Branch-and-Bound Style Resource Constrained Scheduling using Efficient Structure-Aware Pruning

Mingsong Chen, Saijie Huang, Geguang Pu

Software Engineering Institute, East China Normal University, China

Prabhat Mishra

CISE Department, University of Florida, USA

華東師範大學
EAST CHINA NORMAL UNIVERSITY

UF UNIVERSITY of FLORIDA

Outline

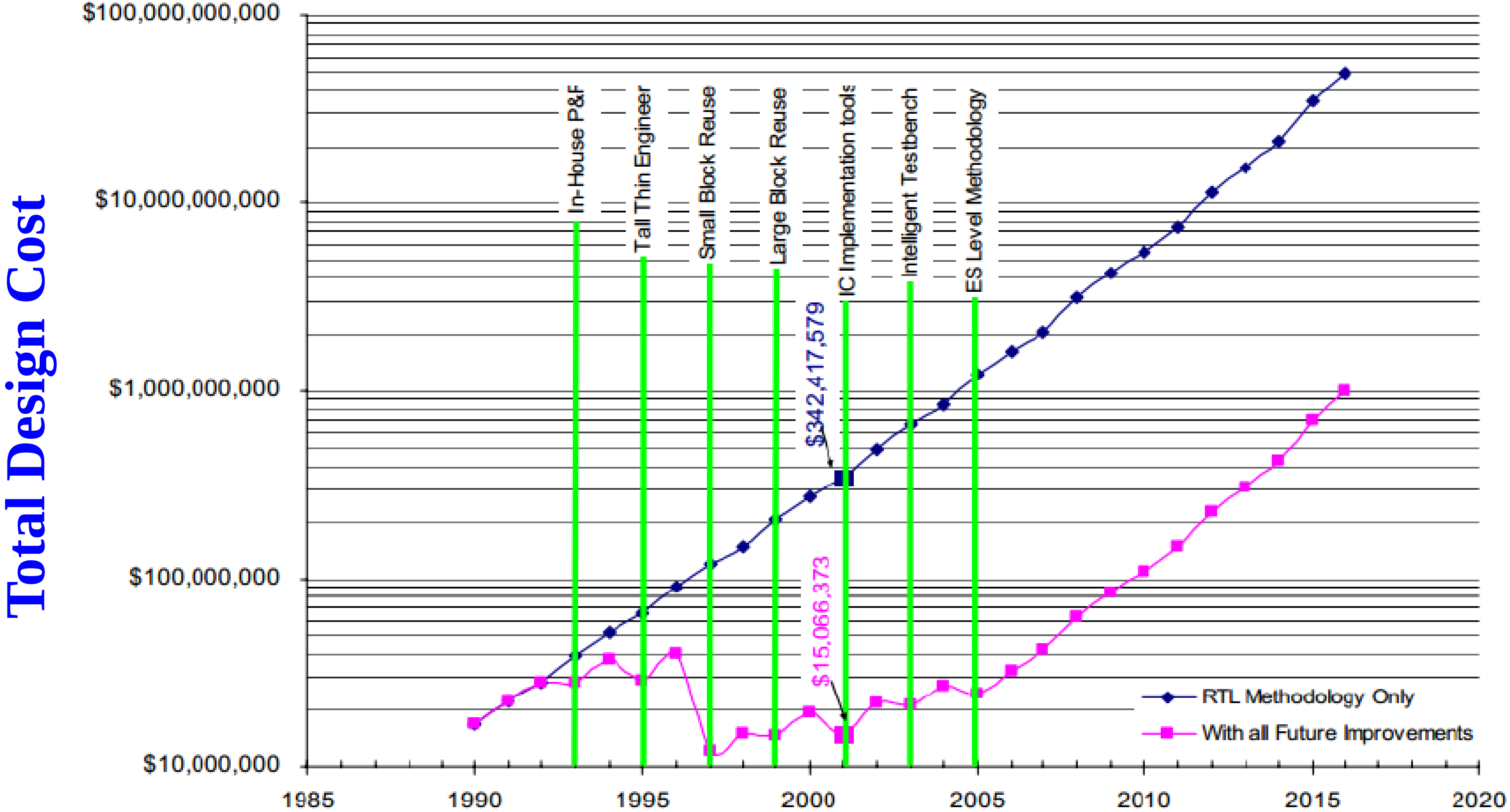
- Introduction
- RCS using Branch-and-Bound Approaches
  - Graph–based Notations
  - BULB Approach
- Our Structure-Aware Pruning Approach
  - Motivation
  - Level-Bound Pruning Heuristics
  - HLS Scheduling using Our Approach
- Experiments
- Conclusion

Outline

- Introduction
- RCS using Branch-and-Bound Approaches
  - Graph–Based Notations
  - BULB Approach
- Our Structure-Aware Pruning Approach
  - Motivation
  - Level-Bound Pruning Heuristics
  - HLS Scheduling using Our Approach
- Experiments
- Conclusion

