Multi-Mode Pipelined MPSoCs for Streaming Applications

Haris Javaid Daniel Witono Sri Parameswaran

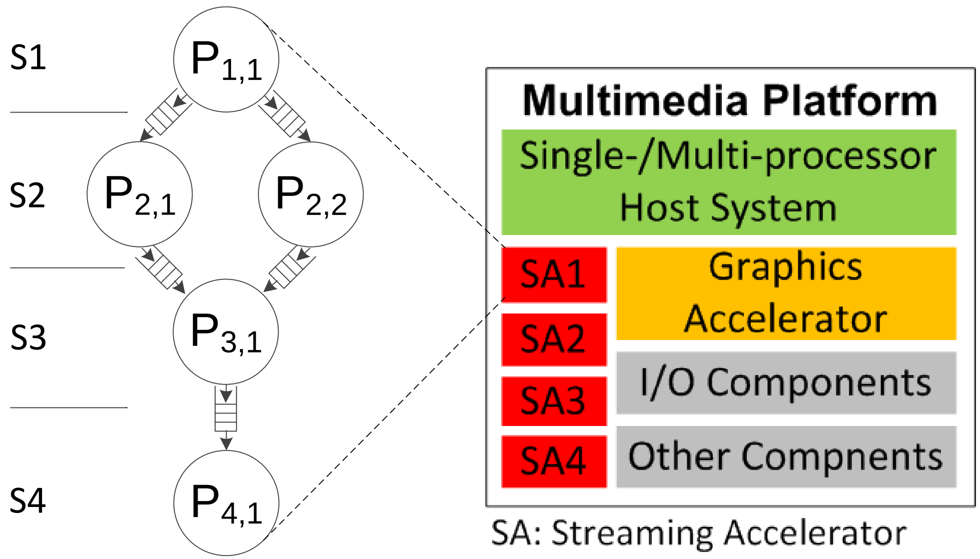
January 23, 2013

School of Computer Science and Engineering
The University of New South Wales
AUSTRALIA

THE UNIVERSITY OF NEW SOUTH WALES
SYDNEY • AUSTRALIA

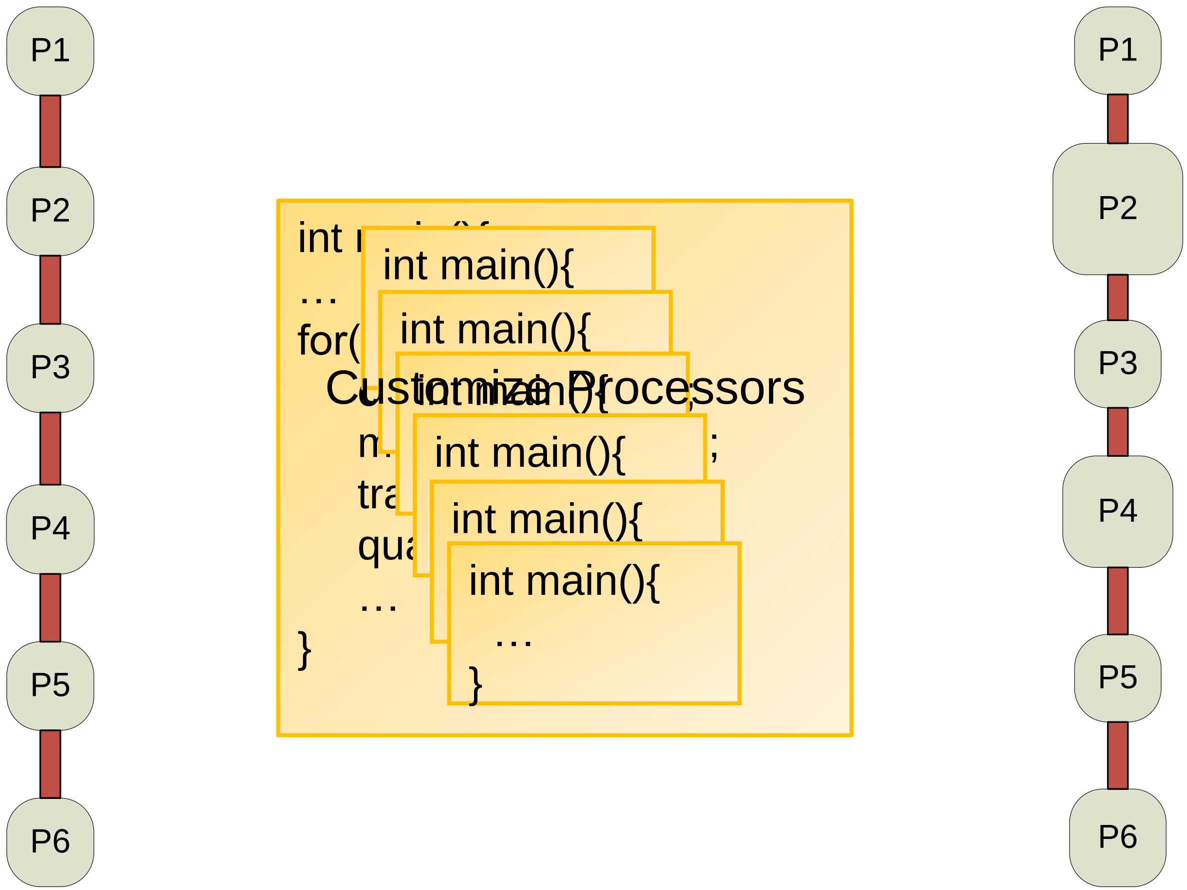
Introduction

- Multimedia platforms are increasingly becoming heterogeneous
  - Single-/Multi-processor host system
  - Graphics and streaming accelerators
  - Examples: Tegra, OMAP, etc.
- Streaming accelerators
  - Optimized at design-time for performance and energy efficiency
  - Based upon pipelined MPSoCs rather than ASICs in this work



Pipelined MPSoCs

- Pipelined MPSoCs
  - Task- and pipeline-level parallelisms (data-flow structure)
  - Data- and instruction-level parallelisms (SIMD and VLIW instructions)

