FernUniversität in Hagen

Combining Design Space Exploration with Task Scheduling of Moldable Streaming Tasks on Reconfigurable Platforms

Jörg Keller

Joint work with S. Litzinger (FernUni), C. Kessler (Linköping Univ.)

International Symposium on Applied Reconfigurable Computing (ARC2021)

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
Prof. Dr. Jörg Keller

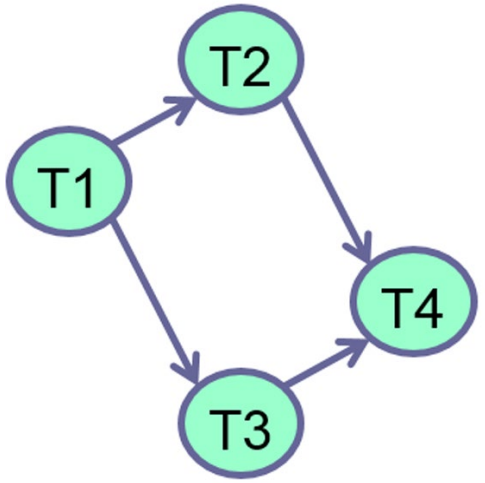
FernUniversität in Hagen

Overview

- Motivation
- Scheduling Moldable Streaming Tasks
- Scheduling for Reconfigurable Platforms
- Experiments
- Conclusions

Motivation

- Streaming applications often used in embedded and edge computing

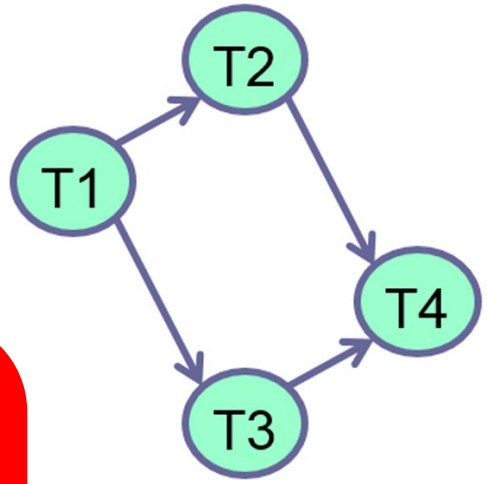


- Platform determines energy consumption for required throughput or vice versa

- Multicore SoC: number of each type of core given ➔ scheduling
  Reconfigurable platform: choose number for each type of cores
  ➔ design space exploration
  ➔ followed by scheduling

Motivation

- **Streaming applications often used in embedded and edge computing**



- **Platform dete… for required …**

This paper:

Combine
design space exploration
and scheduling

- **Multicore So… Reconfigurab… …scheduling …of cores**
  ➔ design spac… …
  ➔ followed by scheduling

