform



Analytical model: FPGA resource consumption

- fully pipelined data-paths
- bit-width optimisation and arithmetic operation transformation
- accumulating resource consumption

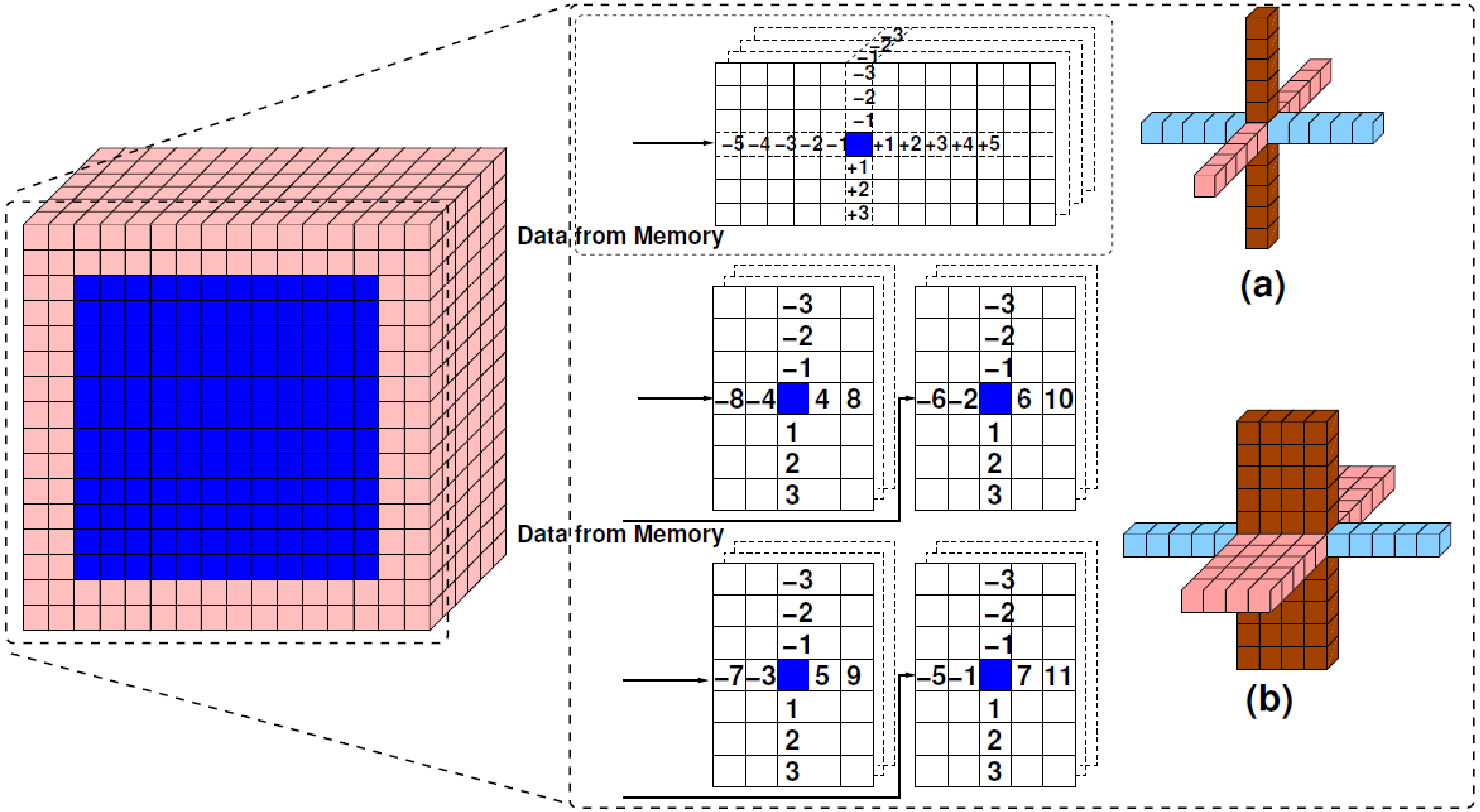
$$Ds = \frac{\sum N_{\mathrm{op,i}} \cdot B_{\mathrm{D}} \cdot T_{\mathrm{D,i}}}{A_{\mathrm{D}}} \longleftarrow \text{DSP consumption}$$

$$Ls = \frac{\sum (N_{\mathrm{op,i}} \cdot B_{\mathrm{L}} \cdot T_{\mathrm{L,i}}) + I_{\mathrm{L}}}{A_{\mathrm{L}}} \longleftarrow \text{LUT consumption}$$

$$Fs = \frac{\sum (N_{\mathrm{op,i}} \cdot B_{\mathrm{F}} \cdot T_{\mathrm{F,i}}) + I_{\mathrm{F}}}{A_{\mathrm{F}}} \longleftarrow \text{FF consumption}$$

