NICS 清華大學

Tsinghua University

Optimal Partition with Block-Level Parallelization in C-to-RTL Synthesis for Streaming Applications

Authors: Shuangchen Li, Yongpan Liu, X.Sharon Hu, Xinyu He, Pei Zhang, and Huazhong Yang

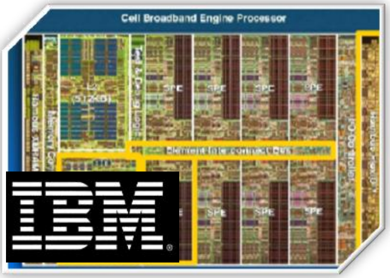
2013/01/23

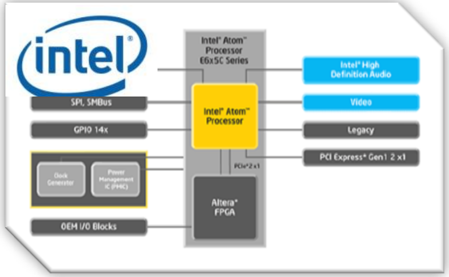
Outline

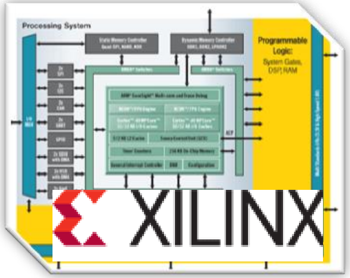
- Introduction
- Overview
- MILP-Based Solution
- Heuristic Solution
- Experimental Evaluation
- Conclusions and Future work

Introduction: Background

- *Application complexity increasing*
- **Well-developed software libraries**
- *Low speed, high power*
- **MPSoCs architecture**
- *Hardware design complexity*





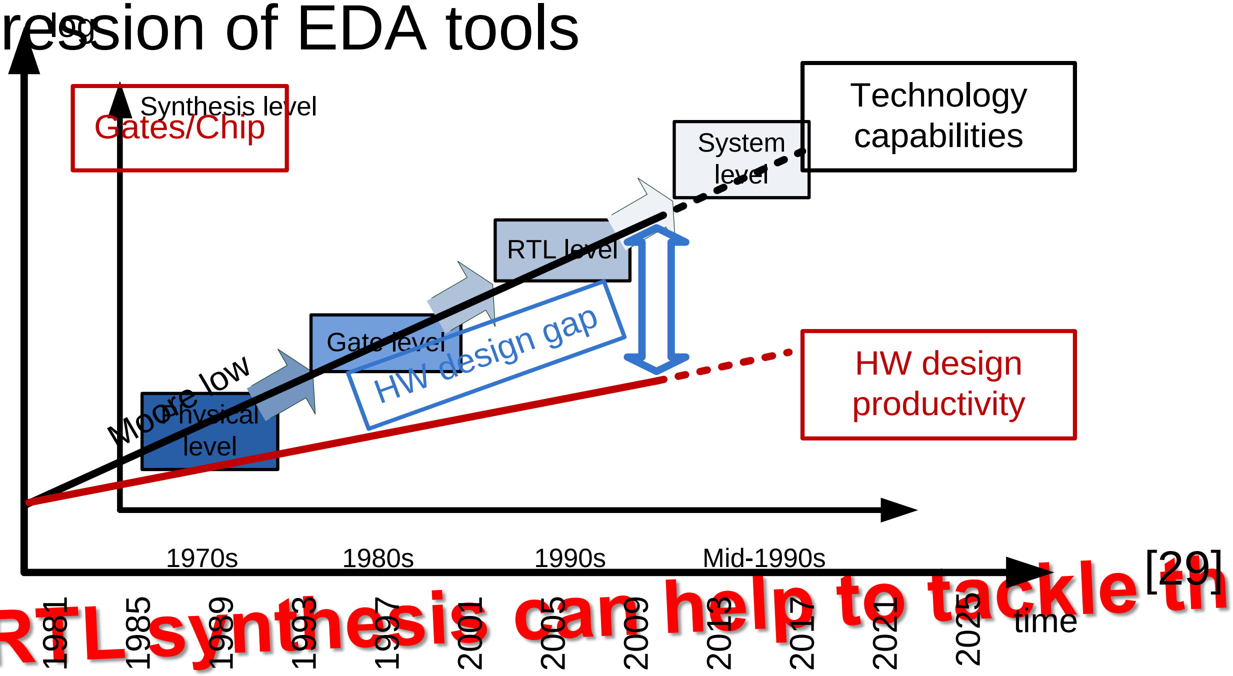


How to rapidly design hardware from existing software algorithms?

Introduction: Background (cont'd)

How to rapidly design hardware from existing software algorithms?

- **This challenge is now new. However,**
  - Ever increasing design gap
  - Progression of EDA tools



[29]

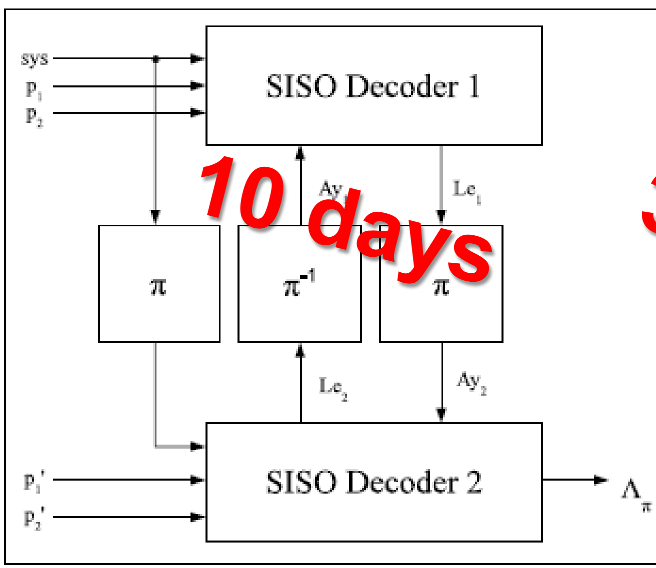
C-to-RTL synthesis can help to tackle this challenge

Introduction: Motivation

- **C2RTL tools are promising**
  - A number of C2RTL tools
  - A ... c ...