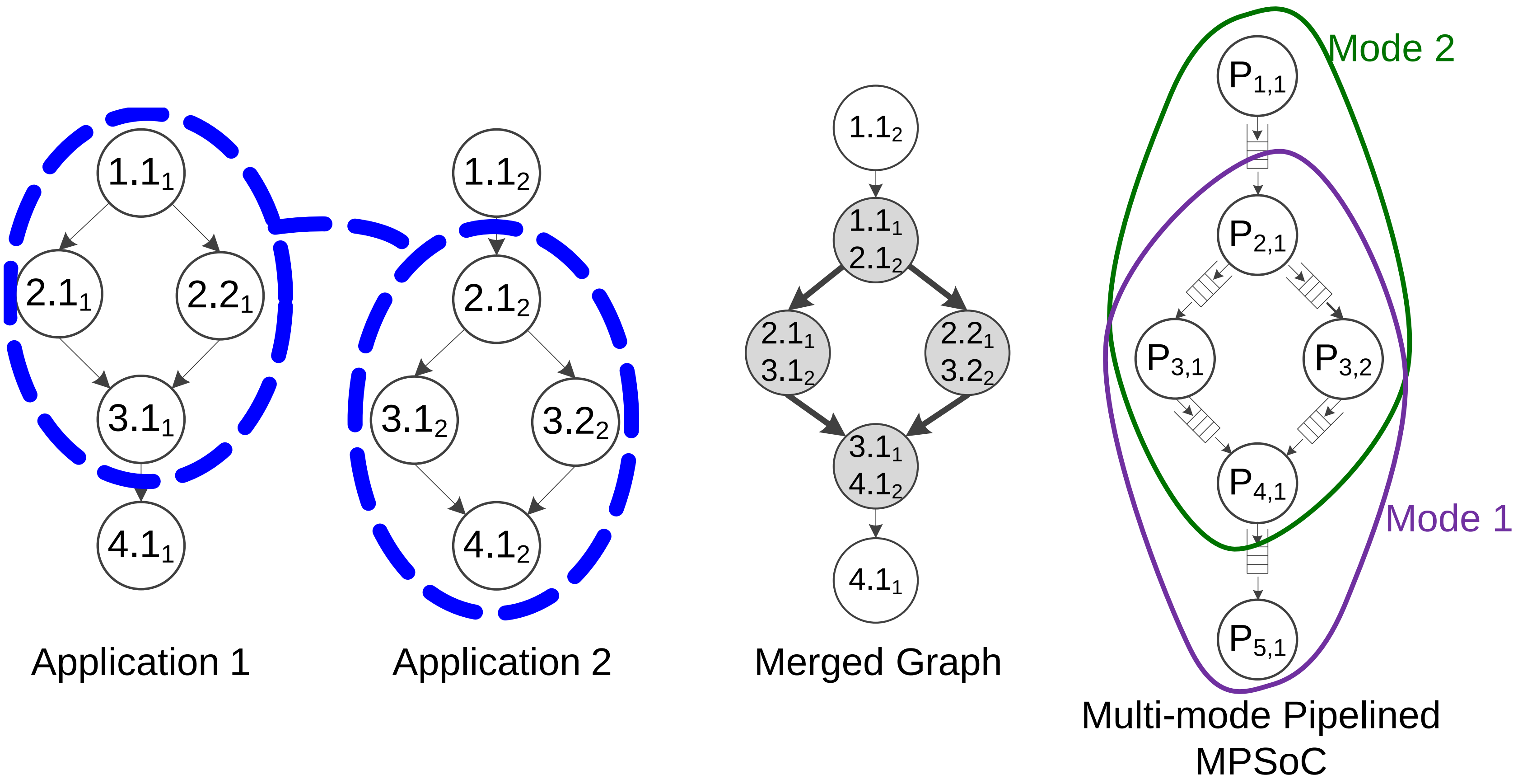


Research Aim

- All the streaming accelerators may not be active at all times
  - Either browse pictures or watch videos (H.264 and JPEG not used simultaneously)
  - Differing standards (H.264, MPEG, VC1 not used simultaneously)
- Multi-mode accelerator
  - Combine mutually exclusive accelerators to reduce area footprint
  - Simultaneously active accelerators are not combined
- **Our proposal**
  - Multi-mode Pipelined MPSoCs – a mode refers to execution of one streaming application
  - Combine application graphs, and then derive a multi-mode pipelined MPSoC

An Example

- Assumptions
  - An application graph represents the corresponding pipelined MPSoC
  - Notation $m.n_x$ means *n-th* node in *m-th* stage of *x-th* application



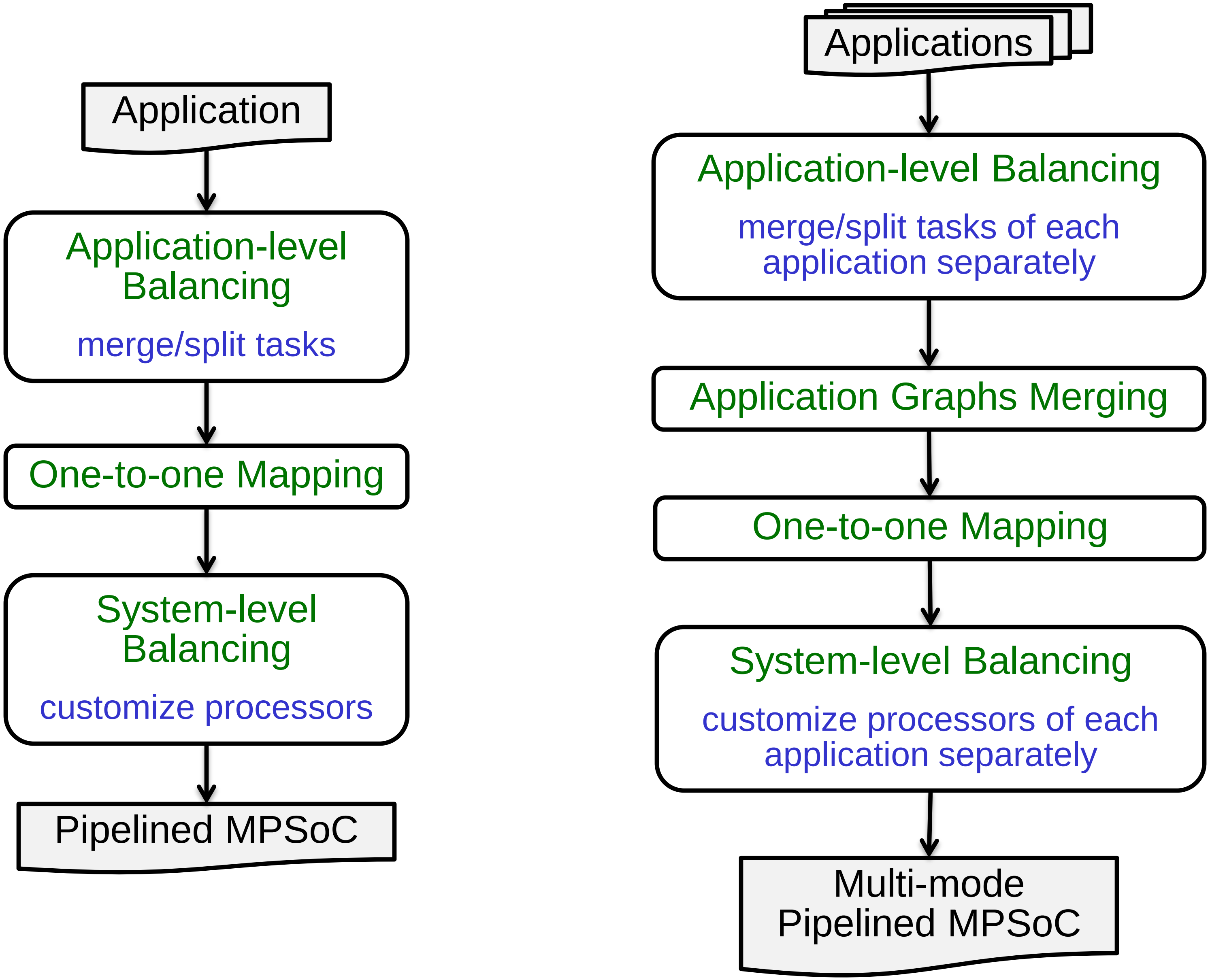
Problem Statement

- Given X application graphs, the goal is to combine them into one graph such that the pipelined MPSoC derived from it has minimal area
- In the merged graph
  - Minimize the number of nodes: cost of processors
  - Minimize the number of edges: cost of processor/FIFO ports
  - Minimize the capacity of edges: cost of FIFO sizes
- Merging applications graph is an NP complete problem [reference in paper]
- Our Solution
  - Near-optimal but fast heuristics
    - MaxS
    - MaxN
  - Optimal heuristic
    - MaxC

Related Work

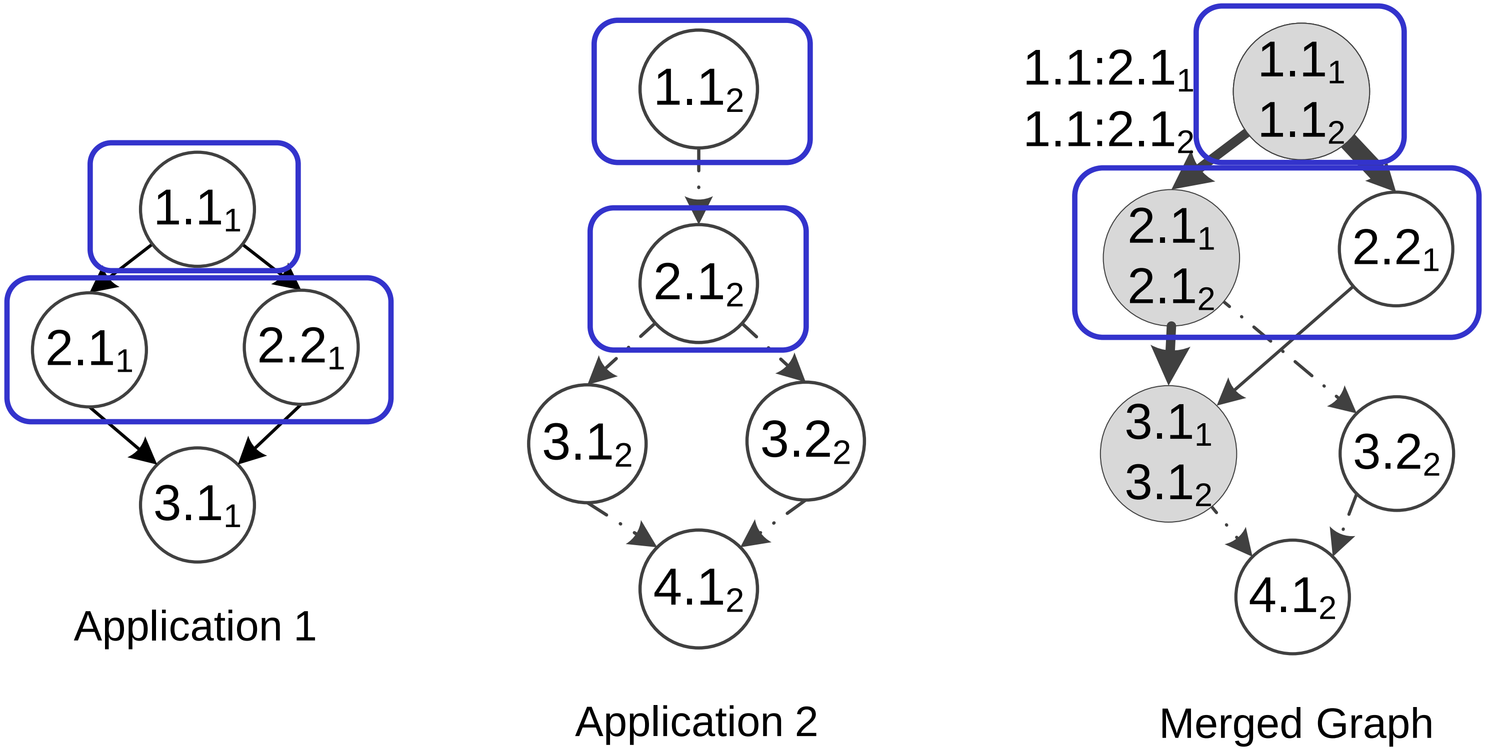
- Data-path merging in digital design
  - Bipartite graph matching [DATE'01]
  - Subsequence/Substring matching [DAC'04]
  - Finding maximum clique [IEEE TCAD'05] [IEEE TCAD'09]
- Typical multi-mode systems [references in paper]
  - Fixed platform
  - Involves selection of processing elements, and mapping and scheduling
- Application graphs merging
  - Merging multiple uses-cases of applications [ACM TODAES'08]
  - Merging based upon subsequence/substring matching [JRC'12]
- Our Contribution
  - Multi-mode pipelined MPSoCs
  - Use of maximum clique approach to find optimal merging
  - Three heuristics to tradeoff accuracy with running time