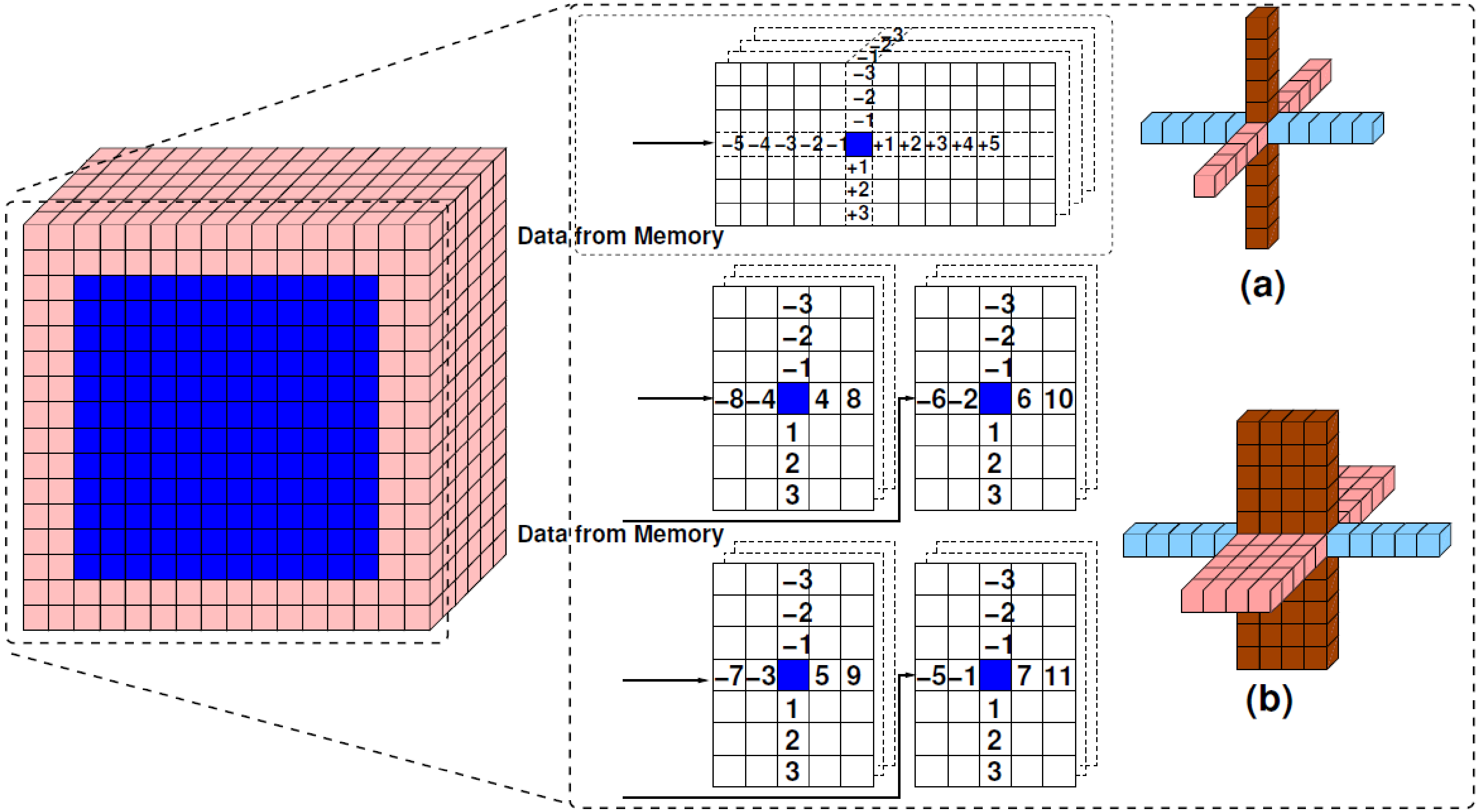
Analytical model: memory systems

- customised memory architecture



Analytical Model: Memory Systems

- customised memory architecture
- data access blocking



Analytical model: memory systems

- customised memory architecture
- analysing resource / bandwidth requirements

memory resource consumption

$$Bs = \frac{\sum_{1}^{P_{\mathrm{knl}} \cdot P_{\mathrm{t}} \cdot P_{\mathrm{dp}}} m_{\mathrm{dp}} \cdot (S \cdot (2 + N_{\mathrm{c}}) + 1)}{A_{\mathrm{B}} / \left(W_{\mathrm{dp}} \cdot B_{\mathrm{w}}\right)}$$

$$m_{\mathrm{dp}} = \frac{nx + (P_{\mathrm{t}} - 1) \cdot 2S}{P_{\mathrm{dp}}} \cdot (ny + (P_{\mathrm{t}} - 1) \cdot 2S)$$

Analytical model: memory systems

- customised memory architecture
- analysing resource / bandwidth requirements

$$Bs = \frac{\sum_{1}^{P_{\mathrm{knl}} \cdot P_{\mathrm{t}} \cdot P_{\mathrm{dp}}} m_{\mathrm{dp}} \cdot (S \cdot (2 + N_{\mathrm{c}}) + 1)}{A_{\mathrm{B}} / (W_{\mathrm{dp}} \cdot B_{\mathrm{w}})}$$

$$m_{\mathrm{dp}} = \frac{nx + (P_{\mathrm{t}} - 1) \cdot 2S}{P_{\mathrm{dp}}} \cdot (ny + (P_{\mathrm{t}} - 1) \cdot 2S)$$