......

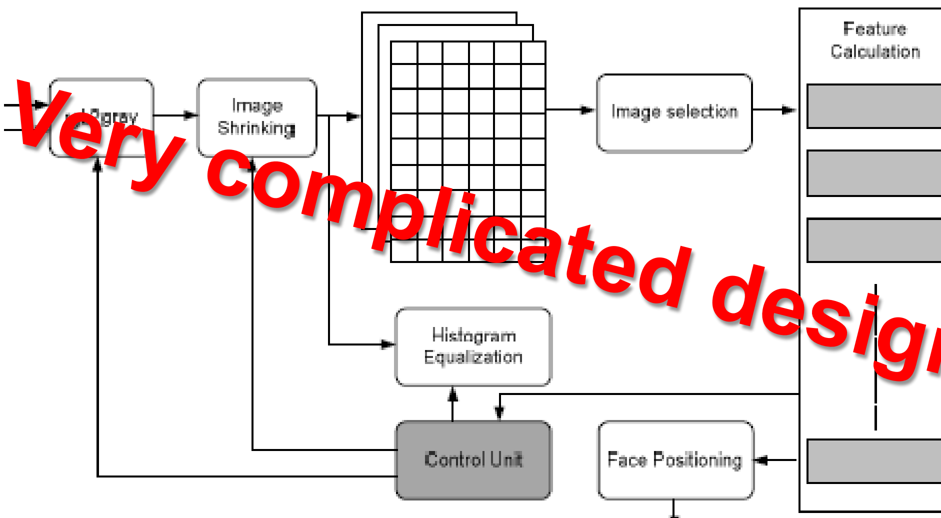
A DVB-SH Turbo Decoder [8]



10 days

3x

A face detection system [9]



Very complicated design

Introduction: Motivation (cont'd)

- ***However*, state-of-the-art C2RTL tools suffer from:**
  - Low Quality of results (QoR) for large C programs
  - System-level optimization options are limited

A Reed-Solomon decoding [28]

	Flatten approach	Hierarchical approach	speedup
Cycles	42,475,202	4,070,603	10.43x
Clock	69.74	74.2	1.06x

A JPEG encoder [10]

Design	Bluespec	Catapult-C	Xilinx
LUTs	5863	29549	2067
FFs	3162	8324	1386
Block RAMs	3	5	4
Equivalent Gate Count	267,741	596,730	297,409
Frequency (MHz)	108.5	91.2	145.3

Control and optimization at the system level are needed

Outline

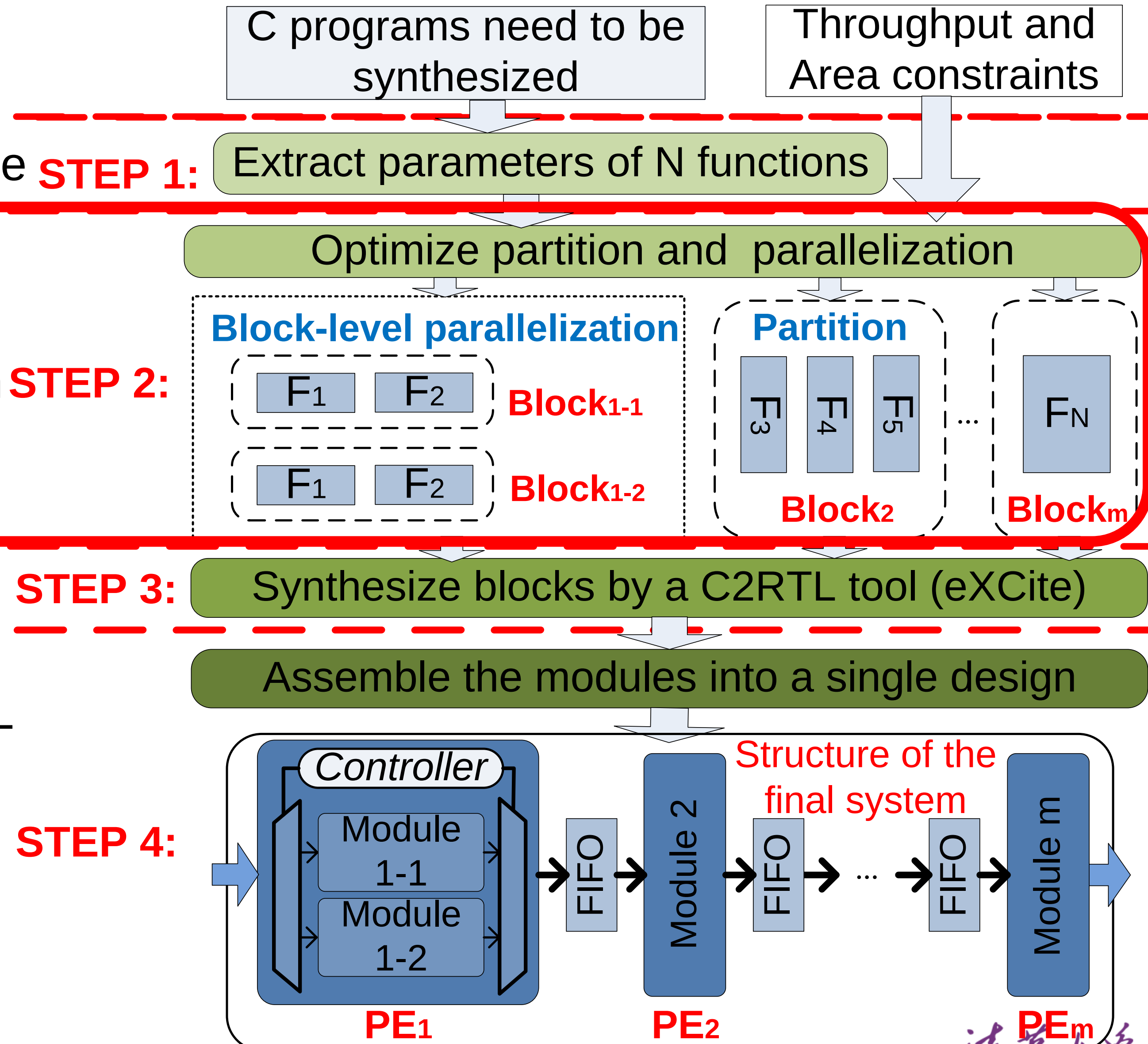
- Introduction
- Overview
- MILP-Based Solution
- Heuristic Solution
- Experimental Evaluation
- Conclusions and Future work