SoC Design Cost Model

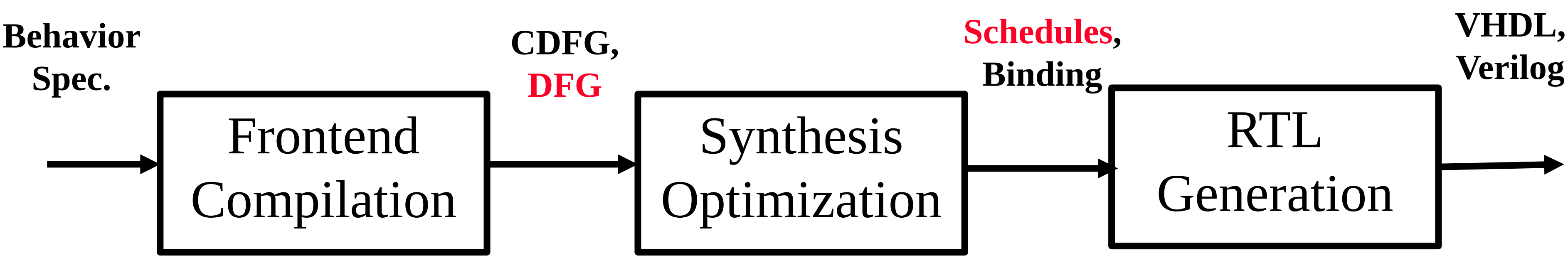
Big Savings by using ESL Methodology



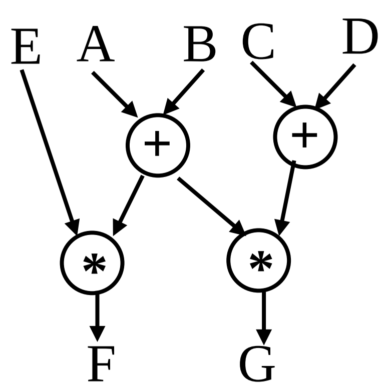
Rising cost of IC design and effect of CAD tools (Courtesy: Andrew Kahng, UCSD and SRC)

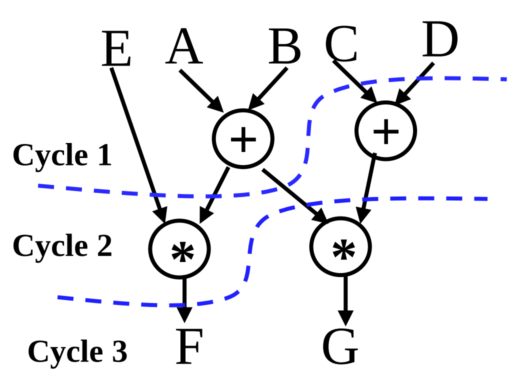
High Level Synthesis

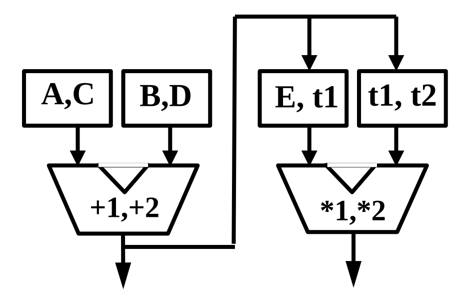
- **Convert ESL specification to RTL implementation, and satisfy the design constraints.**
  - **Input:** Behavior specifications (C, SystemC, etc.), and design constraints (delay, power, area, etc.)
  - **Output:** RTL implementation (datapath, controller)




```
int Sample(){
var  A,B,C,D,E,F,G : int;
Read(A, B, C, D, E);
F = E * (A + B);
G = (A + B) * (C + D);
……
}
```

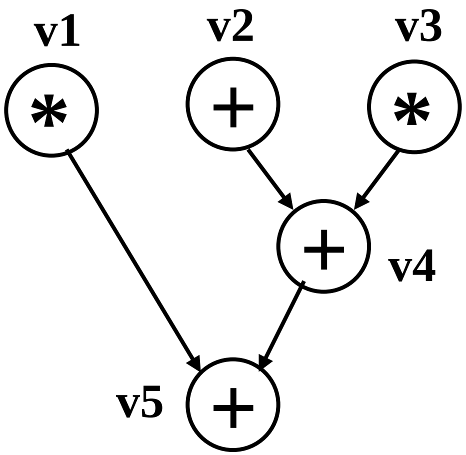




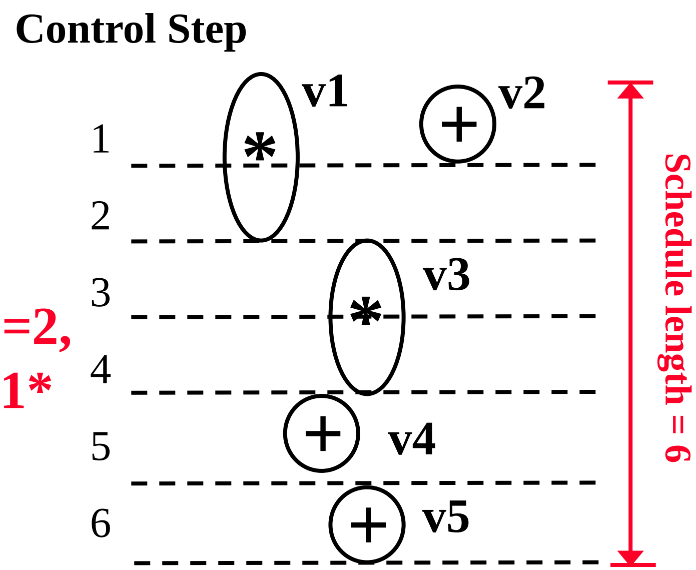


Resource Constrained Scheduling

- **Various resource constraints (e.g., functional units, power, …).**
- **Scheduling is a mapping of operations to control steps**
  - Given a DFG and a set of resource constraints, RCS tries to find a (optimal) schedule with minimum overall control steps.



Constraints:
Delay(+)=1, Delay(*)=2,
functional units: 1+, 1*



RCS is NP-Complete. RCS should take care of
1) Operation precedence. 2) Resource sharing constraints