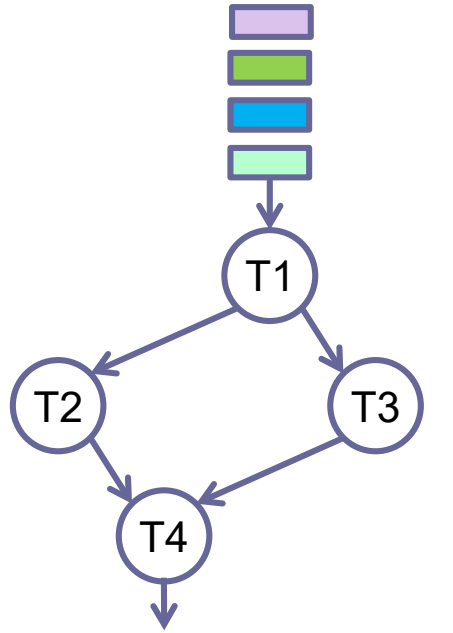
Software Model:
Streaming Computations

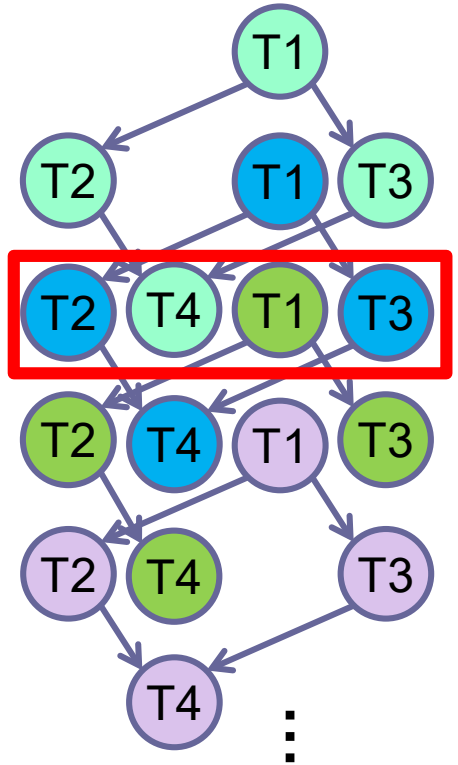
Streaming task graph (= actor network)

- (Acyclic) pipeline graph of streaming tasks
- Producer-consumer communication
- Cf. Kahn Process Networks

