Optimization for Buffer and Splitter Insertion in AQFP Circuits with Local and Group Movement

Bing-Huan Wu, Wai-Kei Mak
National Tsing Hua University

Outline

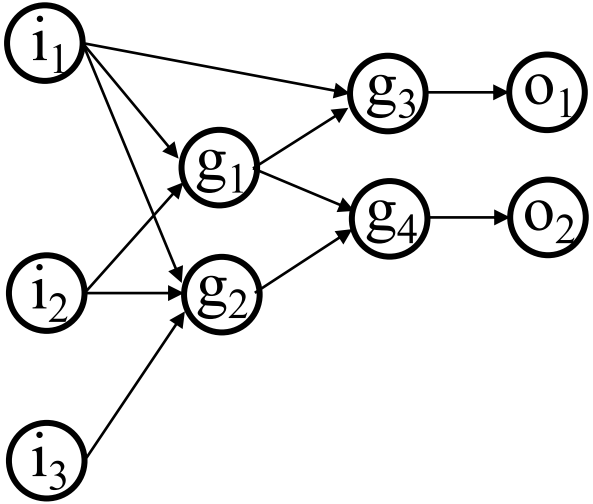
- AQFP Circuit
- Problem Formulation
- Optimization Methods
- Overall Flow
- Experimental Results
- Conclusion

AQFP Circuit

- *Adiabatic quantum-flux parametron* (AQFP) is a superconducting technology with extremely low power consumption compared to traditional CMOS technology
- Special constraints
  1) Fanout branching: each logic gate and buffer can only drive one output, and each splitter can drive multiple outputs no more than the given splitting capacity *sp*
  2) Path balancing: for each node, its input values must be released by nodes at its previous level
- Assumptions
  1) Primary inputs (PIs) are aligned at the same level
  2) Primary outputs (POs) are aligned at the same level
  3) PIs should satisfy the fanout branching constraint

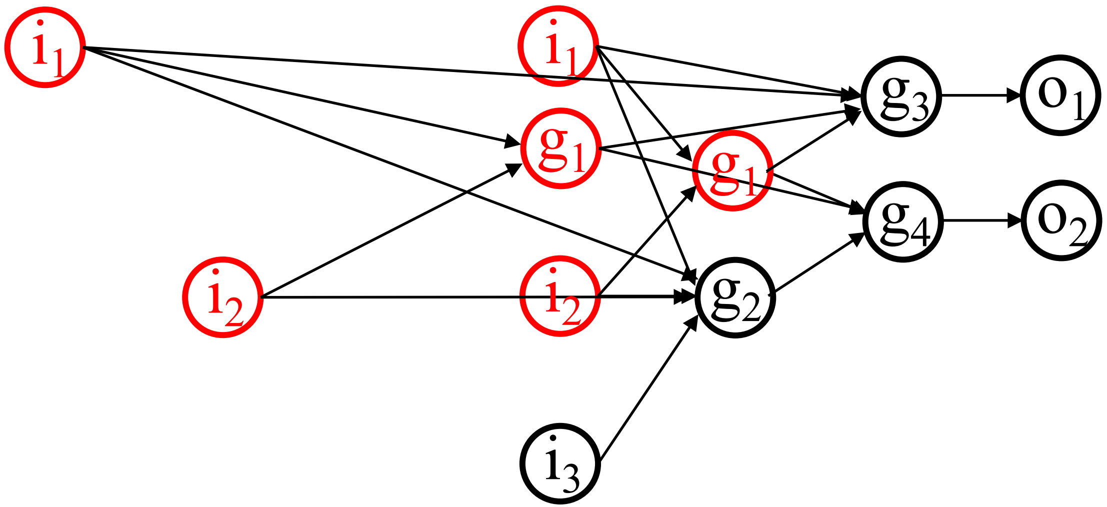
AQFP Circuit

➢ Example (a netlist we want to convert to an legal AQFP circuit)