Design Flows



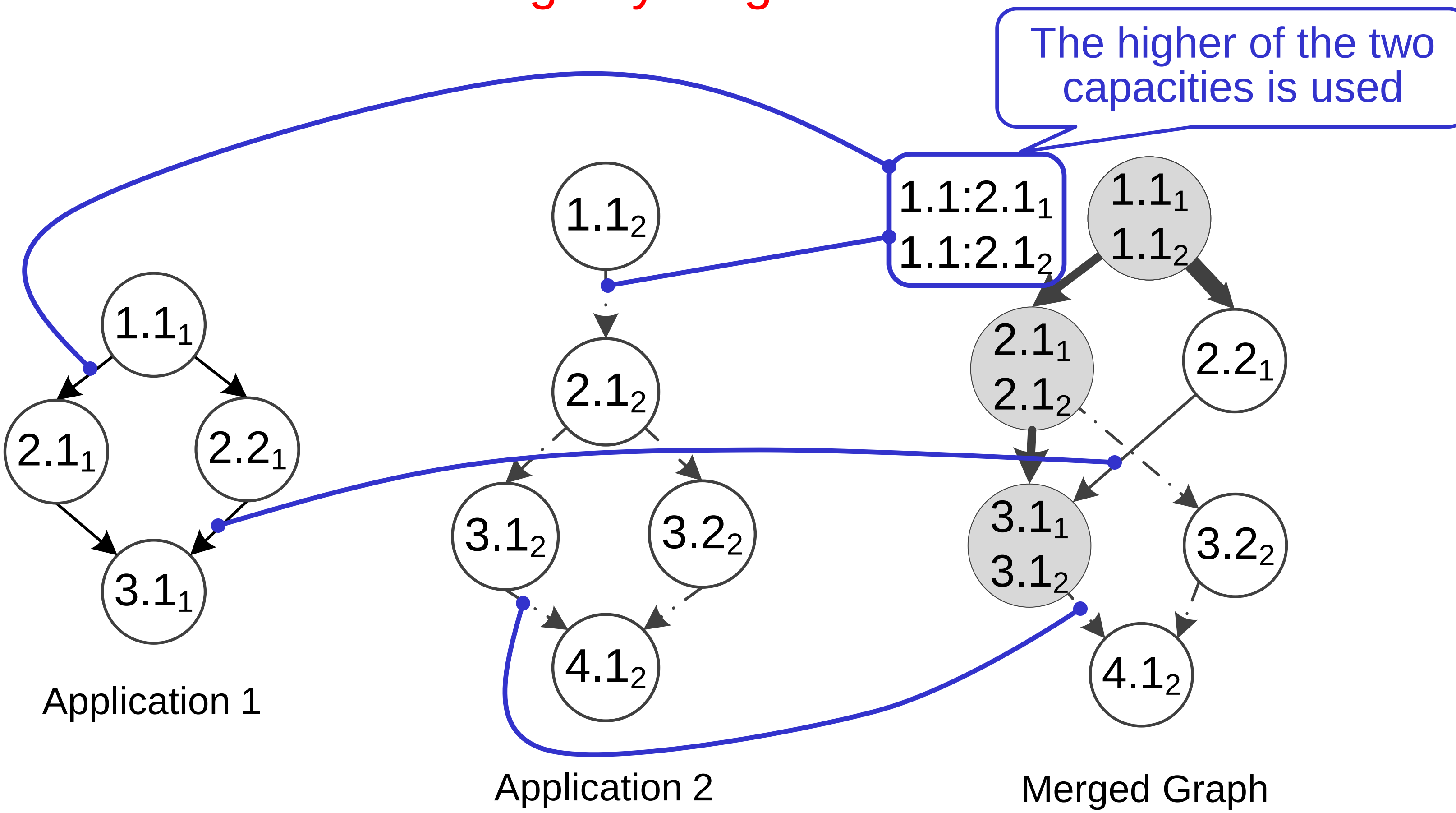
Heuristic MaxS (Max. Stages)

- Works on application graphs' topologies
- Combines nodes on a stage by stage basis



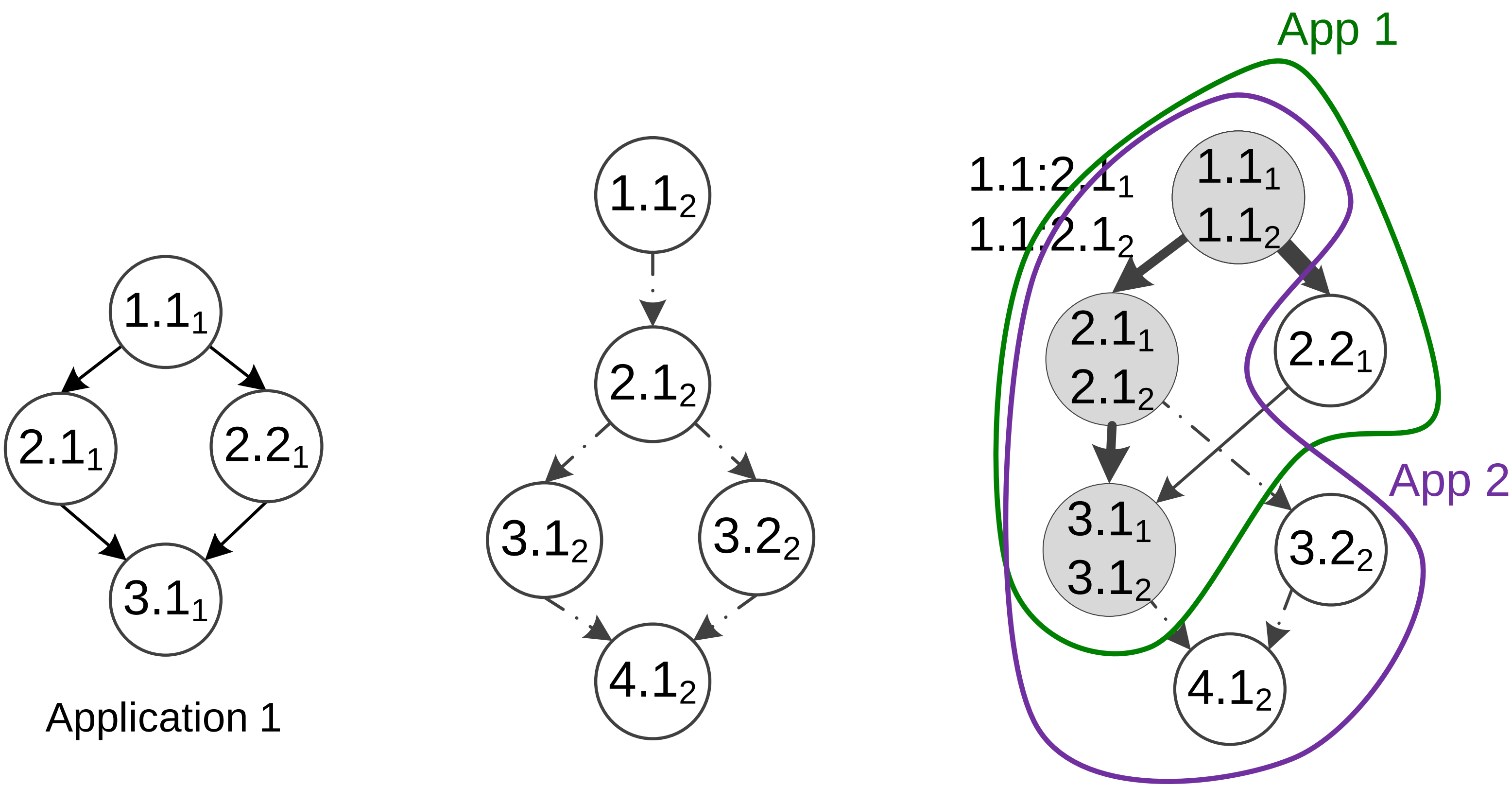
Heuristic MaxS (Max. Stages)