Overview: Our work

- **Given**
  - a large C program for a streaming application
  - system constraints (latency, area, …)
- **Determine**
  - how to partition the code into pipelined blocks **Partition**
  - which blocks should be parallelized **Parallelization**
- **The objectives**
  - Improve synthesis result quality
  - Provide more system-level optimization options

Overview: Design flow

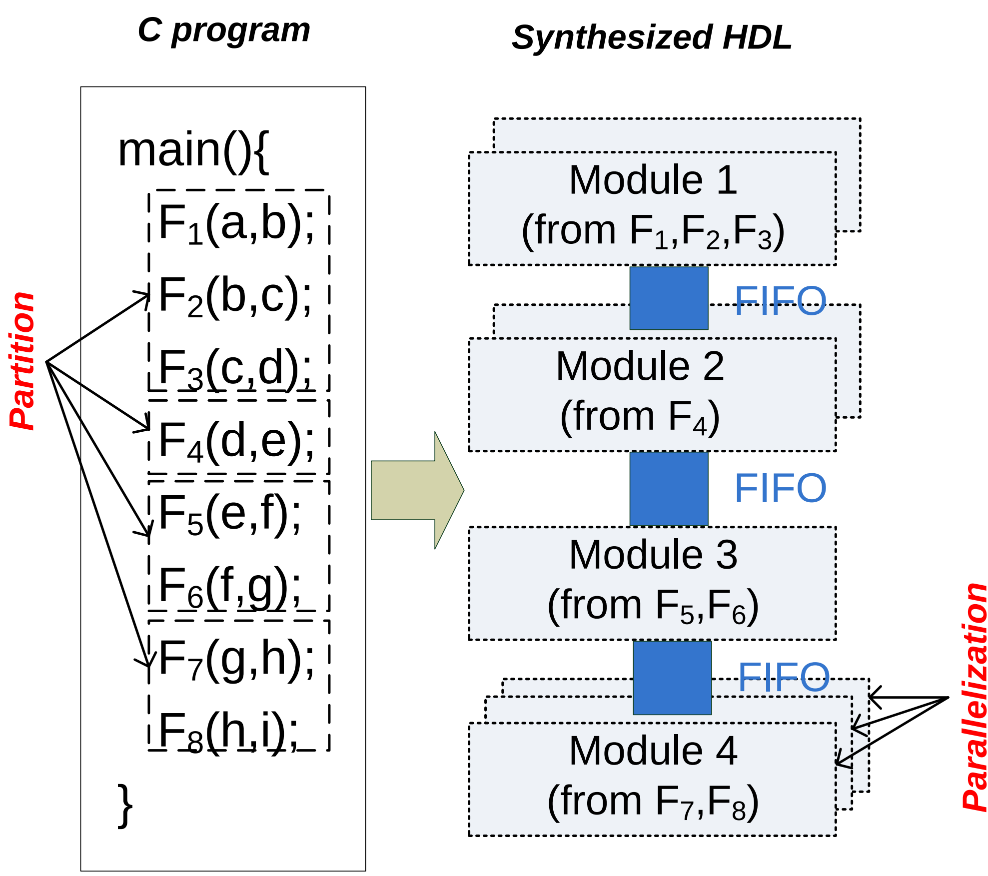
- **STEP 1:**
  - We use *eXCite* here
- **STEP 2:**
  - Determine partition and parallelization
- **STEP 3:**
  - Synthesize each block with a C2RTL tool
- **STEP 4:**
  - Construct the complete system