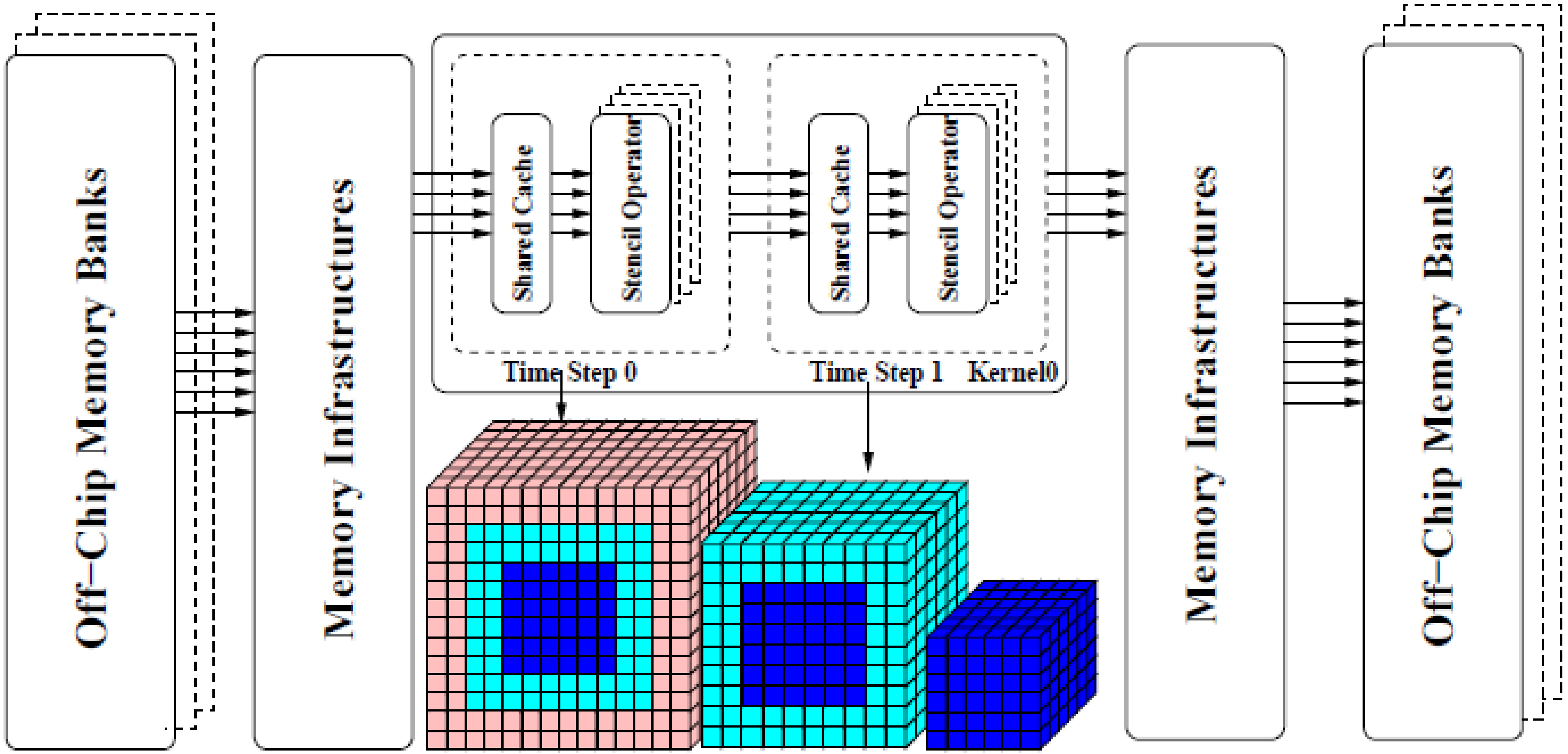
off-chip bandwidth requirement

$$BW_{\mathrm{m}} \geq (W_{\mathrm{dp}} \cdot B_{\mathrm{w}} \cdot P_{\mathrm{dp}}) \cdot P_{\mathrm{knl}} \cdot f_{\mathrm{knl}}$$

$$W_{\mathrm{m}} = N \cdot (W_{\mathrm{dp}} \cdot B_{\mathrm{w}} \cdot P_{\mathrm{dp}}) \quad N \in \{1, 2, 3 \ldots\}$$

Analytical model: design scalability

- parallel data-paths: multiple memory access
- serial: multiple time steps



Analytical model: optimisation

- Objective

Minimise: $C_\mathrm{t} = R_\mathrm{c/c} \cdot \frac{D \cdot O_\mathrm{b} \cdot O_\mathrm{t}}{f_\mathrm{knl} \cdot P_\mathrm{knl} \cdot P_\mathrm{dp} \cdot P_\mathrm{t}}$

$$O_b = \frac{\frac{x-2\cdot S}{nx-2\cdot S} \cdot \frac{y-2\cdot S}{ny-2\cdot S} \cdot nx \cdot ny}{x \cdot y}$$

$$nx = \frac{x - 2 \cdot S}{\alpha} + 2 \cdot S \quad ny = \frac{y - 2 \cdot S}{\beta} + 2 \cdot S$$

$$O_t = \begin{cases} 1 & \alpha \cdot \beta = 1 \\ \frac{nx+(P_\mathrm{t}-1)\cdot 2\cdot S}{nx} \cdot \frac{ny+(P_\mathrm{t}-1)\cdot 2\cdot S}{ny} & \alpha \cdot \beta \neq 1 \end{cases}$$

Analytical model: optimisation

- Objective
- Constraints

Minimise: $C_t = R_{c/c} \cdot \frac{D \cdot O_b \cdot O_t}{f_{knl} \cdot P_{knl} \cdot P_{dp} \cdot P_t}$

Subject to:

$$1 \geq Ds = \frac{\sum N_{op,i} \cdot B_D \cdot T_{D,i}}{A_D}$$

$$1 \geq Ls = \frac{\sum (N_{op,i} \cdot B_L \cdot T_{L,i}) + I_L}{A_L}$$

$$1 \geq Fs = \frac{\sum (N_{op,i} \cdot B_F \cdot T_{F,i}) + I_F}{A_F}$$

$$1 \geq Bs = \frac{\sum_1^{P_{knl} \cdot P_t \cdot P_{dp}} m_{dp} \cdot (S \cdot (2 + N_c) + 1)}{A_B / (W_{dp} \cdot B_w)}$$

$$BW_m \geq (W_{dp} \cdot B_w \cdot P_{dp}) \cdot P_{knl} \cdot f_{knl}$$

$$O_b = \frac{\frac{x-2 \cdot S}{nx-2 \cdot S} \cdot \frac{y-2 \cdot S}{ny-2 \cdot S} \cdot nx \cdot ny}{x \cdot y}$$

$$nx = \frac{x - 2 \cdot S}{\alpha} + 2 \cdot S \quad ny = \frac{y - 2 \cdot S}{\beta} + 2 \cdot S$$

$$O_t = \begin{cases} 1 & \alpha \cdot \beta = 1 \\ \frac{nx+(P_t-1) \cdot 2 \cdot S}{nx} \cdot \frac{ny+(P_t-1) \cdot 2 \cdot S}{ny} & \alpha \cdot \beta \neq 1 \end{cases}$$