- Works on application graphs' topologies
- Combines nodes on a stage by stage basis



Heuristic MaxS (Max. Stages)

- Works on application graphs' topologies
- Combines nodes on a stage by stage basis



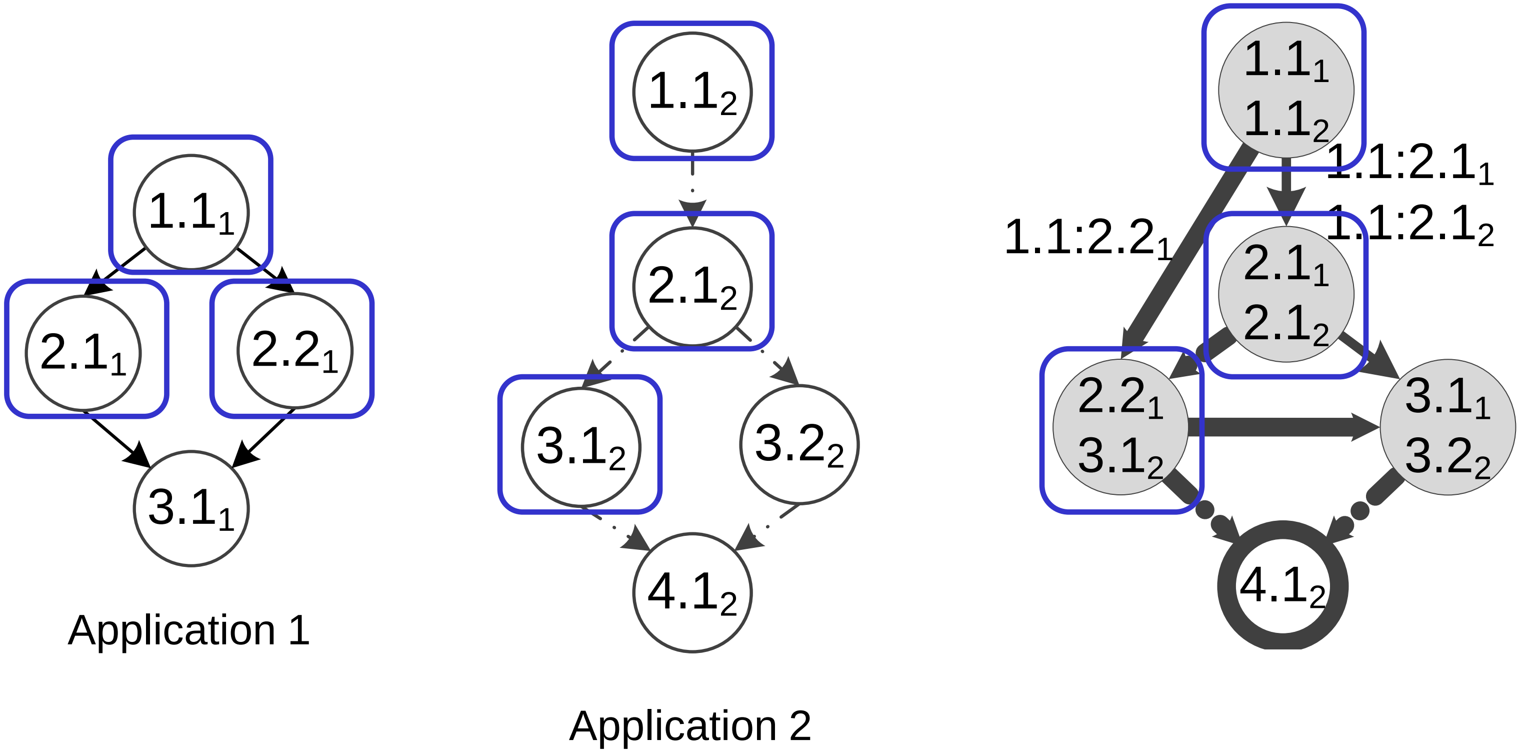
Application 1

Application 2

Merged Graph

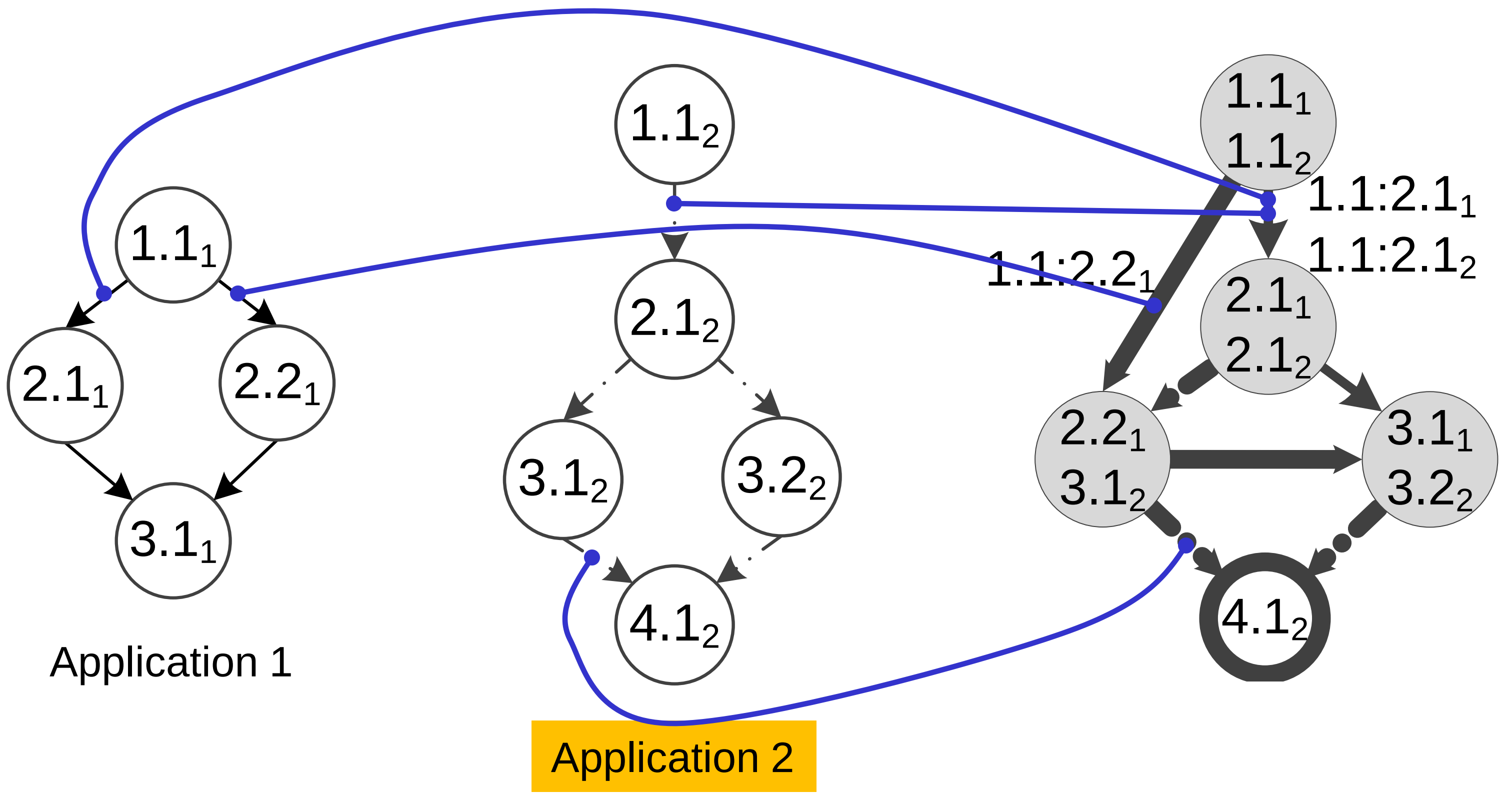
Heuristic MaxN (Max. Nodes)

- Nodes in the merged graph should not exceed the application with maximum number of nodes
- Combines nodes in a breadth-first manner
- Exhausts all permutations of merging graphs



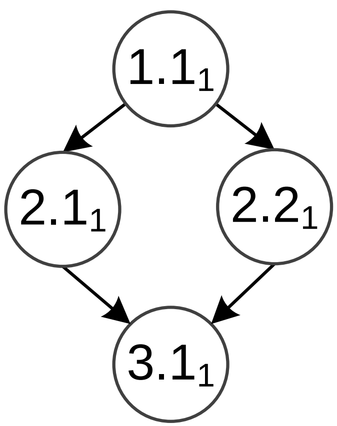
Heuristic MaxN (Max. Nodes)

- Nodes in the merged graph should not exceed the application with maximum number of nodes
- Combines nodes in a breadth-first manner
- Exhausts all permutations of merging graphs