Tsinghua University

Overview: An example

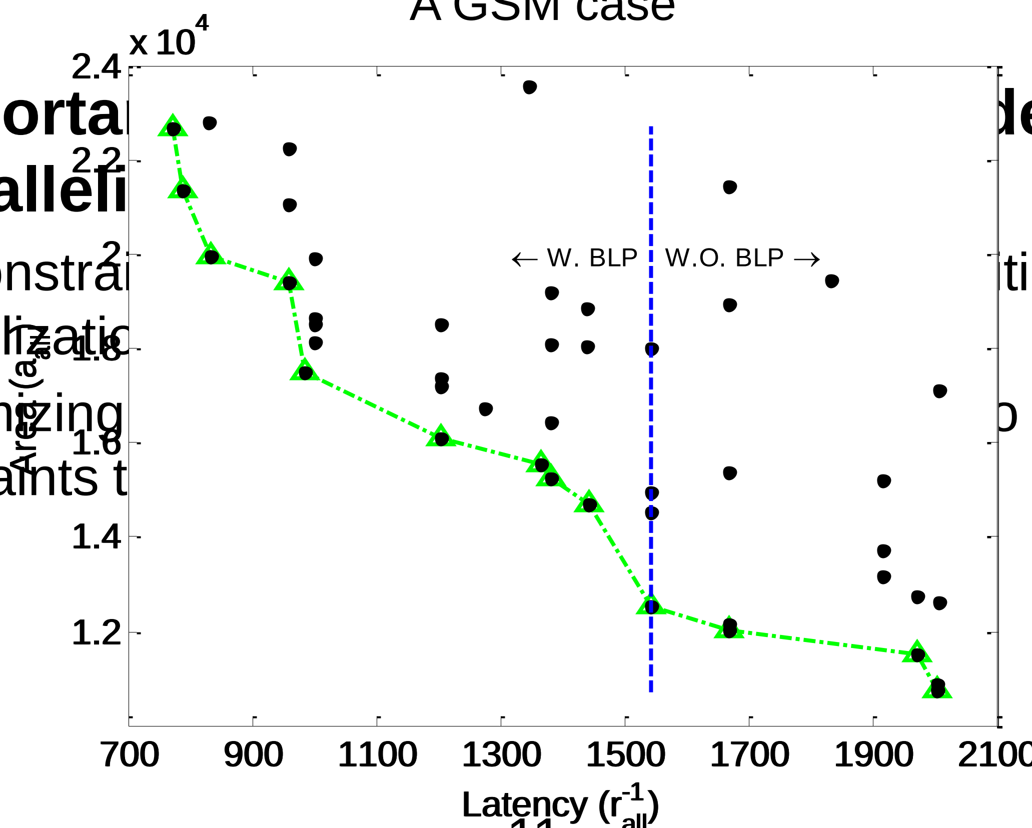
- **Given a C program:**
  - In the straight-line style
- **Given constraints:**
  - System throughput and area
- **Partition:**
  - Which functions should be synthesized together as one pipeline stage
- **Parallelization:**
  - Which synthesized modules should be parallelized



Overview: Challenges

- **The design space is large:**
  - Partition has a great impact on throughput and area
  - Parallelization has a great impact on throughput and area
  - The Pareto optimal solutions

A GSM case



- **The importa ... der partition and paralleli**
  - The constra ... ition and parallelizatic
  - If optimizing ... o apply the constraints t

Tsinghua University

Overview: Related work

	Application	Input	Target	Partition	Parallelization
A. Hagiescu and et al., in DAC2009[11]	Stream	StreamIT	MSoPC	Manually	Heuristic
J. Cong and et al., in DATE2012[12]	Stream	C	FPGA	Manually	ILP
Y. Liu and et al., in Intech Book[13]	Stream	C	FPGA	Manually	Heuristic
Y. Hara and et al., in IEICE[14]	General	C	FPGA	ILP	N/A
This work	Stream	C	FPGA	Both MILP and Heuristic (consider simultaneously)	

- **A somewhat related line of work is mapping C programs to MPSoCs (software mapping):**
  - Blocks (or tasks) can be assigned to the same processor
  - The processor area is given

Overview: Our Contribution

- **A novel *MILP* based formulation**
  - Find a partition and parallelization solution with maximum throughput or minimum area while satisfying a given area or throughput constraint, respectively
- **An efficient *heuristic* algorithm**
  - Overcome the scalability challenge facing the MILP formulation
- **Validation of the proposed methods**
  - Developing FPGA based accelerators for seven streaming applications

Outline

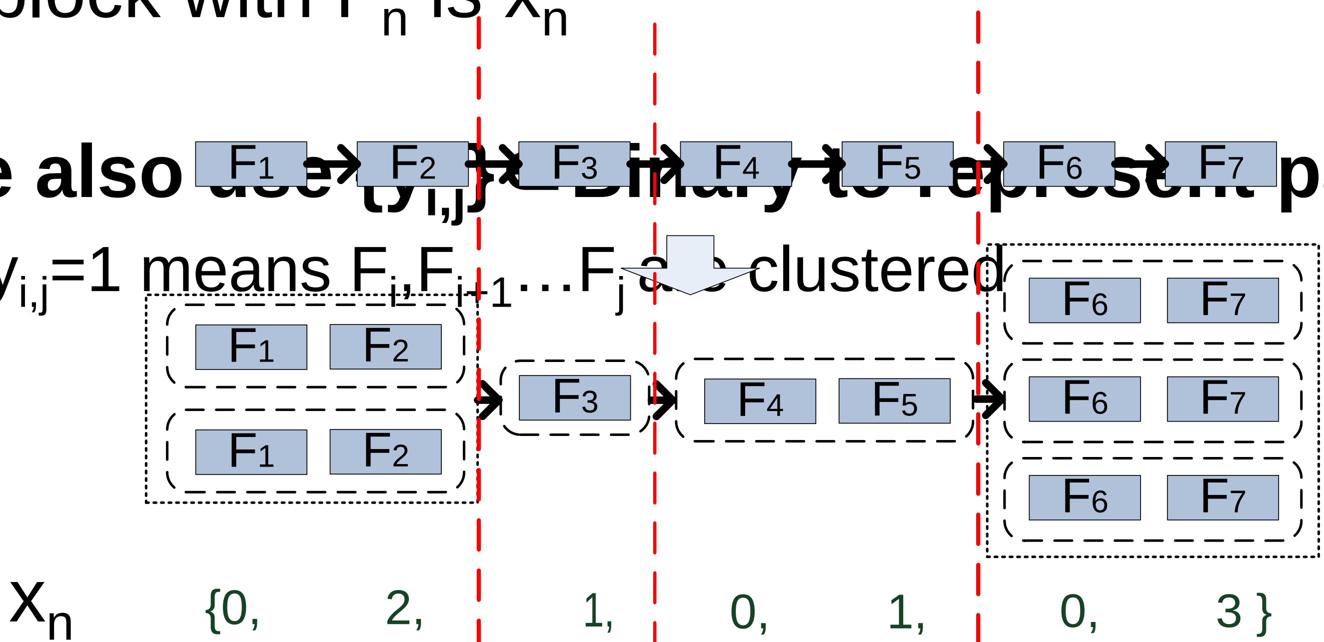
- Introduction
- Overview
- MILP-Based Solution
- Heuristic Solution
- Experimental Evaluation
- Conclusions and Future work