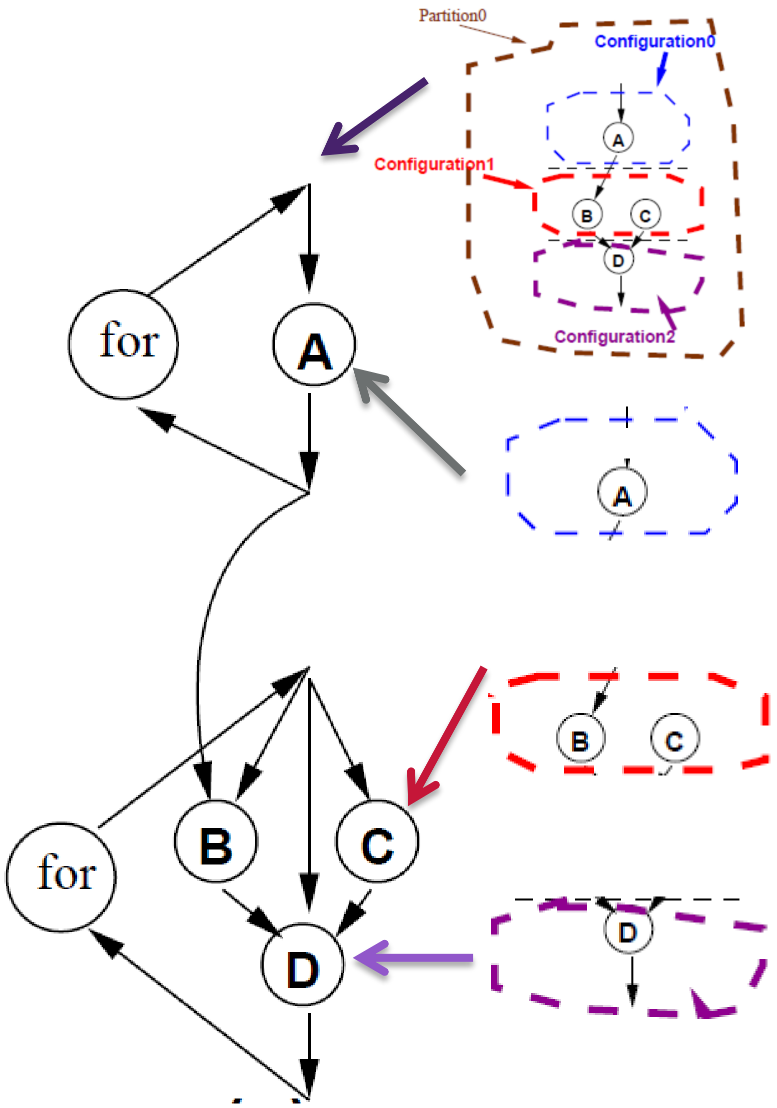
Analytical model: optimisation

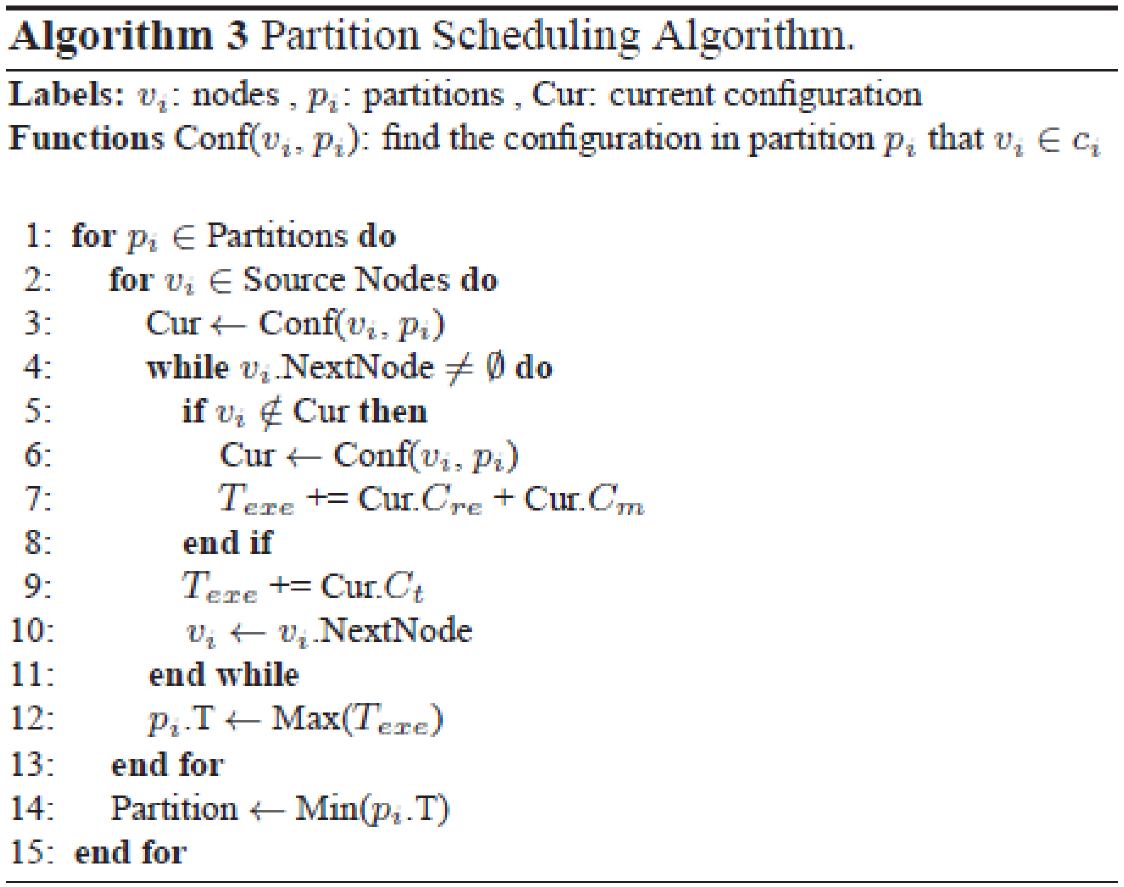
- Objective
- Constraints
- Overhead
  - reconfiguration time
  - data transfer time