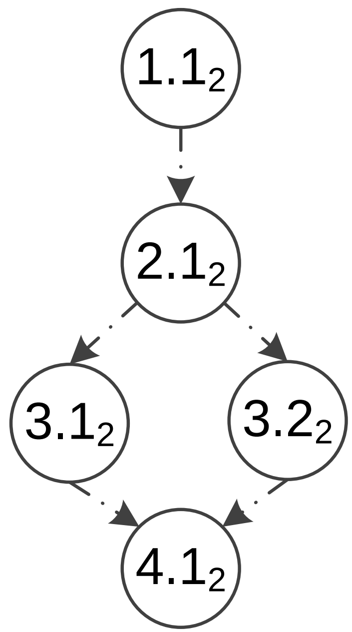


Heuristic MaxN (Max. Nodes)

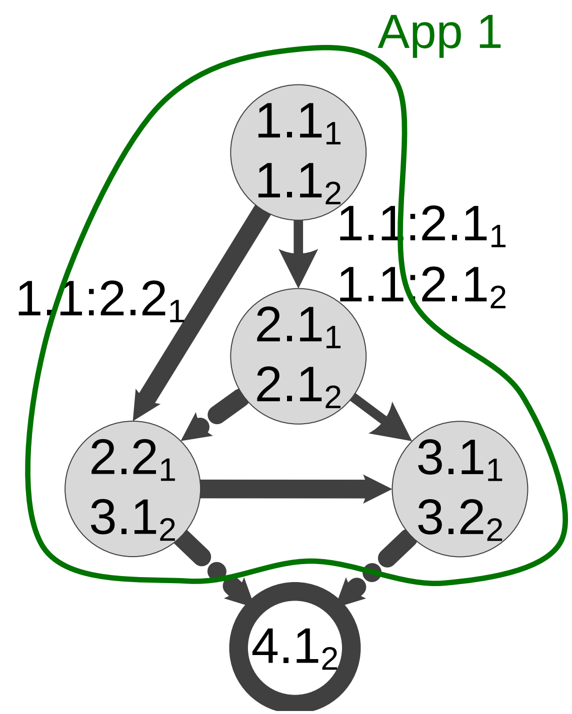
- Nodes in the merged graph should not exceed the application with maximum number of nodes
- Combines nodes in a breadth-first manner
- Exhausts all permutations of merging graphs



Application 1

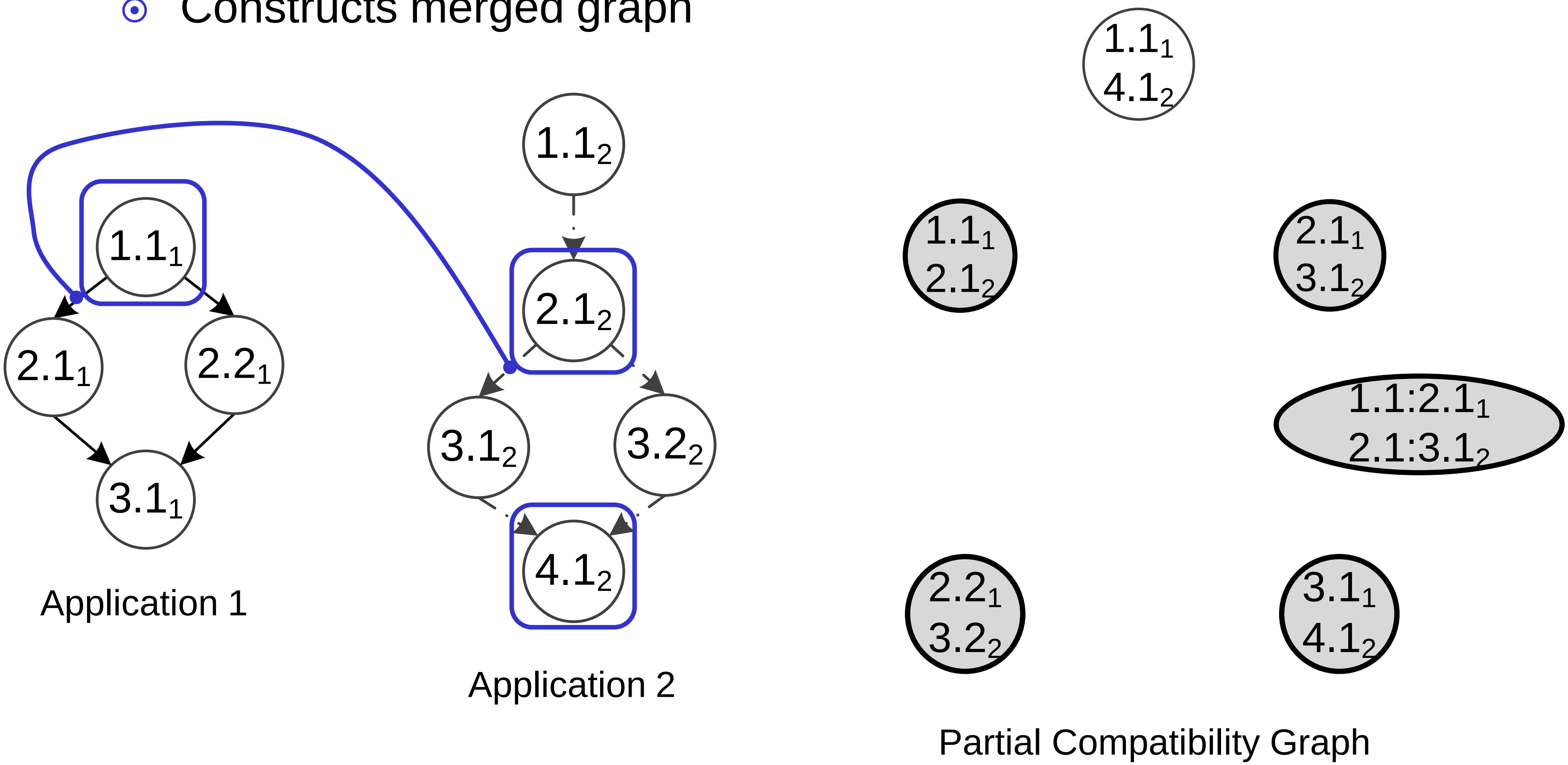


Application 2



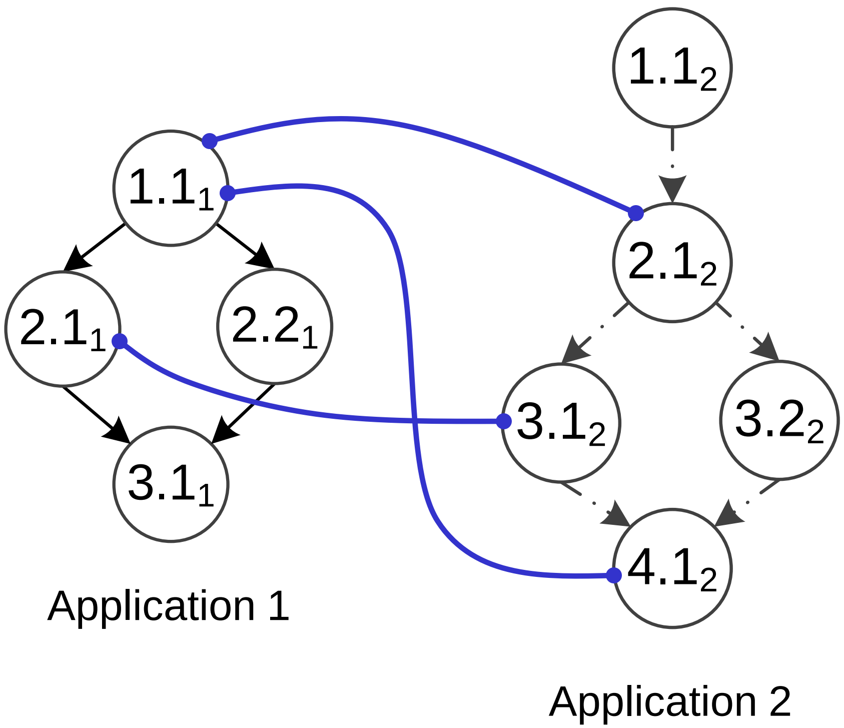
Heuristic MaxC (Max. Weight Clique)