MILP-Based Solution: Formulation

- **Given function parameters (Para)**
  - Area, throughput … of each function
- **Determine (x_n)**
  - Which functions should be clustered to form blocks
  - Which blocks should be parallelized
- **Objective:**
  - min. Area ($a_{all}(x_n,Para)$) or max. Throughput ($r_{all}(x_n,Para)$)
- **Subject to:**
  - Area constraints ($a_{all}<A_{req}$)
  - Throughput constraints ($r_{all}>R_{req}$)
  - Connectivity constraints

MILP-Based Solution: Variable

- **We use $\{x_n\} \in Z$ to represent partition and parallelization:**
  - Partition: If $x_n=0$: F_n and F_{n+1} are in the same block
  - Parallelization: If $x_n \neq 0$: The parallelism degree of block with F_n is x_n
- **We also use $\{y_{i,j}\} \in$ Binary to represent partition**
  - $y_{i,j}=1$ means $F_i, F_{i+1} \ldots F_j$ are clustered



MILP-Based Solution: Details

- **To calculate throughput *r_{all}(x$_n$,Para)*:**

$$r_{all} \le r_{i,j} \quad \text{if } y_{i,j} = 1 \qquad (1)$$

$$r_{i,j} = \begin{cases} x_j y_{i,j} / T_{i,j} & x_j y_{i,j} < P_{i,j} \\ 1/\max\{T_{i,j}^{in}, T_{i,j}^{out}\} & \text{otherwise} \end{cases} \qquad (2)$$

- **To calculate area *a_{all}(x$_n$,Para)*:**

$$a_{all}^{le/mem} = a_{fifo}^{le/mem} + \sum_{i=1}^{i=N} \sum_{j=i}^{j=N} ((x_j - 1) O^{le/mem} + x_j A_{i,j}^{le/mem}) y_{i,j} \qquad (3)$$

- **Connectivity constraints:**

$$\begin{cases} \sum_{i=1}^{i=n} y_{i,n} \le x_n \\ x_n = 1 \text{ when } \sum_{i=1}^{i=n} y_{i,n} = 0 \end{cases} \qquad (4)$$

$$\sum_{i=1}^{i=j-1} y_{i,j-1} = \sum_{i=j}^{i=N} y_{j,i} \quad \forall j \in [2, N] \qquad (5)$$

$$\sum_{i=1}^{i=j} y_{i,j} + \sum_{i=j}^{i=N} y_{j,i} - y_{j,j} \le 1 \quad \forall j \in [1, N] \qquad (6)$$