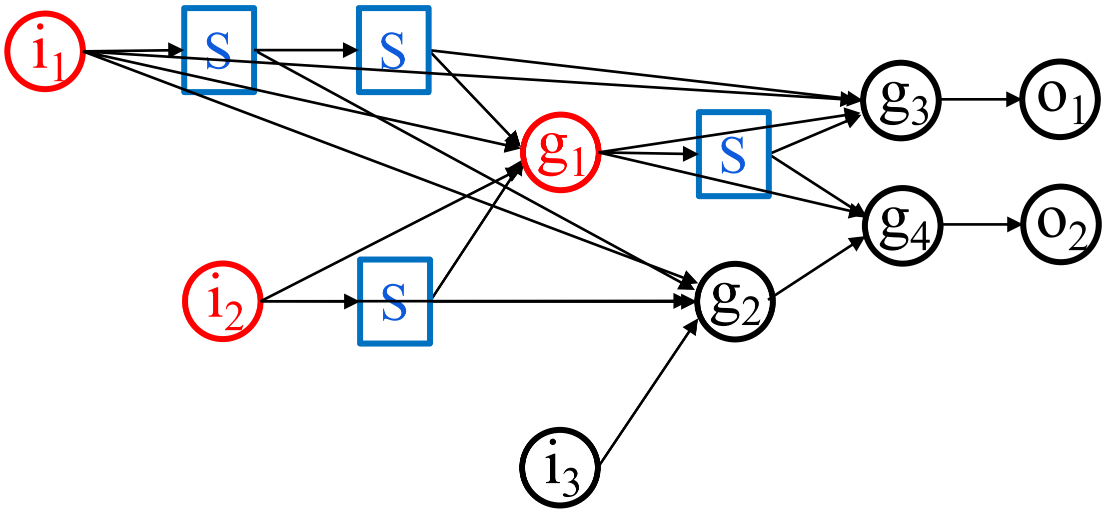
AQFP Circuit

- Example (solving fanout branching constraint, assume $sp = 2$)



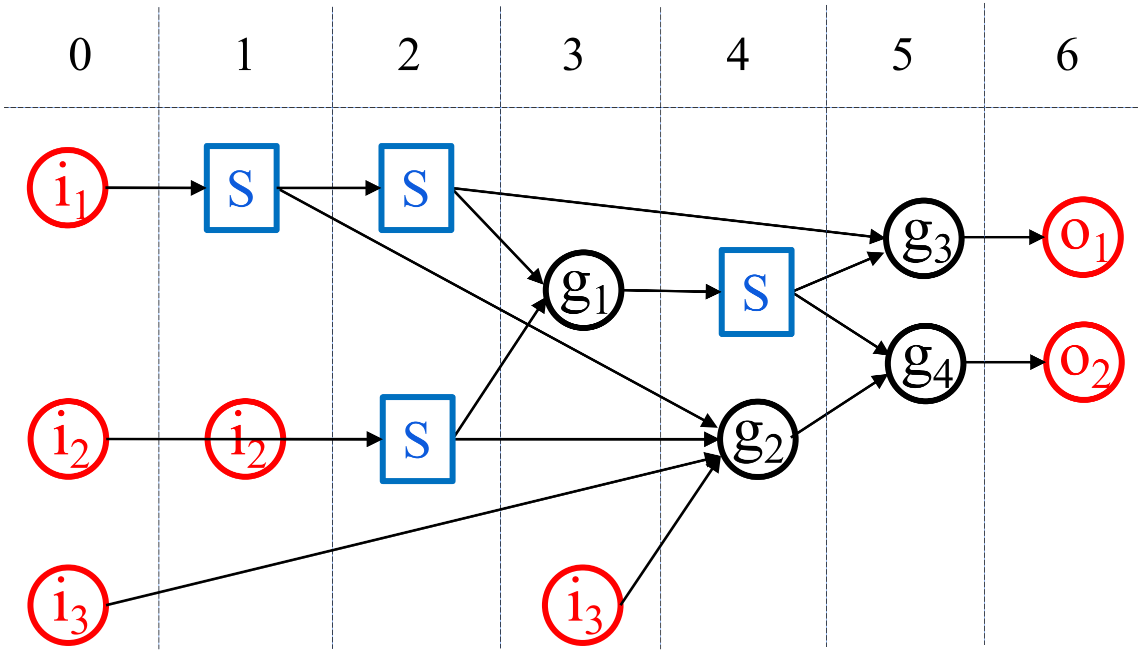
AQFP Circuit

- Example (solving fanout branching constraint, assume $sp = 2$)