$$C_{\mathrm{re}} = \frac{\gamma \cdot Max(B_{\mathrm{s}}, D_{\mathrm{s}}, L_{\mathrm{s}}, F_{\mathrm{s}})}{\theta}$$

$$C_{\mathrm{m}} = \frac{2 \cdot D \cdot W_{\mathrm{dp}} \cdot (1 + N_{\mathrm{c}}) \cdot B_{\mathrm{w}}}{\phi}$$

3. Reconfiguration scheduling





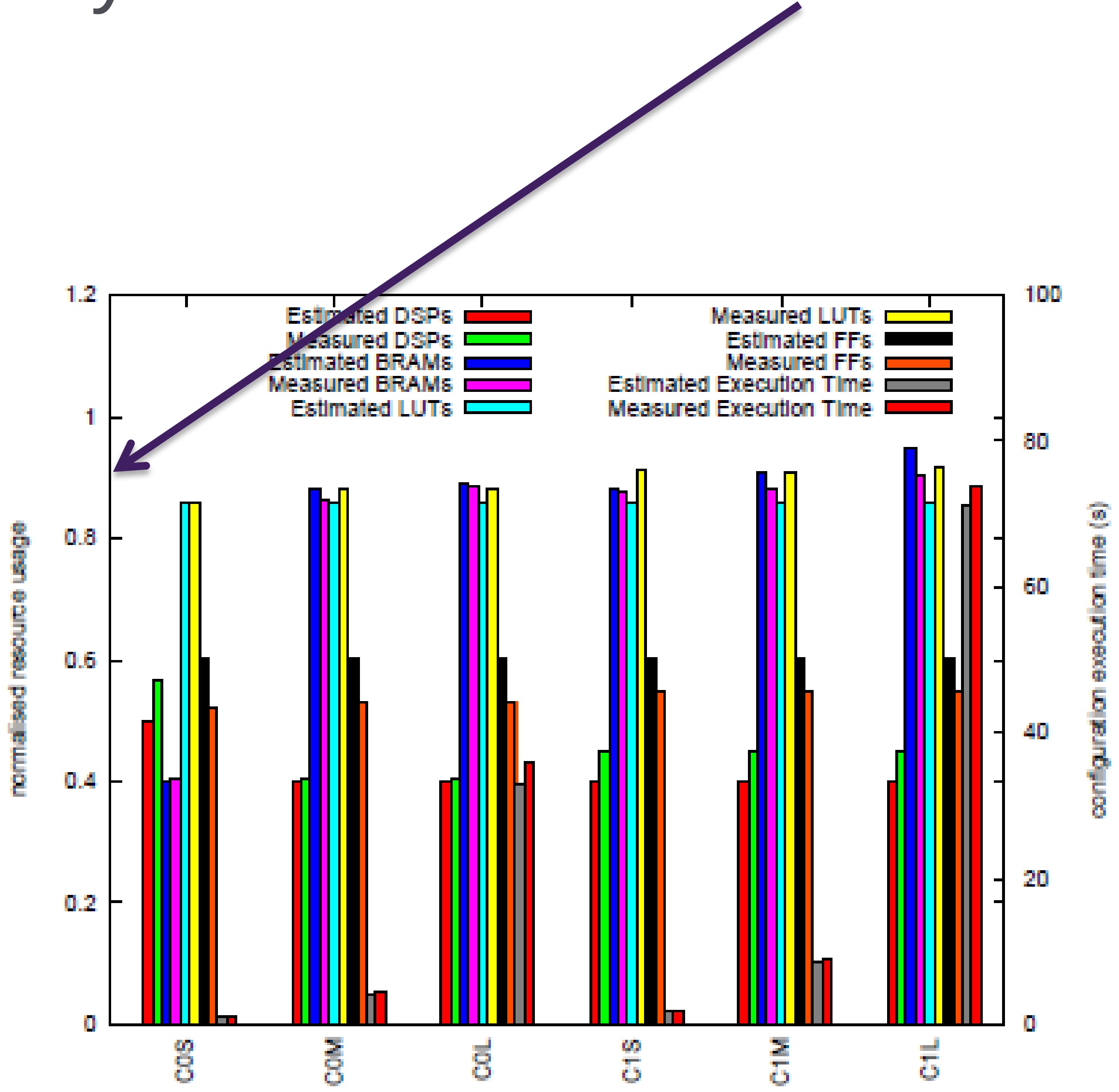
Algorithm 3 Partition Scheduling Algorithm.