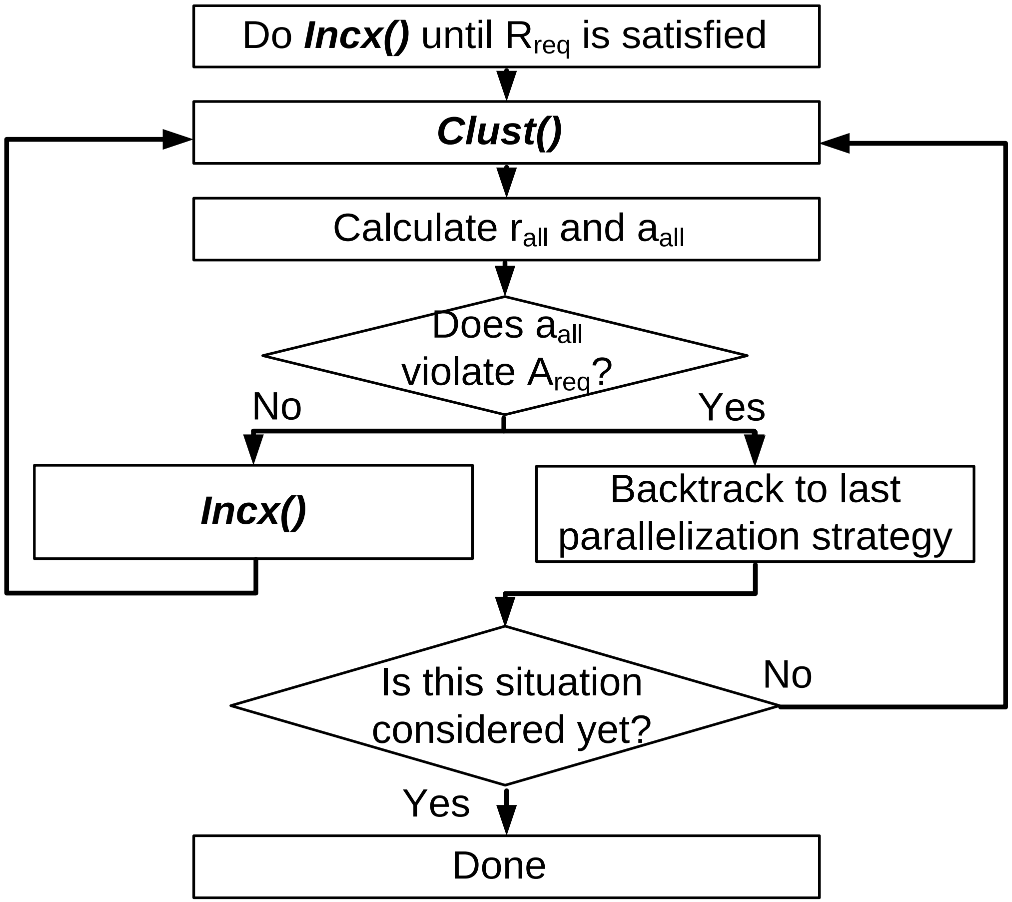
Outline

- Introduction
- Overview
- MILP-Based Solution
- Heuristic Solution
- Experimental Evaluation
- Conclusions and Future work

Heuristic Solution: Overview

- **Motivation:**
  - MILP is not scalable
  - Bad feasible regions may incur long running time even when *N* is small
- **Consider partition and oarallelization separately (constructive algorithm):**
  - Parallelization before partition *to increase throughput*: ***Incx()***
  - Partition for the given parallelization *to reduce area*: ***Clust()***
  - Implement ***Incx()*** and ***Clust()*** in a backtracking iterative way

Heuristic Solution: Algorithm



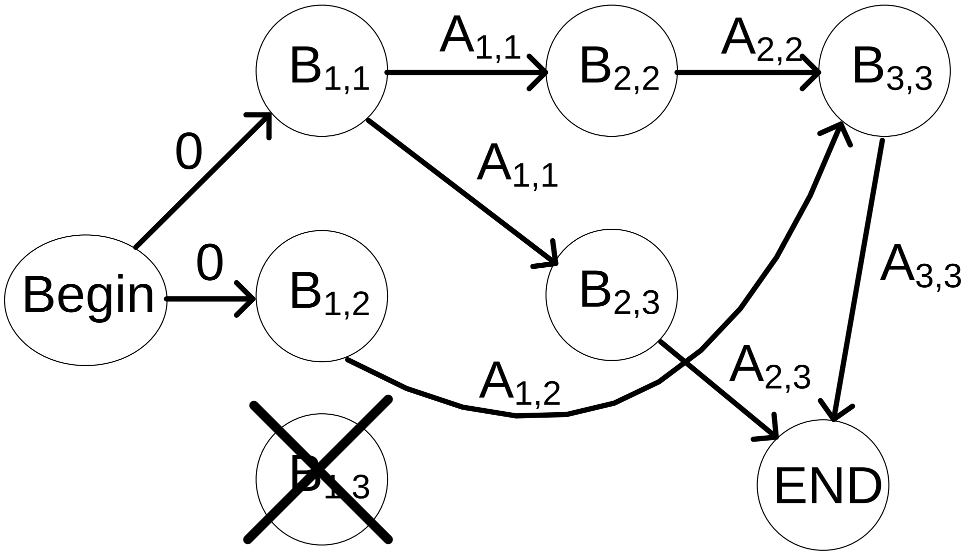
- ***Incx()***: Parallelization before Partition ***to increase throughput***
- ***Clust()***: Partition for the given Parallelization ***to reduce area***

NICS

清華大學 Tsinghua University

Heuristic Solution: Algorithm (cont'd)

- ***Incx()***, Parallelization before Partition:
  - Increase the parallelization degree of the bottleneck function
- ***Clust()***, Partition under the given Parallelization:
  - Model the blocks and their connections as a graph
  - Convert the problem to a *shortest path problem*