Basic Solutions

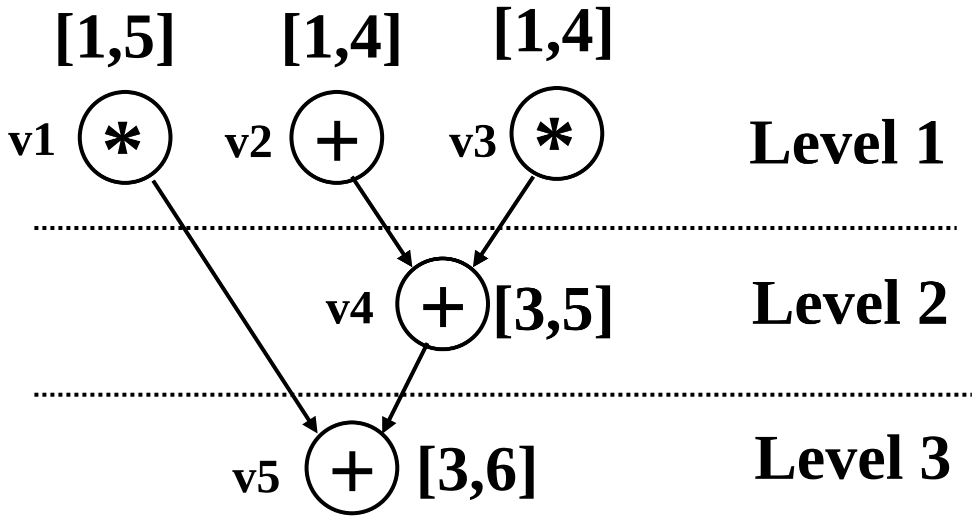
- **Non-optimal heuristics**
  - **Force Directed Scheduling**
  - **List scheduling**
    - ✓ **Pros: Fast to get near-optimal results**
    - ✓ **Cons: schedules may not be tight**
- **Optimal approaches**
  - **Integer linear programming**
    - ✓ **Pros: easy modeling**
    - ✓ **Cons: scalability, cannot handle non-integer time**
  - **Branch-and-bound**
    - ✓ **Pros: can prune the fruitless search space efficiently**
    - ✓ **Cons: only investigate the bound length information,**

Outline

- Introduction
- RCS using Branch-and-Bound Approaches
  - Graph–Based Notations
  - BULB Approach
- Our Structure-Aware Pruning Approach
  - Motivation
  - Level-Bound Pruning Heuristics
  - HLS Scheduling using Our Approach
- Experiments
- Conclusion

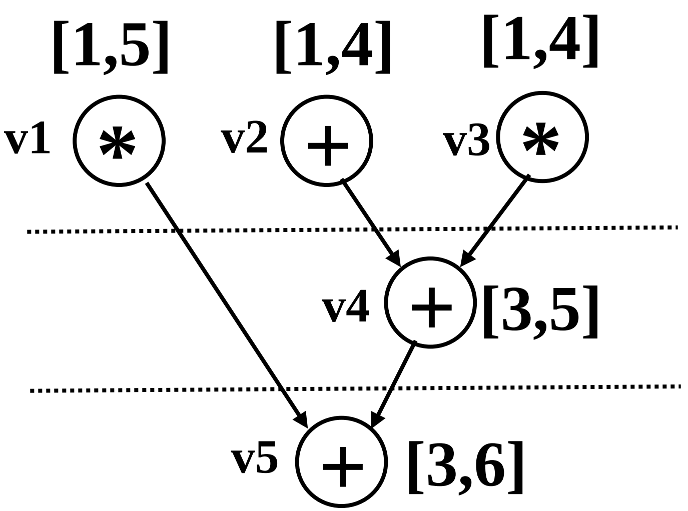
Graph-Based Notations

- [ASAP, ALAP] intervals indicate the earliest and latest start time of operations
- Input operations and output operations
- Level(op) indicates the longest length from some input operations to the current operation op

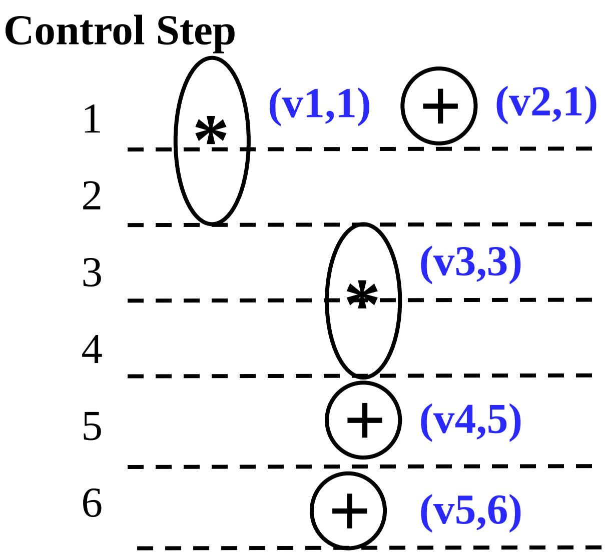


Scheduling Using [ASAP, ALAP]

- A **schedule** is a binary relation of operations and corresponding dispatching control step
  - E.g., {(v1, 1), (v2, 1), (v3, 3), (v4, 5), (v5, 5)}



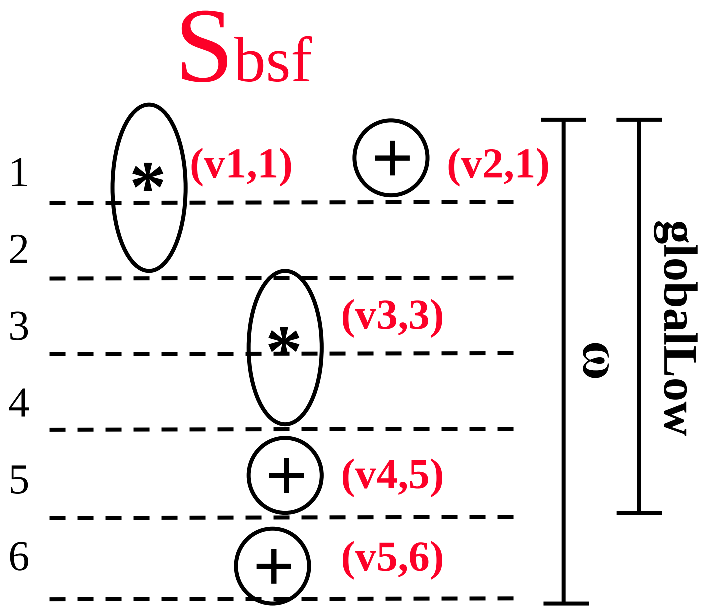
Constraints:
Delay(+)=1,
Delay(*)=2,
1+, 1*