Labels: v_i: nodes , p_i: partitions , Cur: current configuration
Functions Conf(v_i, p_i): find the configuration in partition p_i that $v_i \in c_i$

```
1:  for $p_i \in$ Partitions do
2:      for $v_i \in$ Source Nodes do
3:          Cur ← Conf($v_i$, $p_i$)
4:          while $v_i$.NextNode $\neq \emptyset$ do
5:              if $v_i \notin$ Cur then
6:                  Cur ← Conf($v_i$, $p_i$)
7:                  $T_{exe}$ += Cur.$C_{re}$ + Cur.$C_m$
8:              end if
9:              $T_{exe}$ += Cur.$C_t$
10:             $v_i$ ← $v_i$.NextNode
11:         end while
12:         $p_i$.T ← Max($T_{exe}$)
13:     end for
14:     Partition ← Min($p_i$.T)
15: end for
```

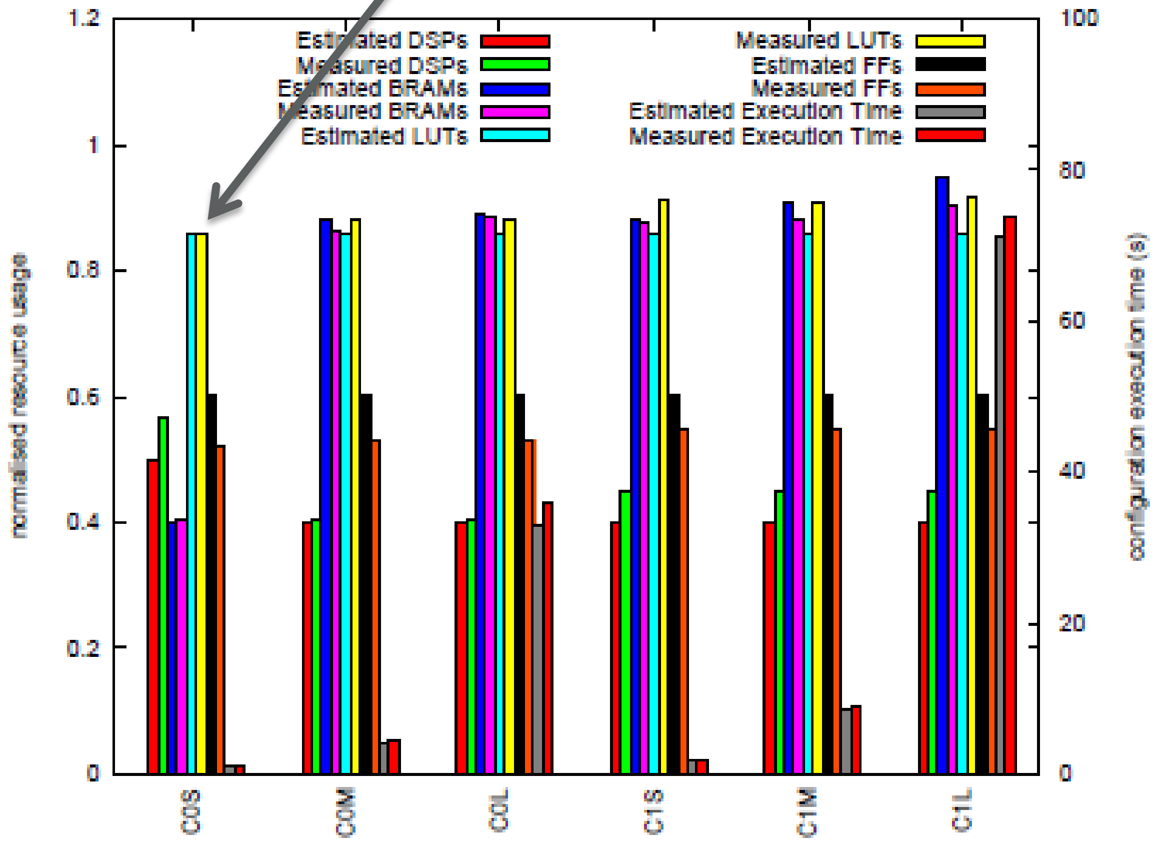
Results: model performance

- fully utilise available resources



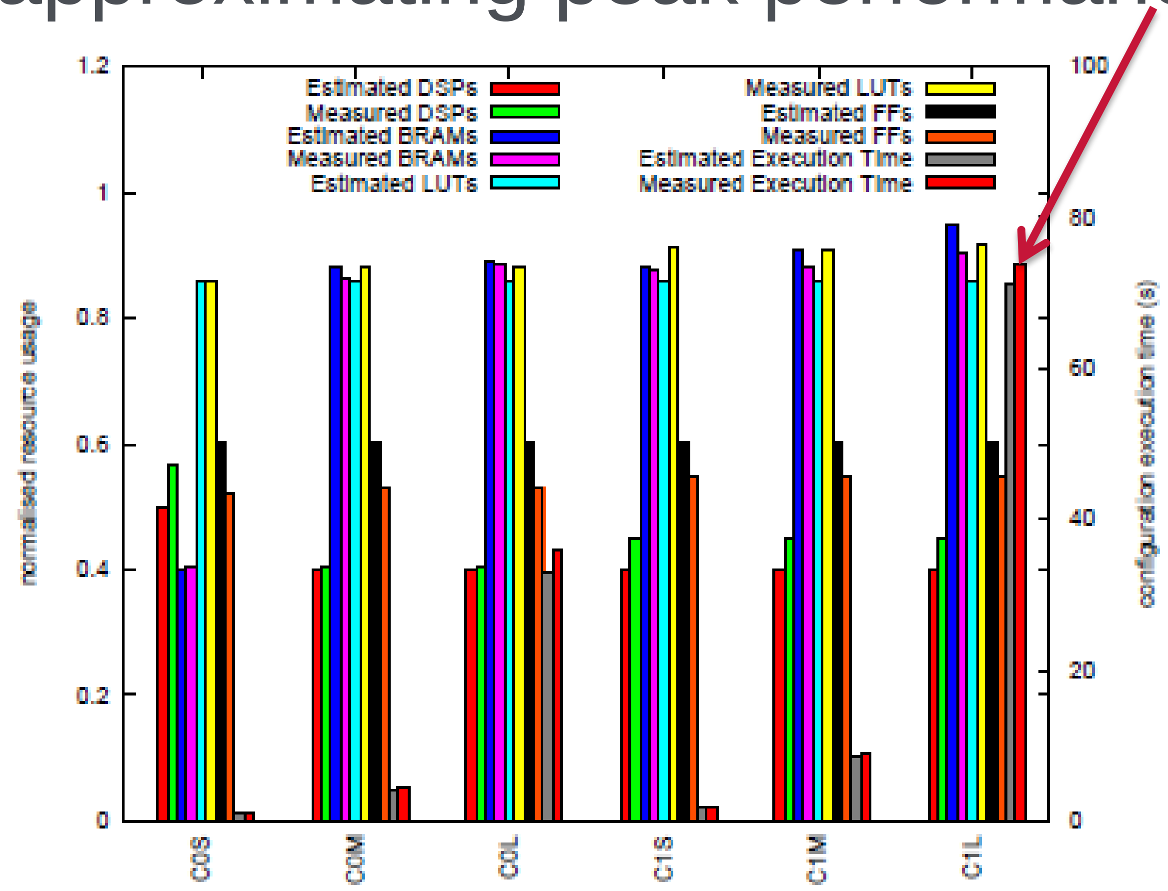
Results: model performance

- fully utilise available resources
- model accuracy

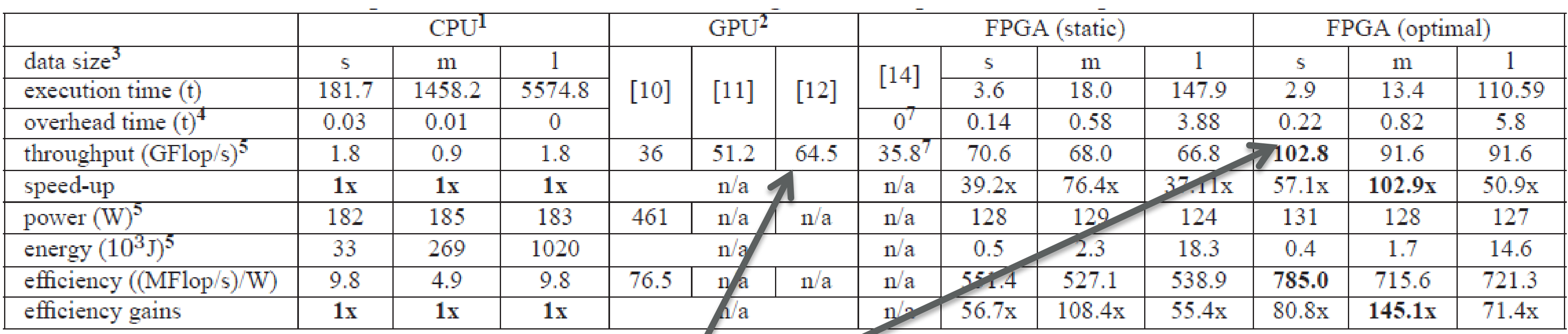