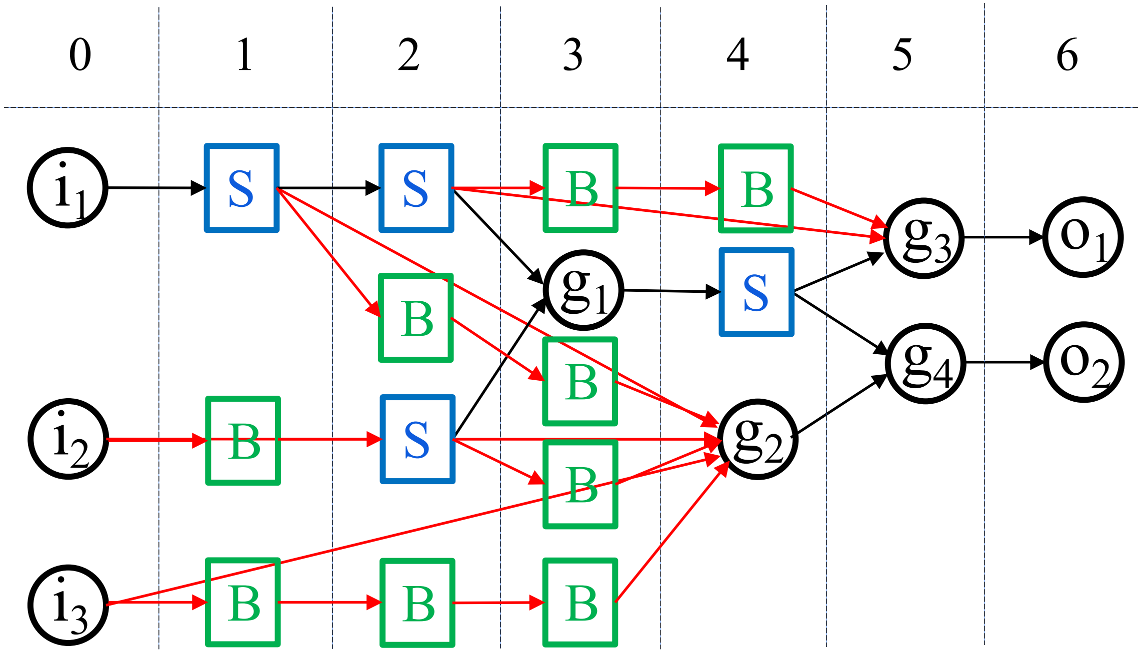
AQFP Circuit

- Example (PI alignment and PO alignment)



AQFP Circuit

- Example (solving path balancing constraint)



Problem Formulation

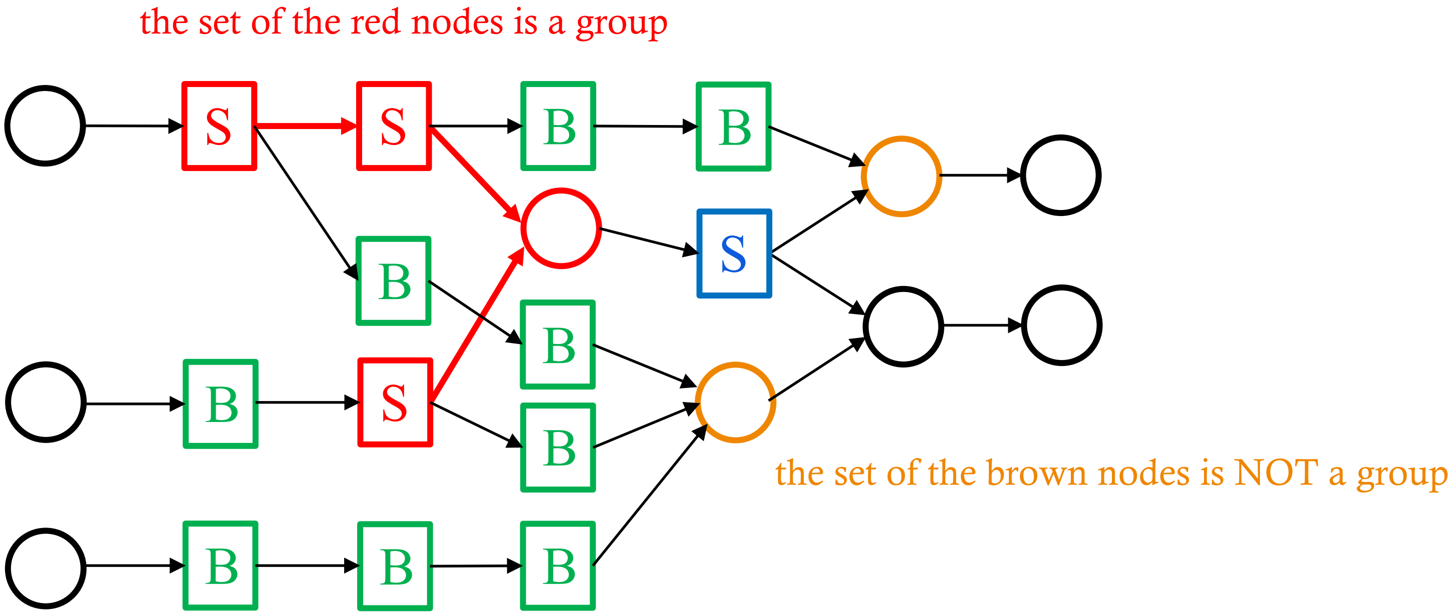
- Input
  1) A netlist N
  2) Splitting capacity sp
- Output
  - An AQFP circuit C which exhibits the same functionality as N and satisfies all the constraints and assumptions
- Target
  - Minimize the cost (number of inserted buffers and splitters) of the output circuit