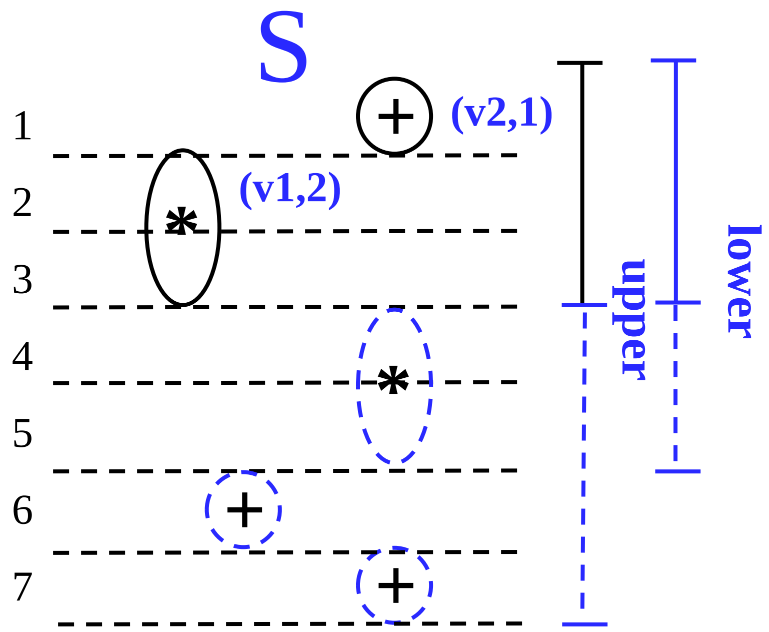


- Based on [ASAP, ALAP], naively enumerating all the possibilities can be extremely time consuming
  - The operations are enumerated in a specific order
  - Each operation are enumerated from ASAP to ALAP

Branch and Bound Style RCS (BULB)

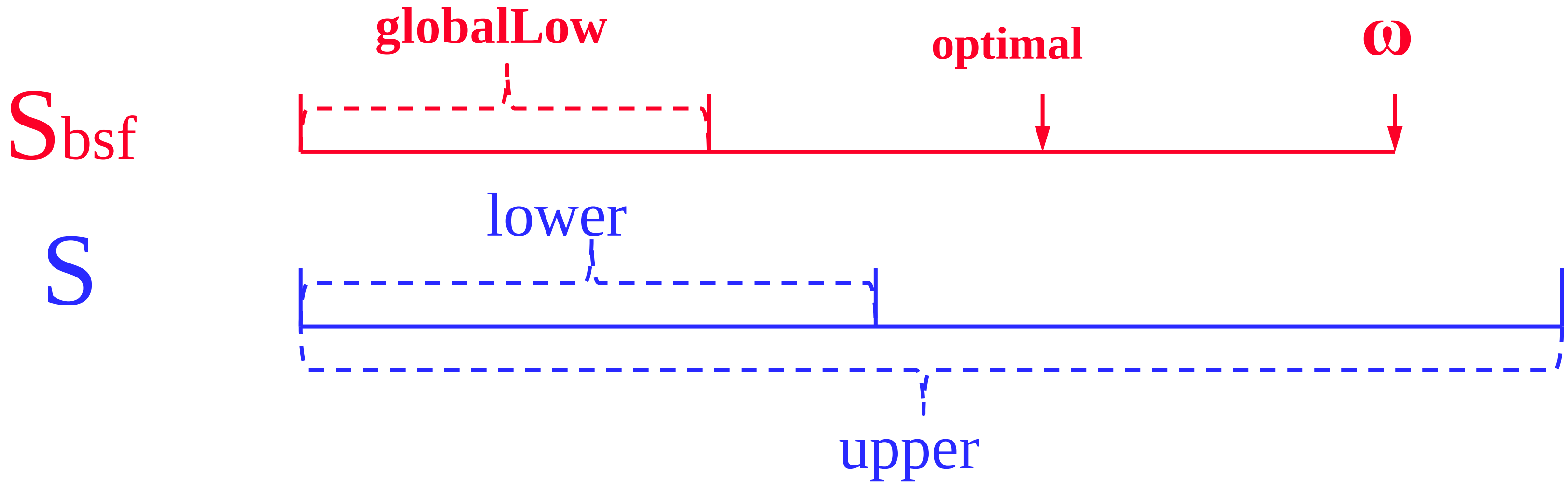
- BULB tries to prune fruitless enumerations.
- B&B approach keeps two data structure regarding bound information.
  - S_{bsf} , best complete schedule searched so far
  - S, current incomplete schedule





Pruning in BULB

- **Pruning [lower > ω]**
- **Termination [globalLow == ω or fully explored]**
- **Substitution [if(upper < ω) ω = upper]**
- **Backtrack [operations are all enumerated]**



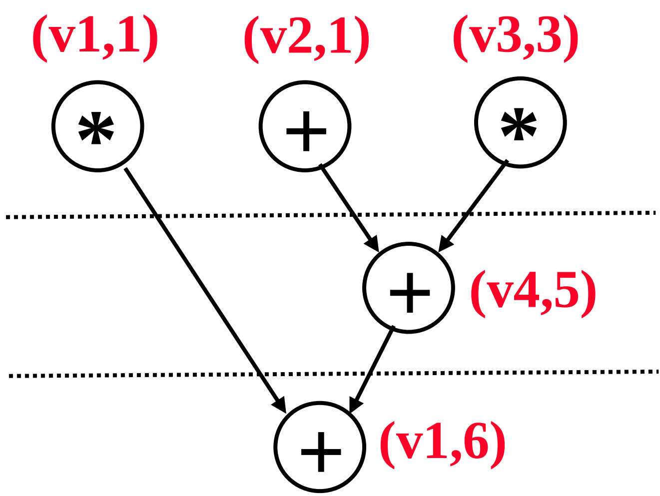
Based on the bound information, no further pruning can be conducted for current B&B approaches when **ω is in [lower, upper].**