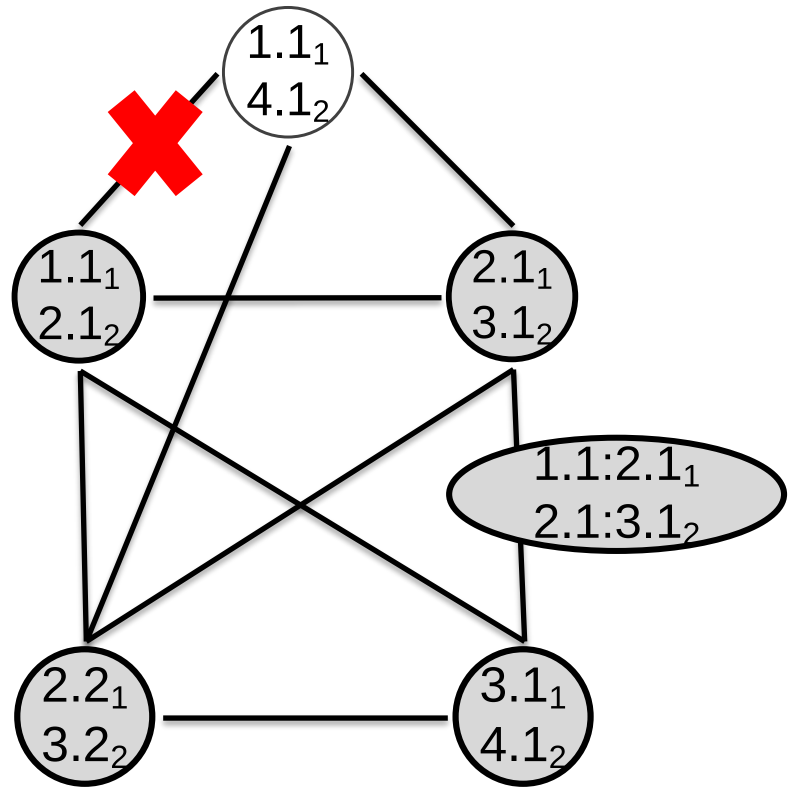
- Finds optimal merging
  - Creates a compatibility graph
  - Finds maximum clique of compatibility graph
  - Constructs merged graph



Heuristic MaxC (Max. Weight Clique)

- Finds optimal merging
  - Creates a compatibility graph
  - Finds maximum clique of compatibility graph
  - Constructs merged graph

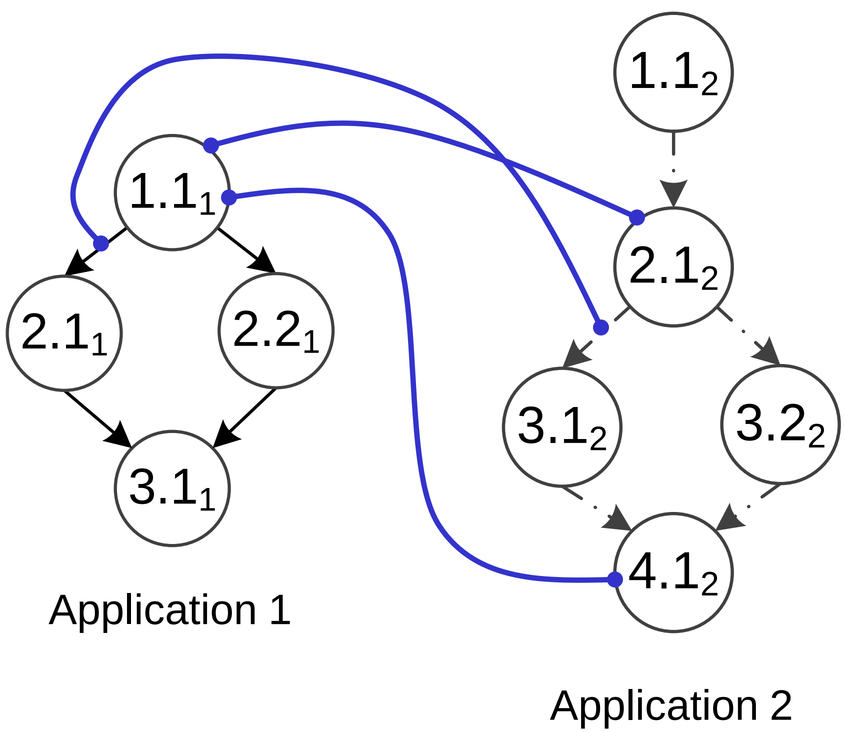


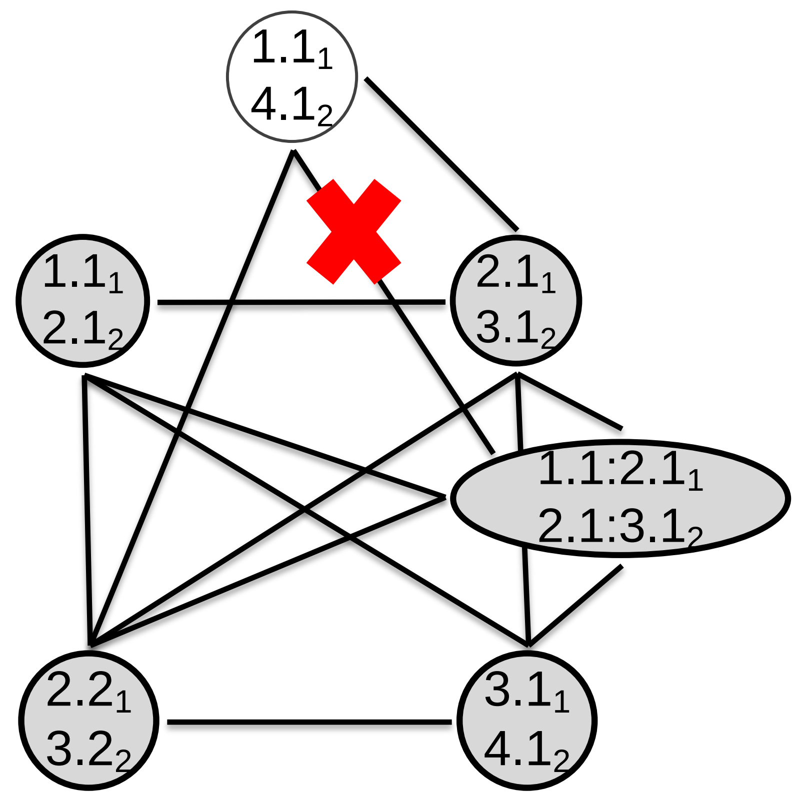


Partial Compatibility Graph

Heuristic MaxC (Max. Weight Clique)

- Finds optimal merging
  - Creates a compatibility graph
  - Finds maximum clique of compatibility graph
  - Constructs merged graph

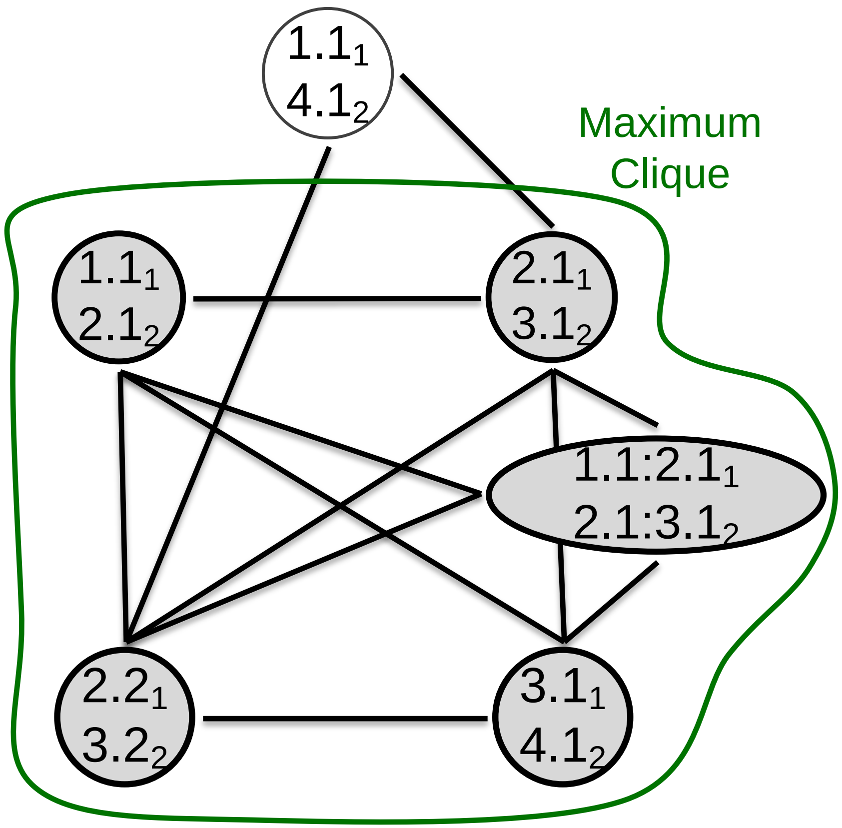




Partial Compatibility Graph

Heuristic MaxC (Max. Weight Clique)

- Finds optimal merging
  - Creates a compatibility graph
  - Finds maximum clique of compatibility graph
  - Constructs merged graph