Software pipelining

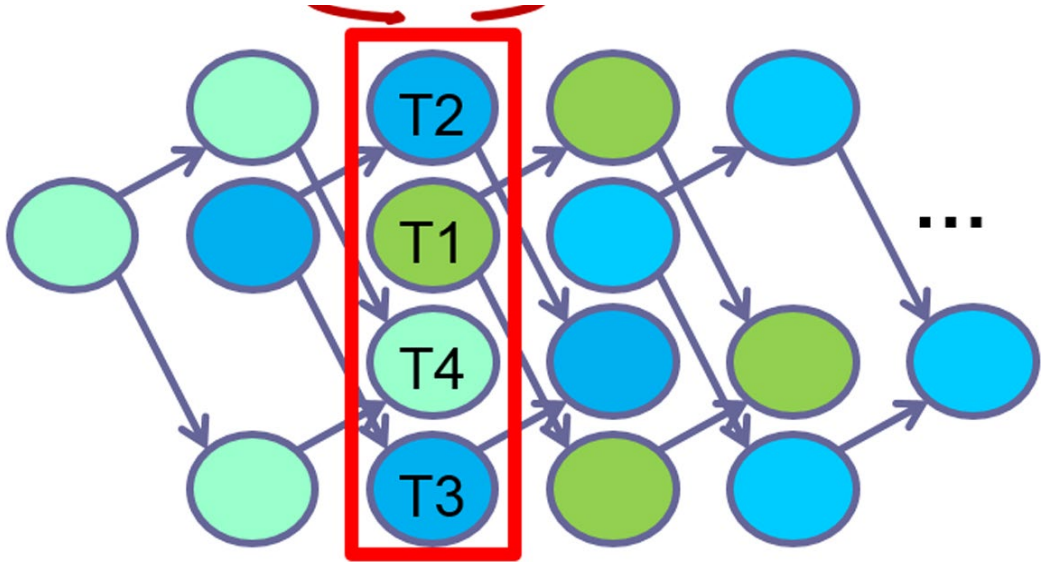
- Steady state



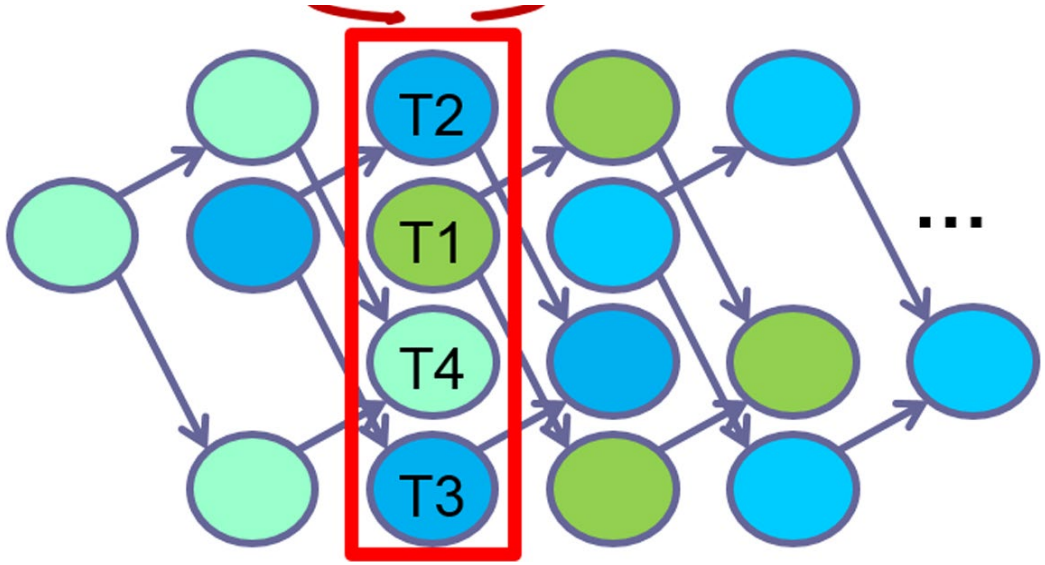


Scheduling Moldable Streaming Tasks I

- Throughput determines length of *round*
- Tasks in one round independent
  Task workloads known
- Tasks can be parallel, speedup function known
  moldable=degree of parallelism fixed before execution

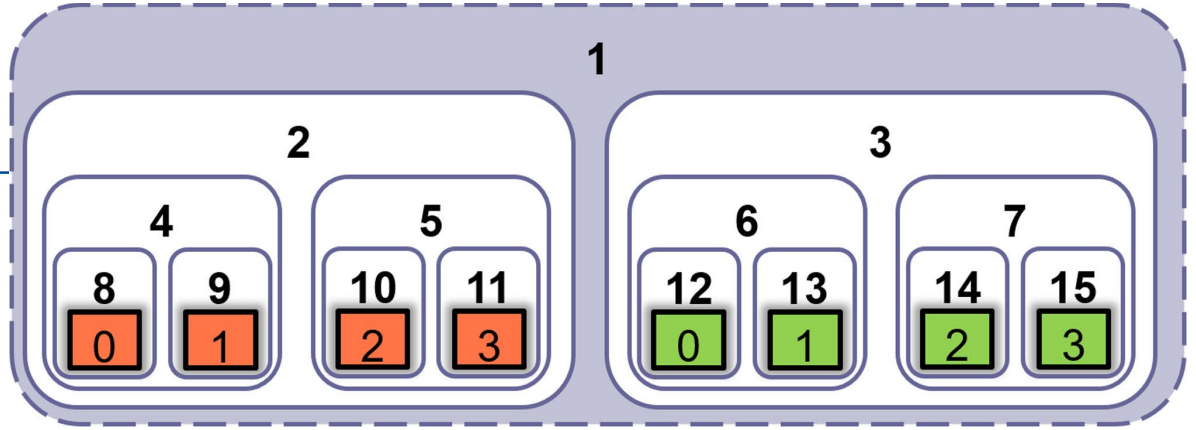


Scheduling Moldable Streaming Tasks II



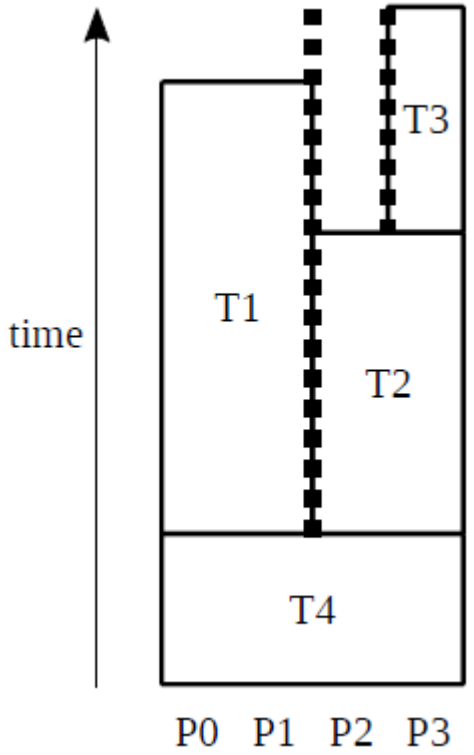
- Cores can be frequency-scaled at discrete frequency levels
- Tasks' power-profiles known
- Scheduling targets minimization of energy consumption in round
- Comprises Allocation, Assignment, Scaling, Ordering

Crown Scheduling



- **Restrict**
  Allocation to powers of 2
  Assignment to core groups
  Order to decreasing width

- **Implement as integer linear program**



Crown Scheduling ILP

Binary variables $x_{i,j,k} = 1$ iff task j is mapped to core group i at freq f_k