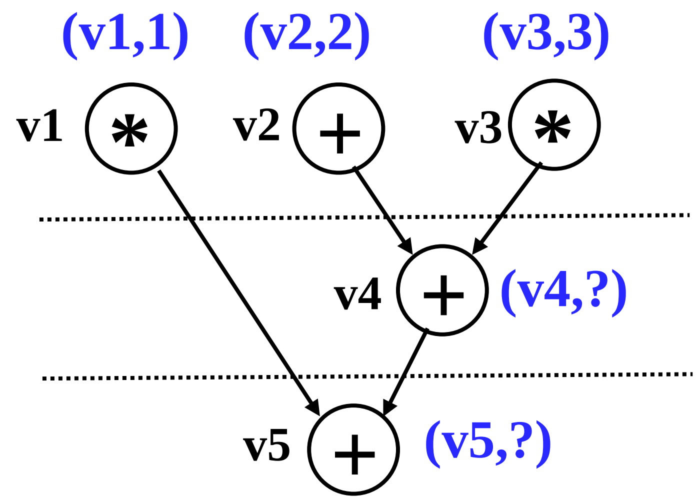
Outline

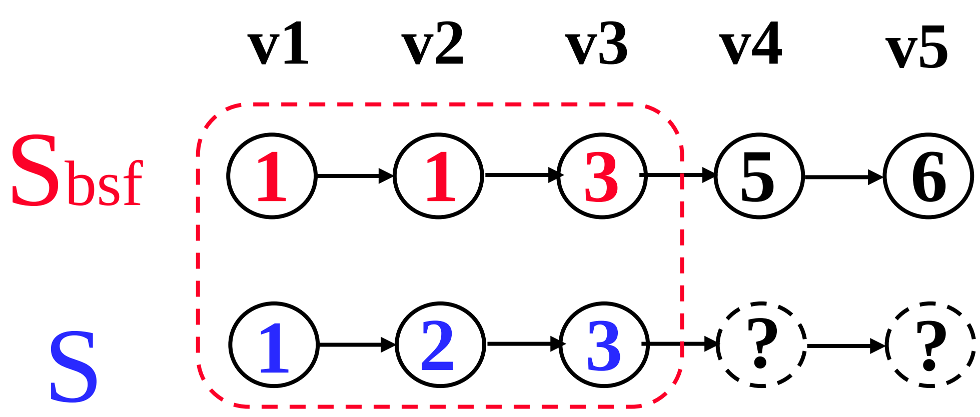
- Introduction
- RCS using Branch-and-Bound Approaches
  - Graph–Based Notations
  - BULB Approach
- Our Structure-Aware Pruning Approach
  - Motivation
  - Level-Bound Pruning Heuristic
  - HLS Scheduling using Our Approach
- Experiments
- Conclusion

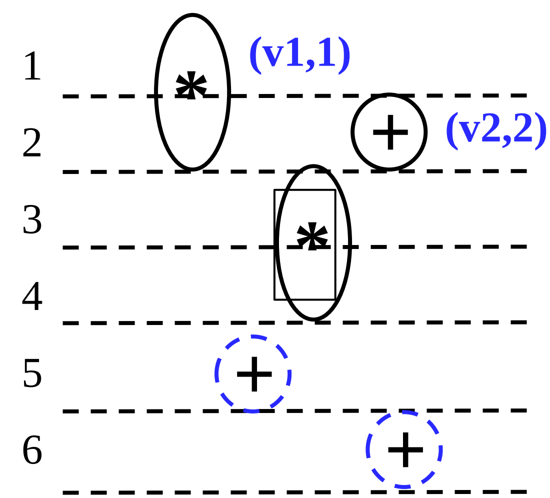
Motivation

- **Pruning based on the structural information of the best schedule (i.e., S_{opt}) searched so far.**



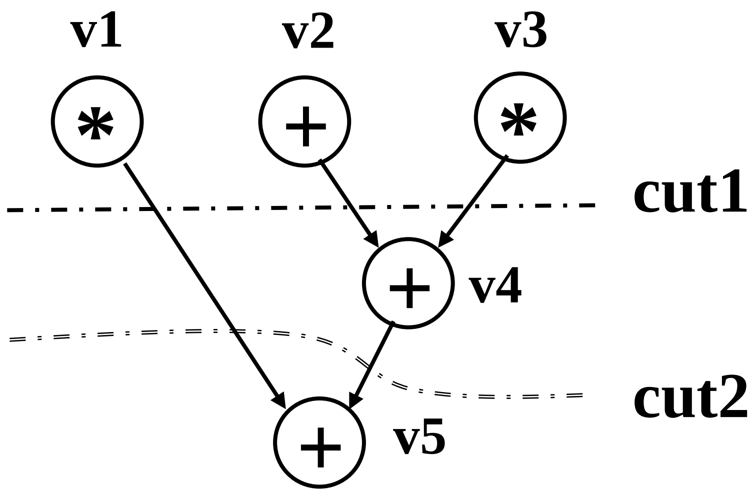






Cut and Complete Level

- A **cut** is an edge set which can separate a DFG into two parts, one part contains all input operations, the other one contains all output operations.
- The **k_{th} complete level** of a cut is a set node, which are adjacent input nodes of all the edges across **k_{th} level** and **$(k+1)_{th}$ level** .