NICS

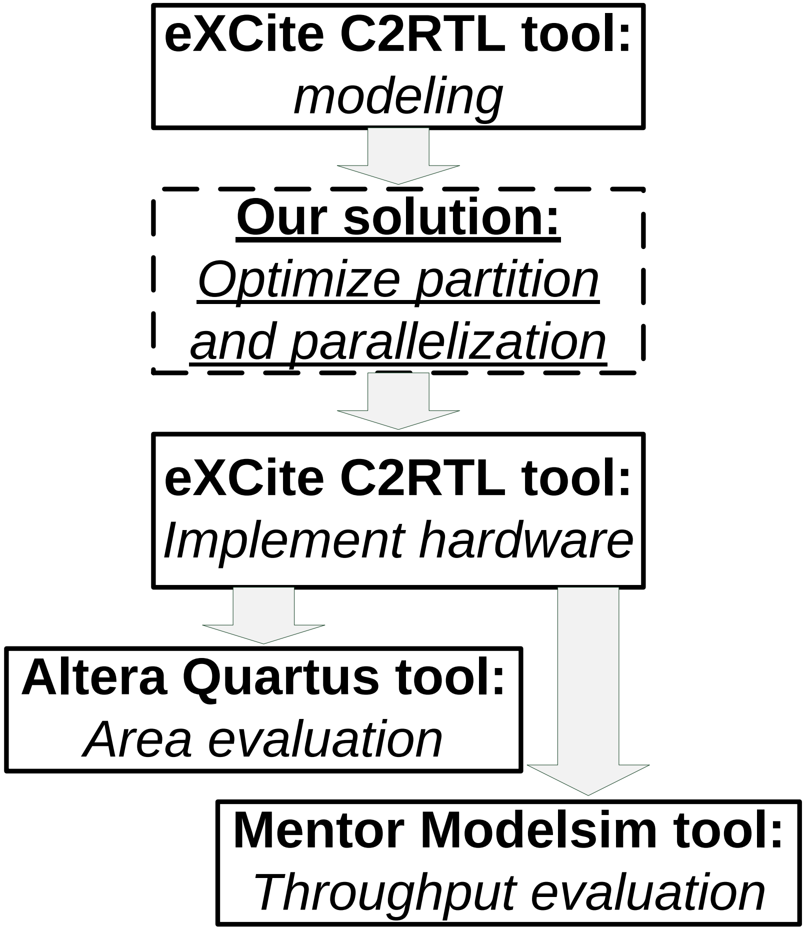
Tsinghua University

Outline

- Introduction
- Overview
- MILP-Based Solution
- Heuristic Solution
- Experimental Evaluation
- Conclusions and Future work

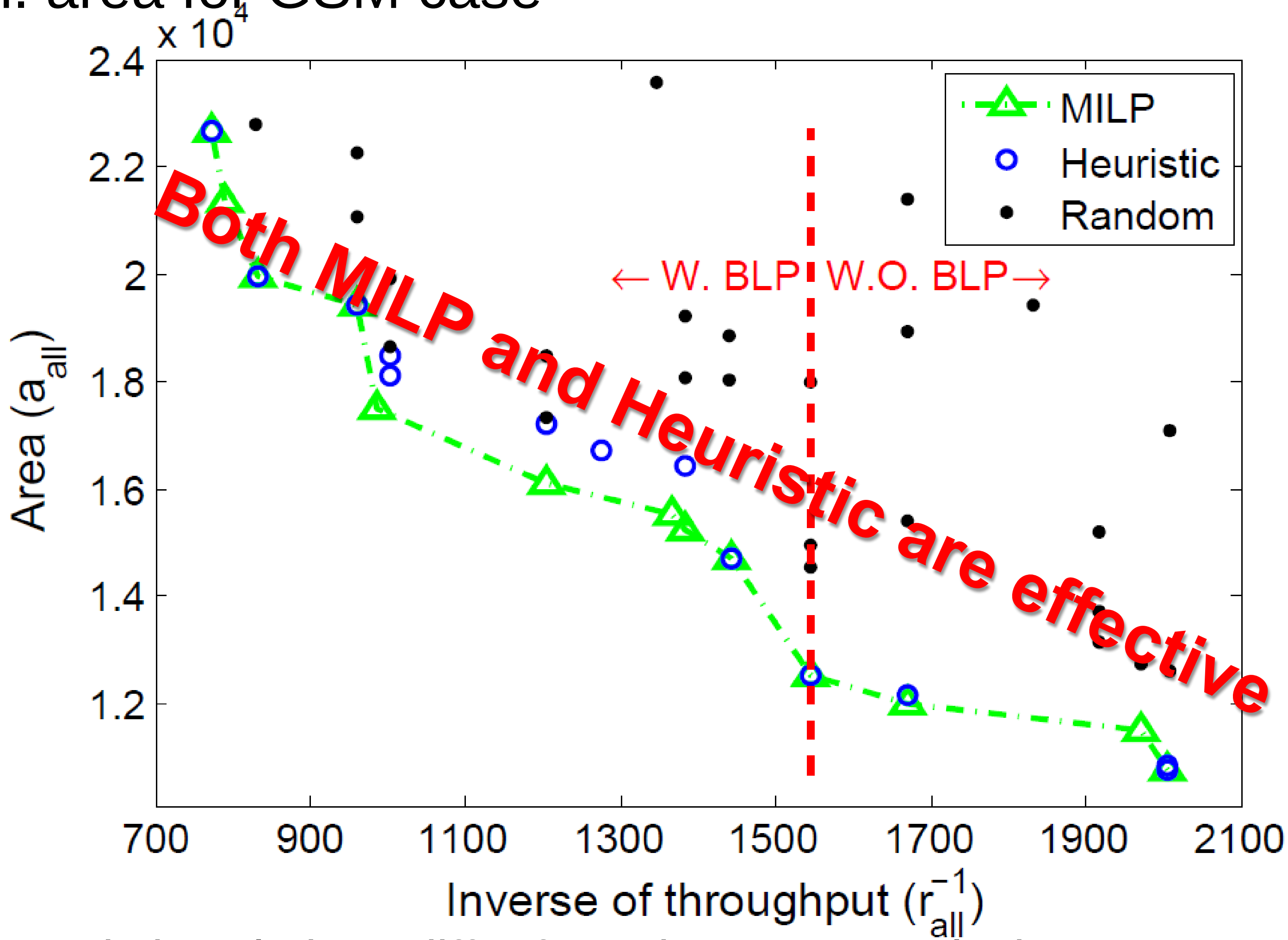
Experiments: Set up

- **7 Benchmark [21]:**
  - ADPCM
  - JPEG encoder/decoder
  - AES encryption/decryption
  - GSM
  - Filter Groups
- **Environment & flow:**
  - C2RTL: eXCite
  - Logic synthesis: Quartus II (cyclone II)
  - Simulation: Modelsim



Experiments: Validate proposed method

- Min. area for GSM case



  - Heuristic solutions differ from the MILP results by 2.3% on average

Exp.: Validate proposed method (cont'd)