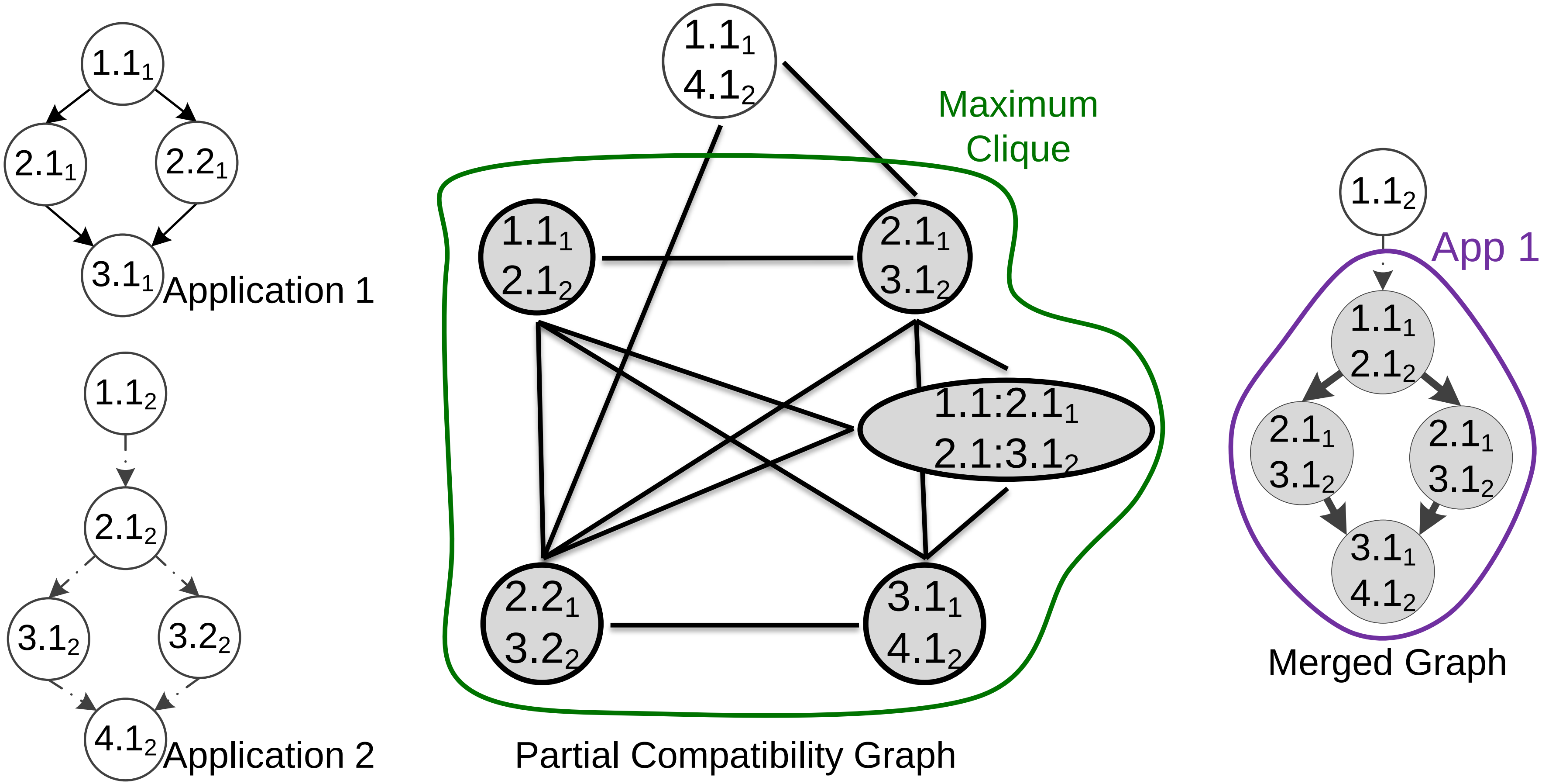


Partial Compatibility Graph

1. Each node has a weight which indicates area saving for that particular merging
2. Area saving is calculated using the cost functions provided by the designer

Heuristic MaxC (Max. Weight Clique)

- Finds optimal merging
  - Creates a compatibility graph
  - Finds maximum clique of compatibility graph
  - Constructs merged graph



Experimental Setup

- Benchmarks

Application	# Nodes	# Edges
JPEG Enc	7	9
JPEG Dec	5	6
MP3 Enc	5	5
FFT	12	12
BF	12	12
TDE	13	12
Syn1	14	15
Syn2	14	15
Syn3	17	20

- Pipelined MPSoCs
  - Used LX3 processors with queue interface from Tensilica
- Cliquer tool to find maximum weight clique

Results (Nodes)

Merge	# Nodes			
	No Merge	MaxS	MaxN	MaxC
JPEGenc/dec	12	9	7	7
JPEGenc/MP3enc	12	8	7	7
JPEGdec/MP3enc	10	6	5	5
JPEGenc/dec/MP3enc	17	9	7	7
FFT/BF	24	14	12	12
FFT/TDE	26	18	13	13
BF/TDE	26	17	13	13
FFT/BF/TDE	38	19	13	13
Syn1/Syn2	28	18	14	14
Syn/Syn3	31	21	17	-
Syn2/Syn3	31	26	17	-

Results (Edges)

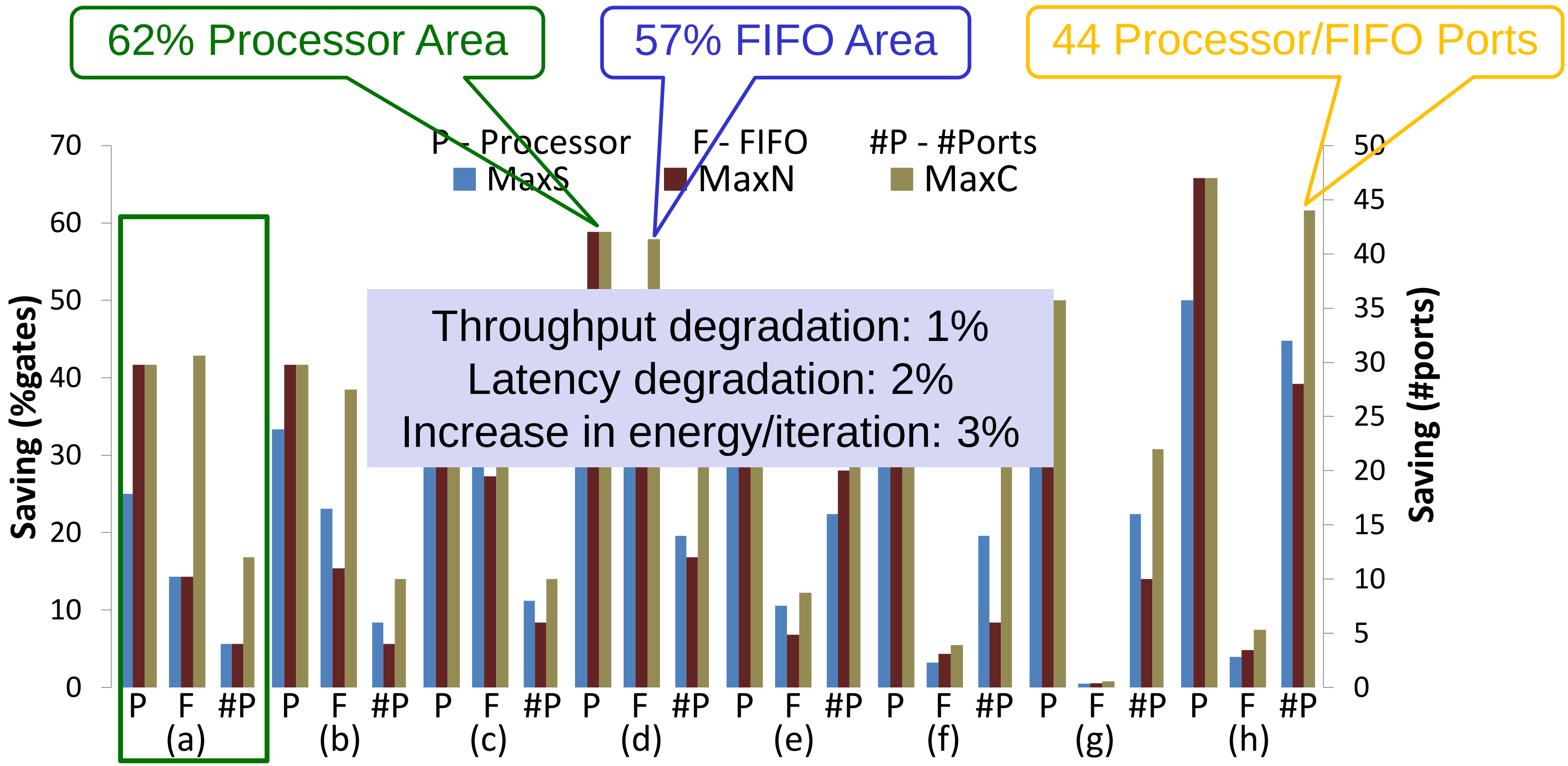
Merge	# Edges			
	No Merge	MaxS	MaxN	MaxC
JPEGenc/dec	14	12	12	8
JPEGenc/MP3enc	13	10	11	8
JPEGdec/MP3enc	11	7	8	6
JPEGenc/dec/MP3enc	19	12	13	8
FFT/BF	24	16	14	13
FFT/TDE	25	18	22	14
BF/TDE	25	17	20	14
FFT/BF/TDE	37	21	23	15
Syn1/Syn2	30	24	29	20
Syn/Syn3	35	25	32	-
Syn2/Syn3	35	23	31	-