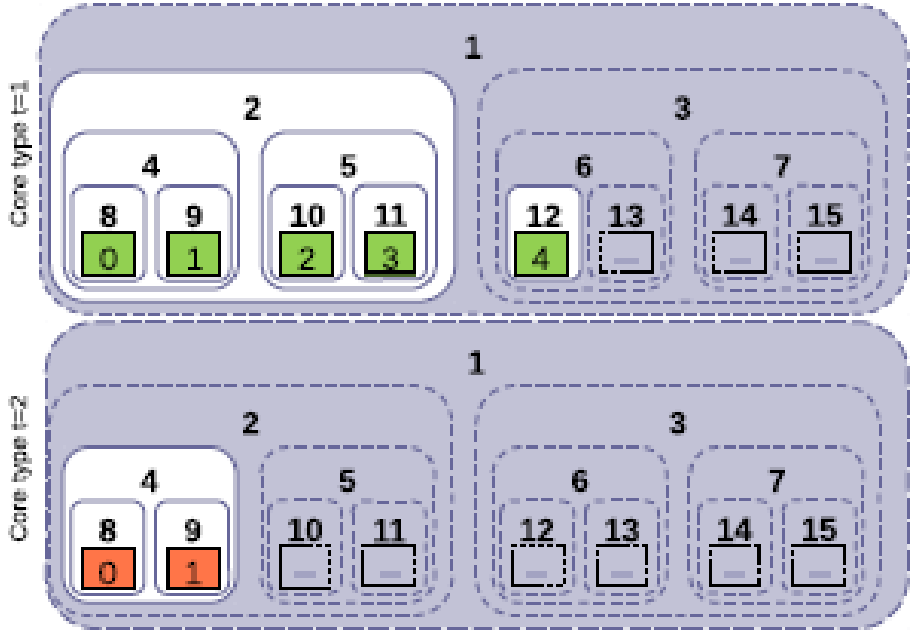
Map each task once: for each j: $\sum_{i,k} x_{i,j,k} = 1$

Each core meets deadline: for each l: $\sum_{k,j,i \text{ in } G(l)} x_{i,j,k} * time(j,i,k) \leq M$

Minimize round energy: $\min \sum_{i,j,k} x_{i,j,k} * time(j,i,k) * Pow(i,j,k) * size(i)$

Scheduling for Reconfigurable Platforms 1

- One crown for each core type
- Number of cores per type variable
  ➔ Grayed groups cannot be used
- Total area of all cores must fit platform

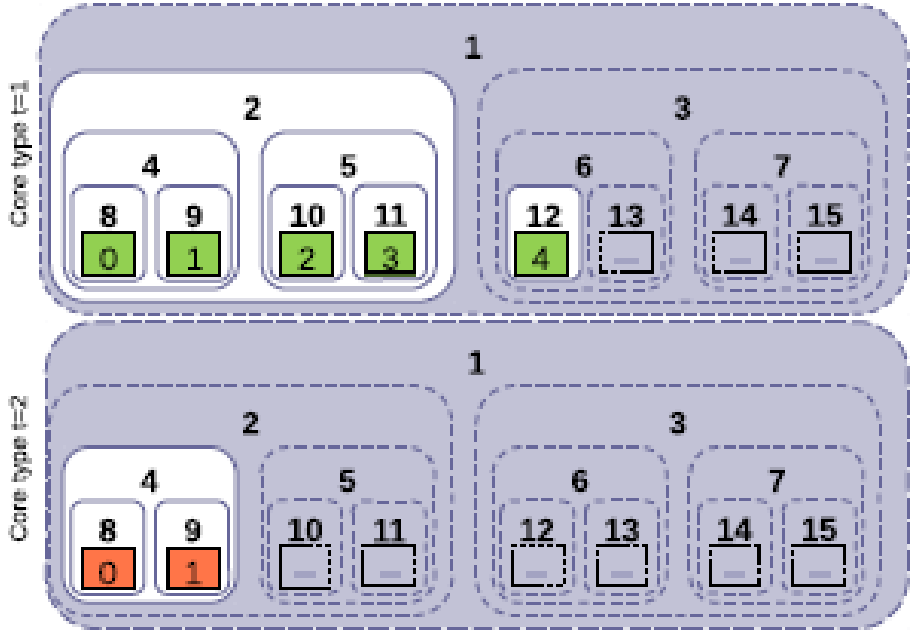


Scheduling for Reconfigurable Platforms 2

- Binary var.s $x_{i,j,k,t}$, $ex_{i,t}$, integer var.s p_t
- p_t = number of cores of core type t