Results: model performance

- fully utilise available resources
- model accuracy
- approximating peak performance



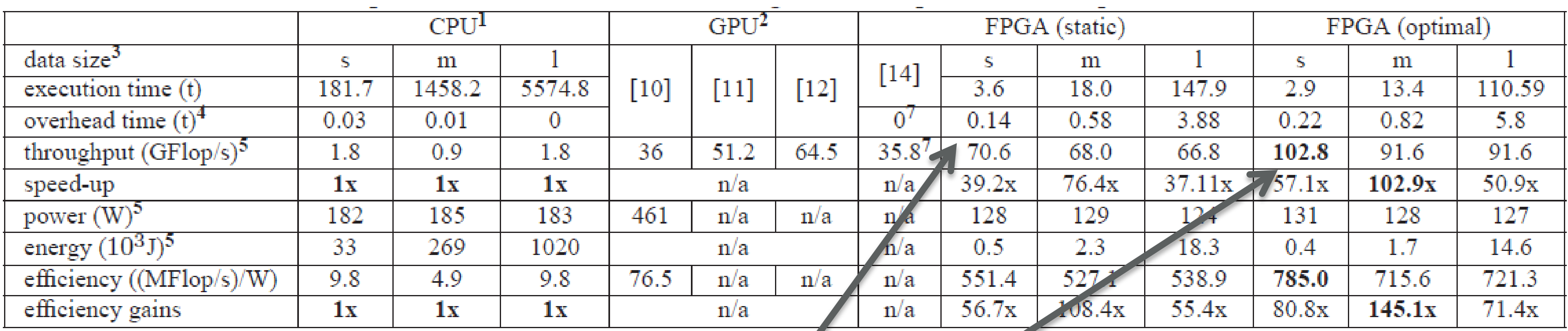
Results: optimal reconfiguration solution



	CPU[1]			GPU[2]			FPGA (static)				FPGA (optimal)		
data size[3]	s	m	l	[10]	[11]	[12]	[14]	s	m	l	s	m	l
execution time (t)	181.7	1458.2	5574.8					3.6	18.0	147.9	2.9	13.4	110.59
overhead time (t)[4]	0.03	0.01	0				0[7]	0.14	0.58	3.88	0.22	0.82	5.8
throughput (GFlop/s)[5]	1.8	0.9	1.8	36	51.2	64.5	35.8[7]	70.6	68.0	66.8	**102.8**	91.6	91.6
speed-up	**1x**	**1x**	**1x**	n/a			n/a	39.2x	76.4x	37.11x	57.1x	**102.9x**	50.9x
power (W)[5]	182	185	183	461	n/a	n/a	n/a	128	129	124	131	128	127
energy (10^3J)[5]	33	269	1020	n/a			n/a	0.5	2.3	18.3	0.4	1.7	14.6
efficiency ((MFlop/s)/W)	9.8	4.9	9.8	76.5	n/a	n/a	n/a	551.4	527.1	538.9	**785.0**	715.6	721.3
efficiency gains	**1x**	**1x**	**1x**	n/a			n/a	56.7x	108.4x	55.4x	80.8x	**145.1x**	71.4x