Results (Running Time)

Merge	Time			
	No Merge	MaxS	MaxN	MaxC
JPEGenc/dec	-	<1s	<1s	1m
JPEGenc/MP3enc	-	<1s	<1s	1m
JPEGdec/MP3enc	-	<1s	<1s	1m
JPEGenc/dec/MP3enc	-	1s	2s	3m
FFT/BF	-	1s	1s	5m
FFT/TDE	-	1s	1s	5m
BF/TDE	-	1s	1s	5m
FFT/BF/TDE	-	1s	2s	12m
Syn1/Syn2	-	1s	1s	16h
Syn/Syn3	-	1s	1s	>4d
Syn2/Syn3	-	1s	1s	>4d

Results (Area Saving)

- Merging cost functions
  - Two nodes saves a processor
  - Two edges saves two FIFO ports + size of smaller FIFO



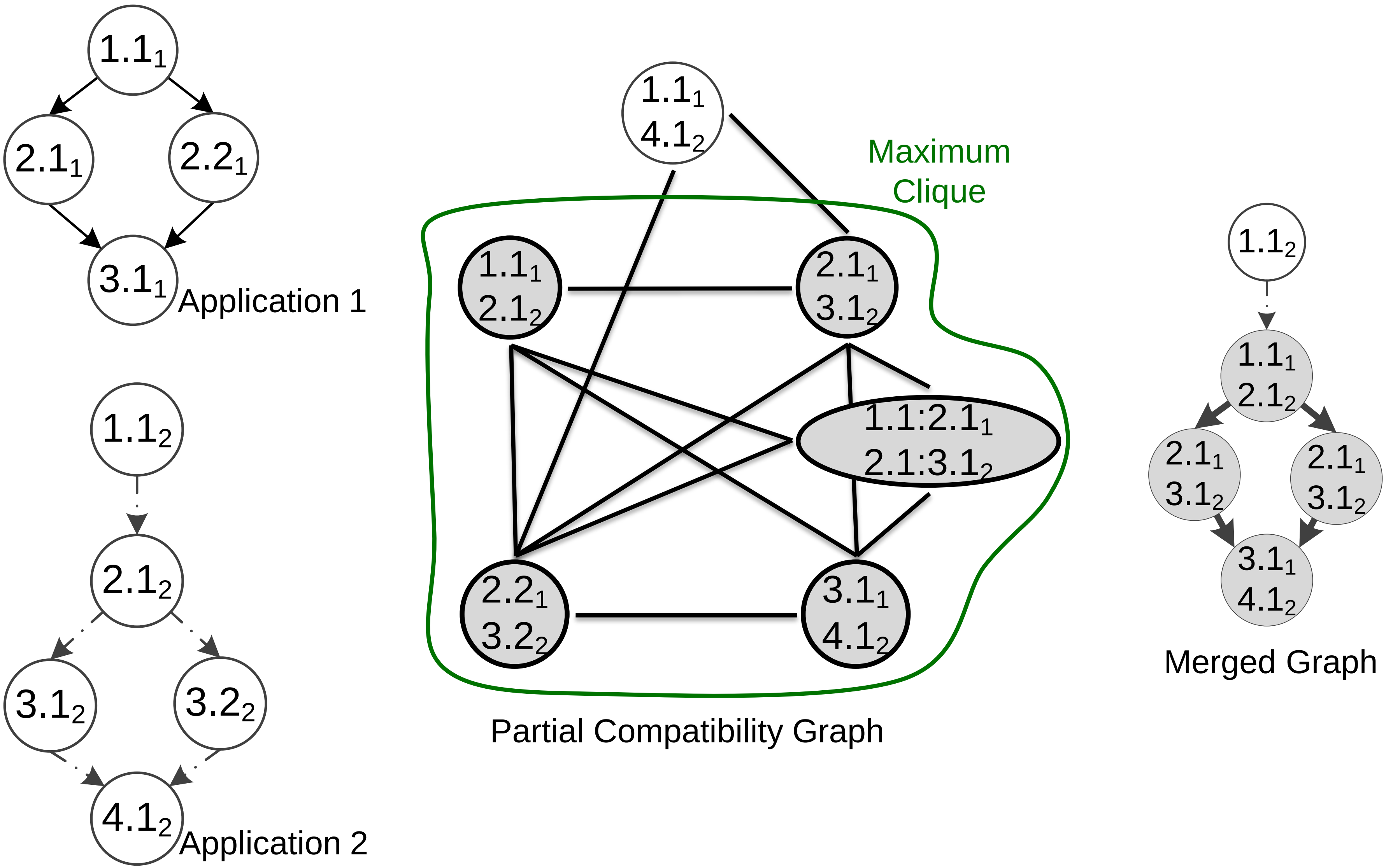
(a) JPEGenc/dec (b) JPEGenc/MP3enc (c) JPEGdec/MP3enc (d) JPEGenc/dec/MP3enc (e) FFT/BF (f) FFT/TDE (g) BF/TDE (h) FFT/BF/TDE

Conclusions

- Multi-mode pipelined MPSoCs can be designed by merging of application graphs
- The proposed heuristics saved up to
  - 62% processor area
  - 57% FIFO area
  - 44 processor/FIFO ports
- Miniscule degradation in performance and energy efficiency
- Future work
  - Consider memories
  - Consider code size
  - Consider simultaneously executing accelerators

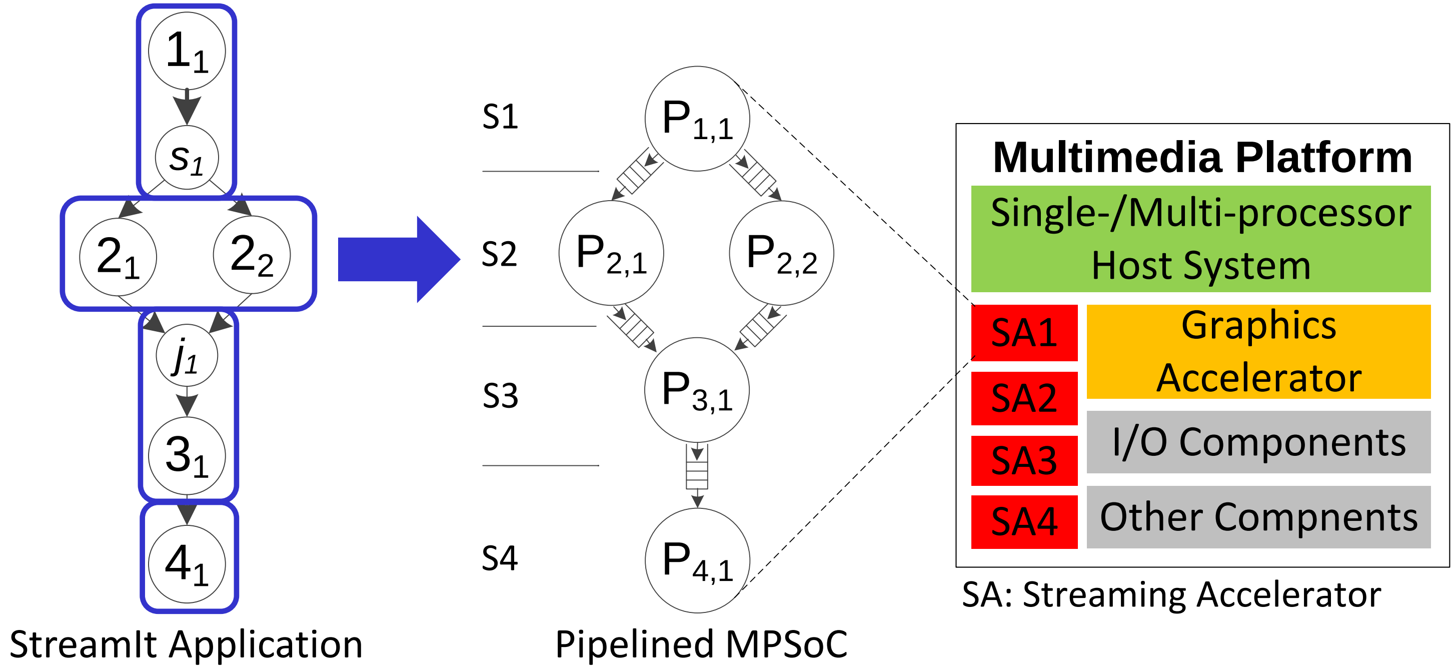
Thank You!

Heuristic MaxC (Max. Weight Clique)



Introduction

- Multimedia platforms are increasingly becoming heterogeneous:
  - Single-/Multi-processor host system
  - Graphics and **streaming accelerators**
- Streaming accelerators
  - Optimized at design-time for performance and energy efficiency
  - Based upon **pipelined MPSoCs** rather than ASICs in this work



Pipelined MPSoCs

- Pipelined MPSoCs
  - Task- and pipeline-level parallelisms (data-flow structure)
  - Data- and instruction-level parallelisms (SIMD and VLIW instructions)