1st Complete level : {v1,v2,v3}

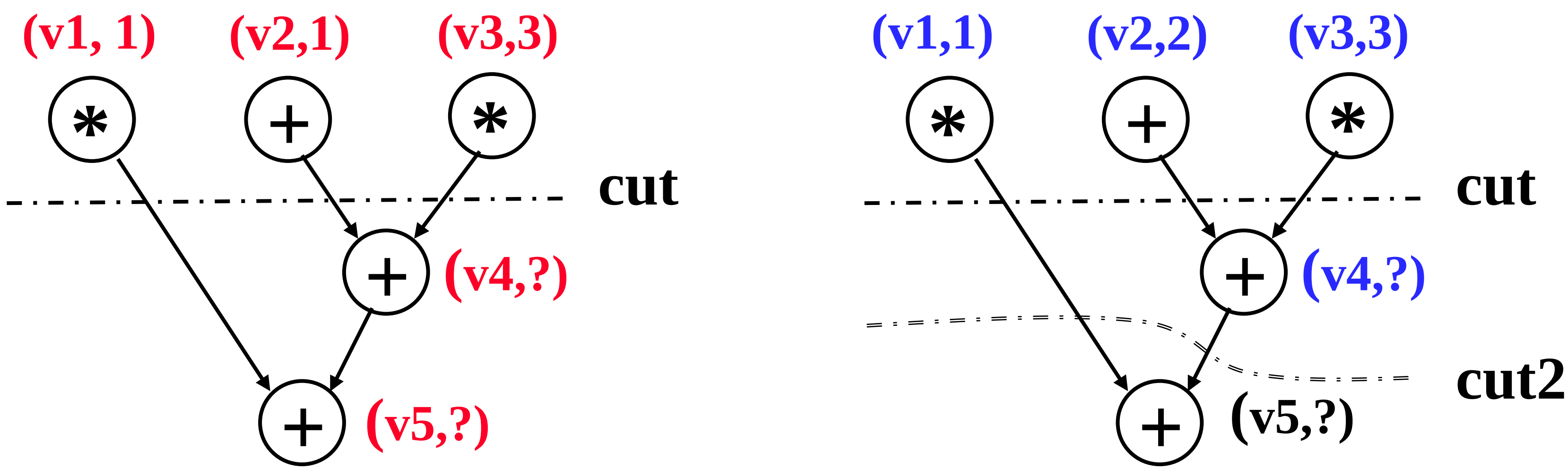
2nd Complete level : {v1,v4}

Level-Bound Pruning

- **Let OP_k be the operation set of k_{th} complete level.**

  The level-bound pruning can be enabled when the following conditions hold:

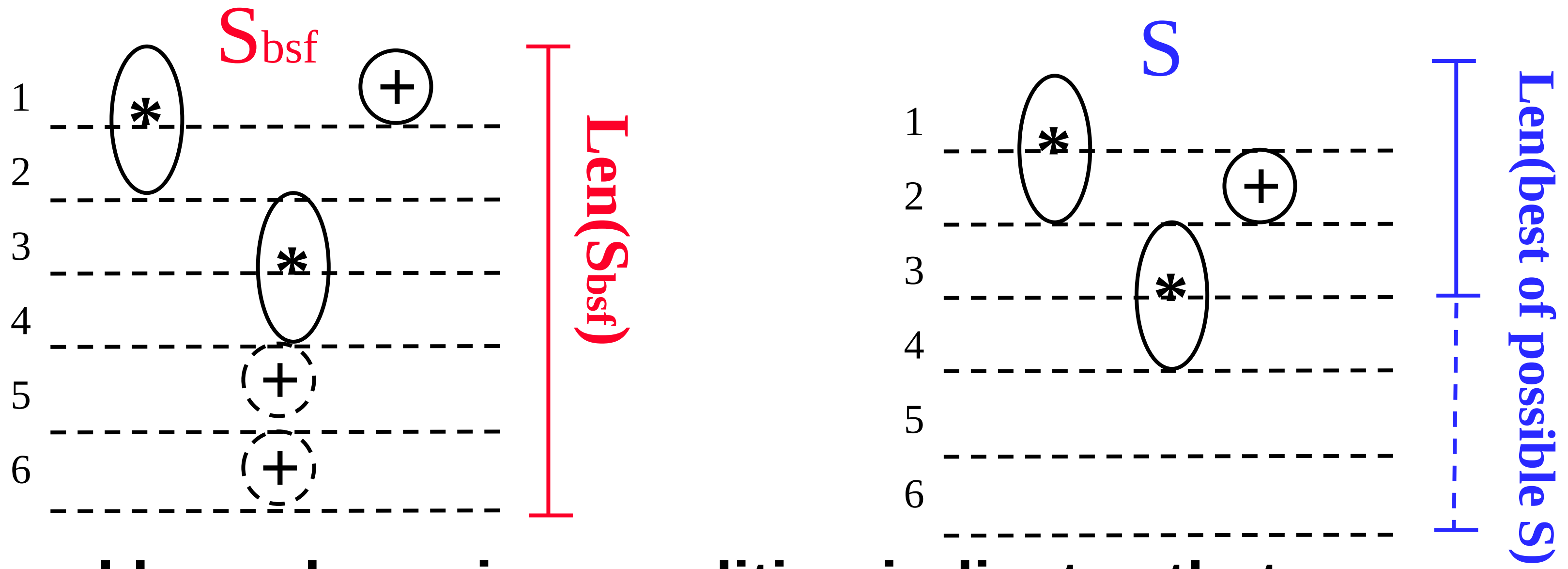
  1. $\forall\, op_i,\ op_i \in OP_k \rightarrow S(op_i) > 0;$
  2. $\forall\, op_i,\ op_i \in OP_k \rightarrow S_{bsf}(op_i) \leq S(op_i);$
  3. $\exists\, opi,\ opi \in OP_k \rightarrow S_{bsf}(op_i) < S(op_i).$



S_{bsf} $\quad Len(S_{bsf}) \leq Len(S)$ $\quad S$

Basic Proof of Level-Bound Pruning

1. Enumeration of operations starts from ASAP to ALAP
2. When a level bound pruning condition holds, for S_{bsf} , all the combination of operation dispatching under the complete level has been fully explored.
3. S_{bsf} is the best schedule founded in all combinations in 2.



4. Level bound pruning condition indicates that

$Len(S_{bsf}) <= Len(\text{best of all possible } S)$

Therefore, the enumeration can be safely pruned.