- Min. Area for 7 benchmarks

Benchmark		Constraints[1] R_{req}^{-1}	MILP vs. Heuristic results $\{x_n\}$	r_{all}^{-1}	a_{all}
ADPCM	MILP	200	{0,2,0,0,1,1}[2]	197	16624
	Heu.		{2,0,0,1,1,1}	164	17513
AES encryption	MILP	5000	{1,1,1,0,1,0,1}	4678	16463
	Heu.		{1,1,1,0,1,0,1}	4678	16463
AES decryption	MILP	4500	{1,0,0,2,0,0,2}	4325	24768
	Heu.		{1,2,0,2,2,0,2}	3867	31036
JPEG encoder	MILP	575	{2,1,1,1,1,1,1,1,0,2}	355	23944
	Heu.		{2,1,1,1,1,1,1,0,2}	355	23944
JPEG decoder	MILP	2500	{0,0,0,0,1,0,0,2}	1766	8065
	Heu.		{0,0,0,0,1,2,0,1}	1670	8562
GSM	MILP	1000	{0,2,1,0,2,0,2,0,0,1}	987	15937
	Heu.		{2,2,0,2,2,0,2,0,0,1}	833	18775
Filter Groups	MILP	18000	{0,0,0,1,0,1,0,2,2,2,0,1,1,1}	17102	28776
	Heu.		{1,0,0,1,0,1,0,2,2,2,0,1,1,1}	10372	28994

Solutions are effective with different applications

– Heuristic with a difference of 7.5% on average

NICS

清華大學 Tsinghua University

Experiments: Running time

- Running time:

Bench-mark	Objective	Constraints R^{-1}_{req}	Constraints A_{req}	Time (sec) MILP	Time (sec) Heu.	Result (r^{-1}_{all}, a_{all}) MILP	Result (r^{-1}_{all}, a_{all}) Heu.
GSM (N=10)	min a_{all}	1000	–	9.089	0.025	987, 15937	833, 18775
	max r_{all}	3000	17000	37.648	0.192	1208, 16008	1442, 15171
	max r_{all}	–	19000/400000[1]	41.135	0.098	833, 18775	1024, 18031
	max r_{all}	–	18900/300000	*Failed*	0.095	*Failed*	1024, 18031
Filter groups (N=14)	min a_{all}	19000	–	355.80	0.026	18548, 20829	18548, 20829
	max r_{all}	–	50000	395.47	0.025	17102, 26571	10372, 28994
	max r_{all}	–	30000/25000	*Failed*	0.121	*Failed*	10222, 23907

[1] With two separated constraints for A^{le}_{req} and A^{mem}_{req}, respectively.

350-15000x shorter time than the MILP

– The heuristic solutions are worse by 7.2% on average

N I C S

清華大學 Tsinghua University

Outline

- Introduction
- Overview
- MILP-Based Solution
- Heuristic Solution
- Experimental Evaluation
- Conclusions and Future work

Conclusions and Future work

- Conclusions :
  - Our work adopts a hierarchical framework with automatic C-code partition and block-level parallelization
  - Both an MILP-based solution and a heuristic solution are proposed
  - Experimental results obtained from seven real applications show that our approaches are effective
- Future work:
  - Extend the solution to C program with feedback
  - Taking power into consideration

Reference

- [1]-[27] is listed in the paper
- [28] "Comparison of high level design methodologies for algorithmic ips: Bluespec and c-based synthesis," Ph.D. dissertation, MIT, 2009
- [29] "ITRS roadmap on Design" 2011 Edition

THANK YOU!

MILP-Based Solution: Linearization

- **Linearize $x_j y_{i,j}$: $z_{i,j}=x_j y_{i,j}$**

$$-My_{i,j} \le z_{i,j} \le My_{i,j}$$

$$x_j - M(1-y_{i,j}) \le z_{i,j} \le x_j + M(1-y_{i,j})$$

- **Linearize Equation (1):**

$$r_{all} \le r_{i,j} + M(1-y_{i,j}) \quad \forall 1 \le i \le j \le N$$

- **Linearize Equation (2):**

$$r_{i,j} \le \begin{cases} z_{i,j} / T_{i,j} \\ 1/\max\{T_{i,j}^{in}, T_{i,j}^{out}\} \end{cases}$$

- **Linearize Equation (4):**

$$\sum_{i=1}^{i=n} y_{i,n} \le x_n \le M \cdot \sum_{i=1}^{i=n} y_{i,n} \quad x_n \in \mathbf{N}, \ \ y_{i,j} \in binary$$

NICS

清華大學
Tsinghua University

Exp.: Validate proposed method (cont'd)

- Min. area or Max. throughput for GSM

	Objective	Constraints		MILP vs. Heuristic result		
		R_{req}^{-1}	A_{req}	$\{x_n\}$	r_{all}^{-1}	a_{all} or $a_{\text{all}}^{\text{le}}/a_{\text{all}}^{\text{mem}}$
MILP	min a_{all}	1000	–	{0,2,1,0,2,0,2,0,0,1	987	15937
Heuristic				{2,2,0,2,2,0,2,0,0,1	833	18775
MILP	min a_{all}	1600	–	{1,0,1,1,1,1,0,0,0,1	1545	12132
Heuristic				{1,0,1,1,1,1,0,0,0,1	1545	12132
MILP	max r_{all}	–	15000	{1,0,1,1,1,1,0,0,0,1	1545	12132
Heuristic				{1,0,1,1,1,1,0,0,0,1	1545	12132
MILP	max r_{all}	–	19000	{0,2,1,0,2,2,1,0,0,1	987	17415
Heuristic				{0,2,2,0,2,1,1,0,0,1	1204	18341
MILP	max r_{all}	–	[1]19000	{2,2,0,2,2,0,2,0,0,1	987	17094/67284
Heuristic			/70000	{0,0,2,0,2,1,1,0,0,1	1204	19927/52208

Solutions are effective with various settings