- improved system performance
  - 1.59 times faster than the best published GPU and FPGA results

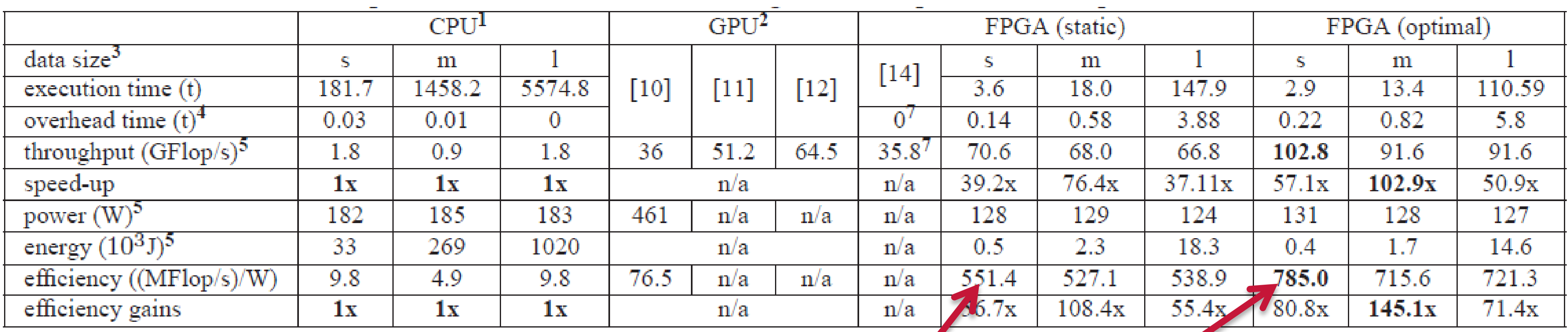
Results: optimal reconfiguration solution



	CPU[1]			GPU[2]			FPGA (static)				FPGA (optimal)		
data size[3]	s	m	l	[10]	[11]	[12]	[14]	s	m	l	s	m	l
execution time (t)	181.7	1458.2	5574.8					3.6	18.0	147.9	2.9	13.4	110.59
overhead time (t)[4]	0.03	0.01	0				0[7]	0.14	0.58	3.88	0.22	0.82	5.8
throughput (GFlop/s)[5]	1.8	0.9	1.8	36	51.2	64.5	35.8[7]	70.6	68.0	66.8	**102.8**	91.6	91.6
speed-up	**1x**	**1x**	**1x**	n/a			n/a	39.2x	76.4x	37.11x	57.1x	**102.9x**	50.9x
power (W)[5]	182	185	183	461	n/a	n/a	n/a	128	129	124	131	128	127
energy (10^3J)[5]	33	269	1020	n/a			n/a	0.5	2.3	18.3	0.4	1.7	14.6
efficiency ((MFlop/s)/W)	9.8	4.9	9.8	76.5	n/a	n/a	n/a	551.4	527.1	538.9	**785.0**	715.6	721.3
efficiency gains	**1x**	**1x**	**1x**	n/a			n/a	56.7x	108.4x	55.4x	80.8x	**145.1x**	71.4x

- improved system performance
  - 1.59 times faster than the best published GPU and FPGA results
  - 1.45 times faster than optimized static implementation

Results: optimal reconfiguration solution



	CPU[1]			GPU[2]			FPGA (static)				FPGA (optimal)		
data size[3]	s	m	l	[10]	[11]	[12]	[14]	s	m	l	s	m	l
execution time (t)	181.7	1458.2	5574.8					3.6	18.0	147.9	2.9	13.4	110.59
overhead time (t)[4]	0.03	0.01	0				0[7]	0.14	0.58	3.88	0.22	0.82	5.8
throughput (GFlop/s)[5]	1.8	0.9	1.8	36	51.2	64.5	35.8[7]	70.6	68.0	66.8	**102.8**	91.6	91.6
speed-up	**1x**	**1x**	**1x**	n/a			n/a	39.2x	76.4x	37.11x	57.1x	**102.9x**	50.9x
power (W)[5]	182	185	183	461	n/a	n/a	n/a	128	129	124	131	128	127
energy (10^3J)[5]	33	269	1020	n/a			n/a	0.5	2.3	18.3	0.4	1.7	14.6
efficiency ((MFlop/s)/W)	9.8	4.9	9.8	76.5	n/a	n/a	n/a	551.4	527.1	538.9	**785.0**	715.6	721.3
efficiency gains	**1x**	**1x**	**1x**	n/a			n/a	56.7x	108.4x	55.4x	80.8x	**145.1x**	71.4x

- improved system performance
  - 1.59 times faster than the best published GPU and FPGA results
  - 1.45 times faster than optimized static implementation
  - 1.42 times more energy efficient than the static designs

Future work

- generalise design methods
- partial reconfiguration
- run-time scheduling
- further applications

Conclusion

- design method to exploit run-time reconfiguration
  - partitioner
  - analytical model
  - scheduler
- improved system performance
  - 1.59 times faster than the best published GPU and FPGA results
  - 1.45 times faster than optimized static implementation
  - 1.42 times more energy efficient than the static designs