  $x_{i,j,k,t}$ = 1 iff task j is mapped to core group i of core type t at freq f_k

  $ex_{i,t}$ = 1 iff core group i of core type t is non-existing

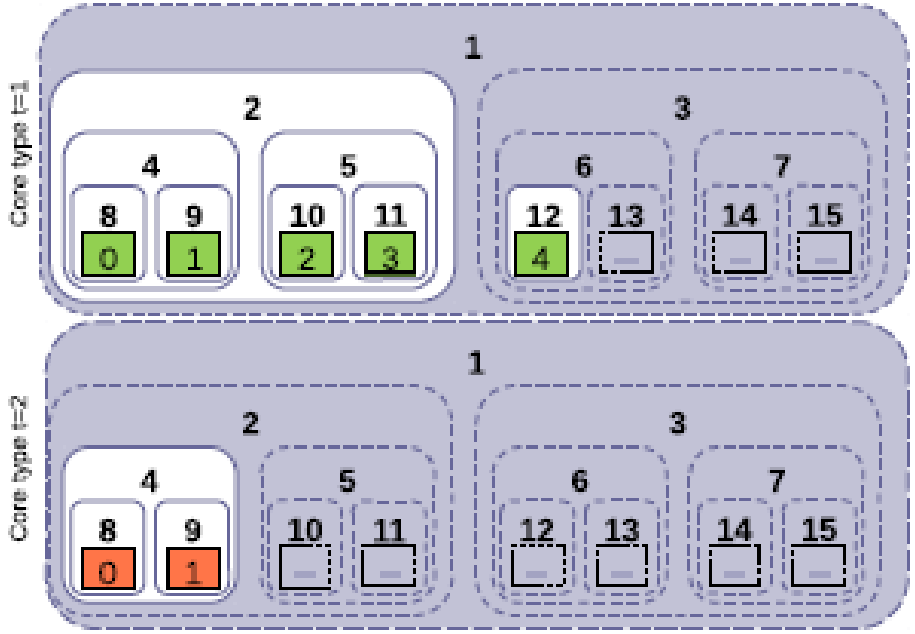


Scheduling for Reconfigurable Platforms 2

- **Additional constraints:**
  for each i,j,t: $\sum_k x_{i,j,k,t} \leq 1 - ex_{i,t}$

  for each i,t: $max(i) - p*ex_{i,t} < p_t$
  for each i,t: $max(i) + (1 - ex_{i,t})*p \geq p_t$

  $\sum_t A_t * p_t \leq A_{total}$



Experiments 1: synthetic tasksets

- **Compare with results for fixed platform (Eusipco 2019): 4 big+4 LITTLE**
  Same task sets:
  sizes 10, 20, 40, 80 tasks (10 instances of each)
  task types: Memory, Branch, Fmult, SIMD, MatMul
  Deadlines: quite tight
  Frequencies: 0.6, 0.8, 1.0, 1.2, 1.4 GHz
  Power profiles for big/LITTLE and task types from ICA3PP-2017
  Core area: big 19 mm^2, LITTLE 3.8 mm^2 from Exynox 5 Octa
  Total area = 4*19 + 4*3.8 = 91.2 mm^2

Experiments 1: synthetic tasksets

- **Compare with results for fixed platform (Eusipco 2019): 4 big+4 LITTLE**
  Same task sets: sizes, task types, deadlines, etc
  Same total area

n		10	20	40	80	Total
E_{total} current vs. original experiments	min.	65.2%	70.3%	74.5%	78.5%	65.2%
E_{total} current vs. original experiments	avg.	83.5%	79.2%	80.9%	80.4%	81.0%
E_{total} current vs. original experiments	max.	94.5%	86.4%	86.3%	83.9%	94.5%
#big vs. orig	avg.	−2.2	−2.0	−2.6	−2.3	−2.3
#LITTLE vs. orig	avg.	0.5	−0.6	0.3	−0.3	0.0
E_{idle}/E_{total} current	avg.	0.662	0.642	0.647	0.628	0.645
E_{idle}/E_{total} orig	avg.	0.741	0.770	0.764	0.770	0.761

Experiments 2: scheduling times

- **Scheduling times not much higher**

n	Experiments in [10]		Current experiments	
	Avg. scheduling time (s)	#Timeouts	Avg. scheduling time (s)	#Timeouts
10	0.355	0	2.906	0
20	5.175	0	142.798	0
40	1702.486	3	2491.273	5
80	1671.283	3	1984.073	6
Total	844.825	6	1155.262	11