Structure-Aware Pruning approach

```
Struture-arwarePruning (D, i, N, Sbsf, S, ω) {
  if i≤n then{
    for step = ASAP(opi) to ALAP(opi){
      1. if LevelBound(S, Sbsf, opi)  return (Sbsf, ω);
      if precedence(opi) ∧ resAvailable(step, type(opi)){
          2. recalculate lower and upper;
          if upper < ω{   3. ω = upper;
                          4. Sbsf = ListScheduling(opi);
                          5. if ω == globalLow(D) Terminate;
                          6.  UpdateALAP(); }
          if lower < ω{   7. S(opi) = step;
                          8. ResOccupy(step, type(opi), delay(opi));
                          9. Struture-arwarePruning (D, i+1, N, Sbsf, S, ω);
                         10. ResRestore(step, type(opi), delay(opi)); }
      }
}
```

Outline

- Introduction
- RCS using Branch-and-Bound Approaches
  - Graph–Based Notations
  - BULB Approach
- Our Structure-Aware Pruning Approach
  - Motivation
  - Level-Bound Pruning Heuristics
  - HLS Scheduling using Our Approach
- Experiments
- Conclusion

Benchmarks & Settings

- Used benchmarks from *MediaBench*.
- BULB & our approach are implemented using C++.
- All the experiments were conducted on a Linux machine with Intel 2.0GHz CPU and 3G RAM.
- Setting of functional units:

Functional Unit	Operation class	Delay (unit)	Power (unit)	Energy (unit)	Area (unit)
ADD/SUB	+/-	1	**10**	10	**10**
MUL/DIV	*/	2	**20**	40	**40**
MEM	LD/STR	1	**15**	15	**20**
Shift	<</>>	1	**10**	10	**5**
Others	…	1	**10**	10	**10**