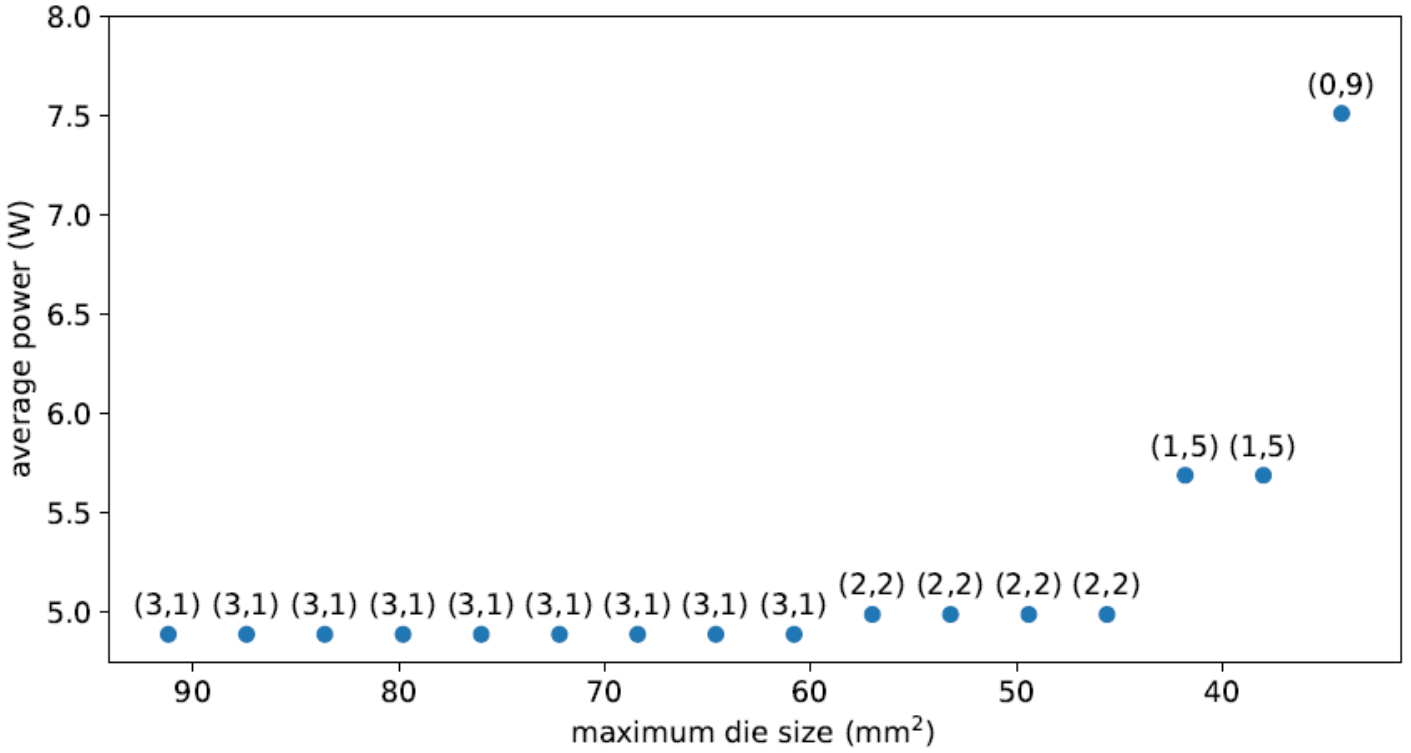
Experiments 3: tasksets from real applications

- **Real applications:**
  H.263 encoder
  Stereo depth estimation (SDE)
  Edge detection

application	H.263 encode	SDE	edge detection
E_{total} design & sched. vs. sched. only	64.9%	55.3%	62.0%
# big cores	3	2	2
# LITTLE cores	1	1	2

Experiments 4: Multiple Optimization Targets

- **Compute Pareto curve for average power vs chip area, fixed round len**



Conclusions

- Combining DSE and scheduling possible
- Results promising
- Generalizations possible: more core types, other target functions
- Area constraint only first approximation
- Evaluations for >2 core types
- Try out on real platform

Thanks a lot for your kind attention!