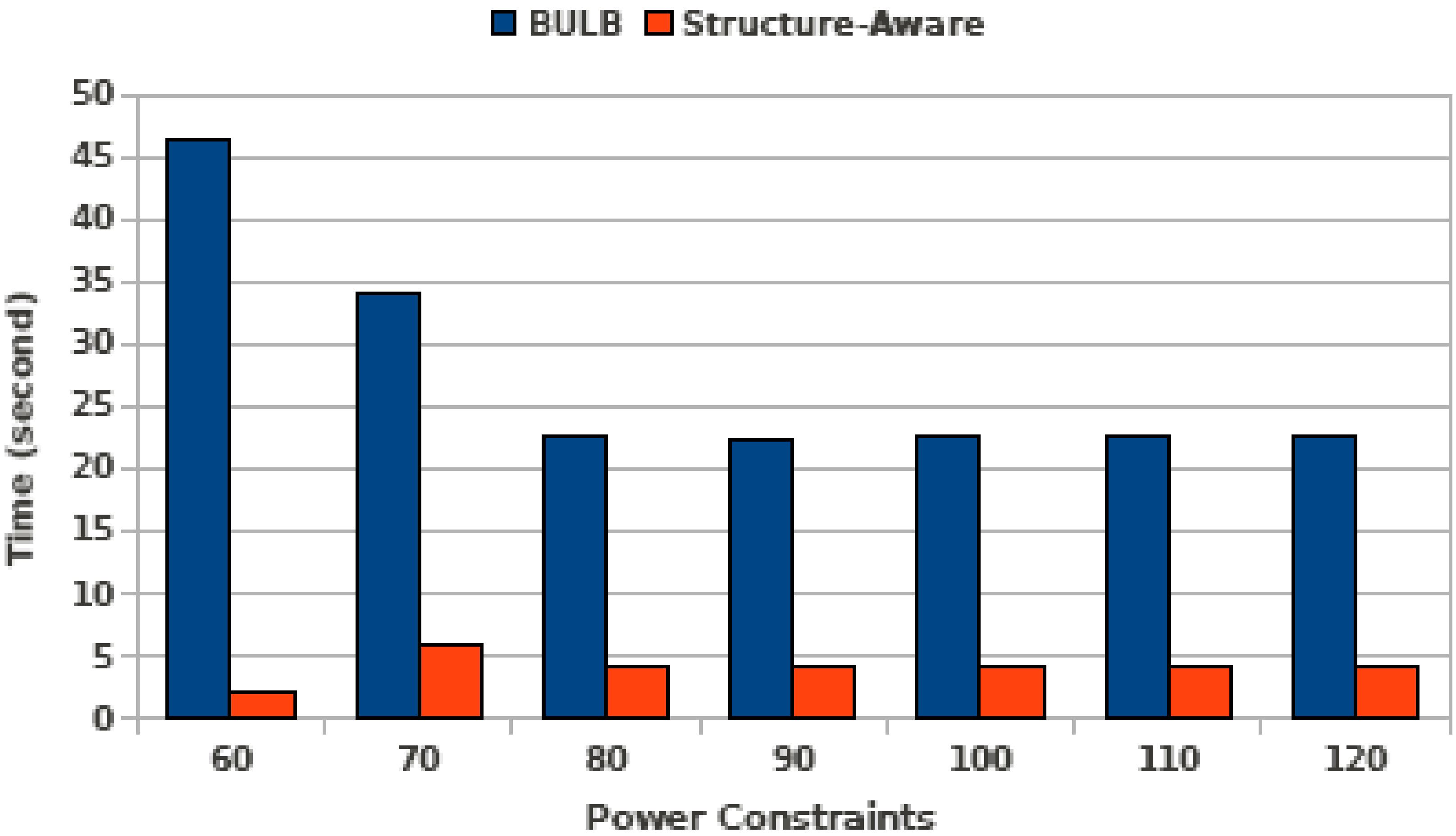
Results under Functional Constraints

Design					CPLEX ILP (sec.)	BULB (sec.)	Ours (sec.)	Speedup
name	# of +, ×	lower	upper	*c-step*				
ARFilter	1, 3	14	16	16	Timeout	0.34	0.14	2.43
	1, 4	14	16	16	Timeout	0.86	0.26	3.31
	1, 5	14	16	16	Timeout	0.85	0.26	3.27
	2, 3	14	15	15	2.32	0.01	$<$0.01	$>$1.00
Collapse	2, 1	22	23	22	Timeout	Timeout	234.76	$>$15.33
	2, 2	21	23	21	Timeout	Timeout	Timeout	NA
Cosine	1, 2	28	29	28	Timeout	105.67	23.38	4.52
	2, 2	20	23	20	Timeout	611.75	65.91	9.28
	3, 3	16	17	16	Timeout	0.02	$<$0.01	$>$2.00
Feedback	4, 4	13	14	13	Timeout	171.67	154.94	1.11
	4, 5	13	NA	NA	Timeout	Timeout	Timeout	NA
	5, 5	13	14	13	Timeout	5.53	4.96	1.11
FDCT	1, 2	26	27	26	Timeout	38.05	23.52	1.62
	2, 2	18	22	18	Timeout	210.22	18.67	11.26
	2, 3	14	17	14	Timeout	21.26	4.12	5.16
	2, 4	13	15	13	Timeout	4.31	2.00	2.16
	2, 5	13	14	13	Timeout	0.99	0.61	1.62
	3, 4	11	13	11	Timeout	0.64	0.51	1.25
	4, 4	11	12	11	Timeout	0.13	0.02	6.50

RCS efforts are significantly improved:

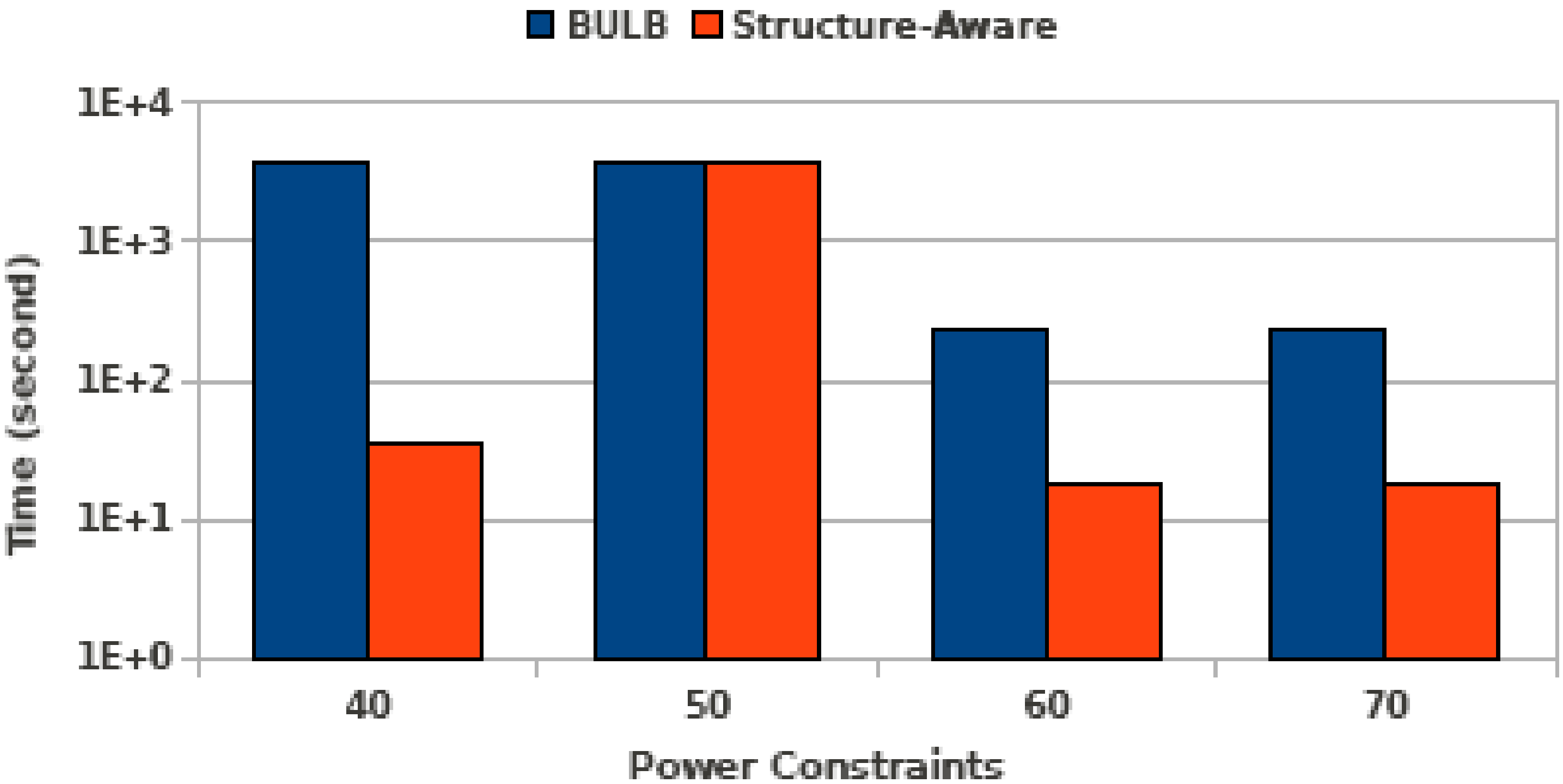
- BULB approach outperforms ILP approach
- Our approach can still get up to 15X speedup against BULB

Scheduling Using Area of 140 Units



When power is 60, up to 22x speedup.

Scheduling Using Area of 100 Units



When power is 40, up to 101x speedup.

Conclusions

- **RCS is a major bottleneck in HLS**
  - Branch-and-bound approaches are promising for optimal resource-constrained scheduling.
- **Proposed a structure-aware pruning heuristic**
  - Based on structural scheduling information of explored optimal schedule candidates
  - Synergy with state-of-the-art B&B methods
- **Successfully applied on various benchmark with different resource constraints**
  - Significant reduction in overall RCS efforts

Thank you !