Optimization Methods

- Backward Movement Optimization
  - Backward Group Movement
  - Input Modifying Backward Movement
- Forward Movement Optimization
  - Forward Group Movement
  - Buffer Integrating Forward Movement
- Level Perturbation
  - Flexibility-driven Branching Tree
  - Forced Movement
  - Fanout-pair Level Adjustment

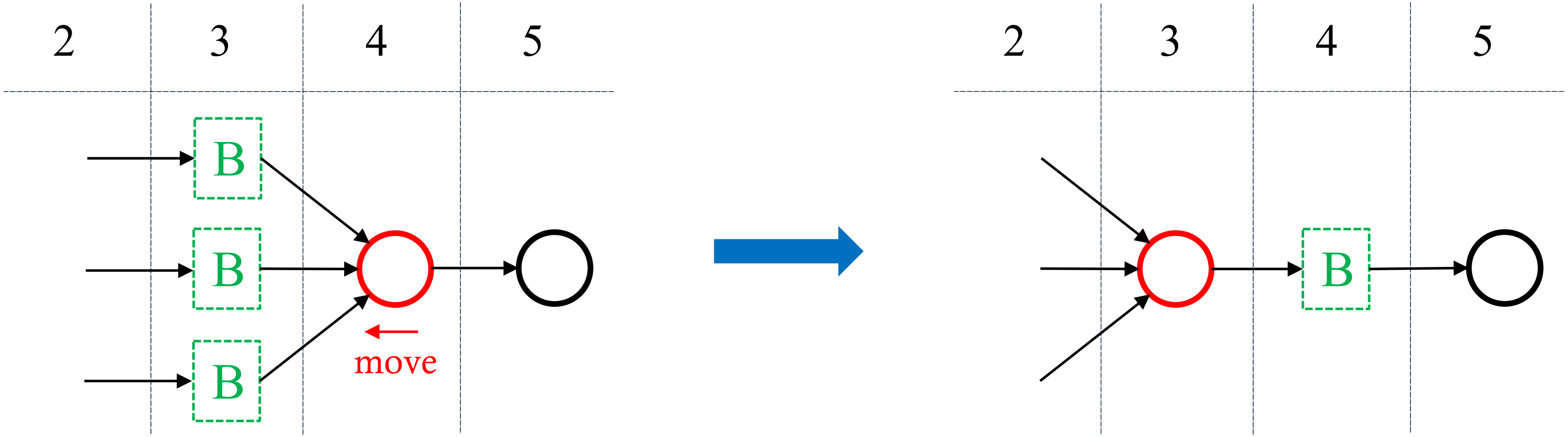
Backward Group Movement

➢ A *group* is a set of gate(s) and splitter(s) which form a connected subgraph in an AQFP circuit



Backward Group Movement

- In backward movement optimization, we want to move logic gates backward for reducing the number of buffers

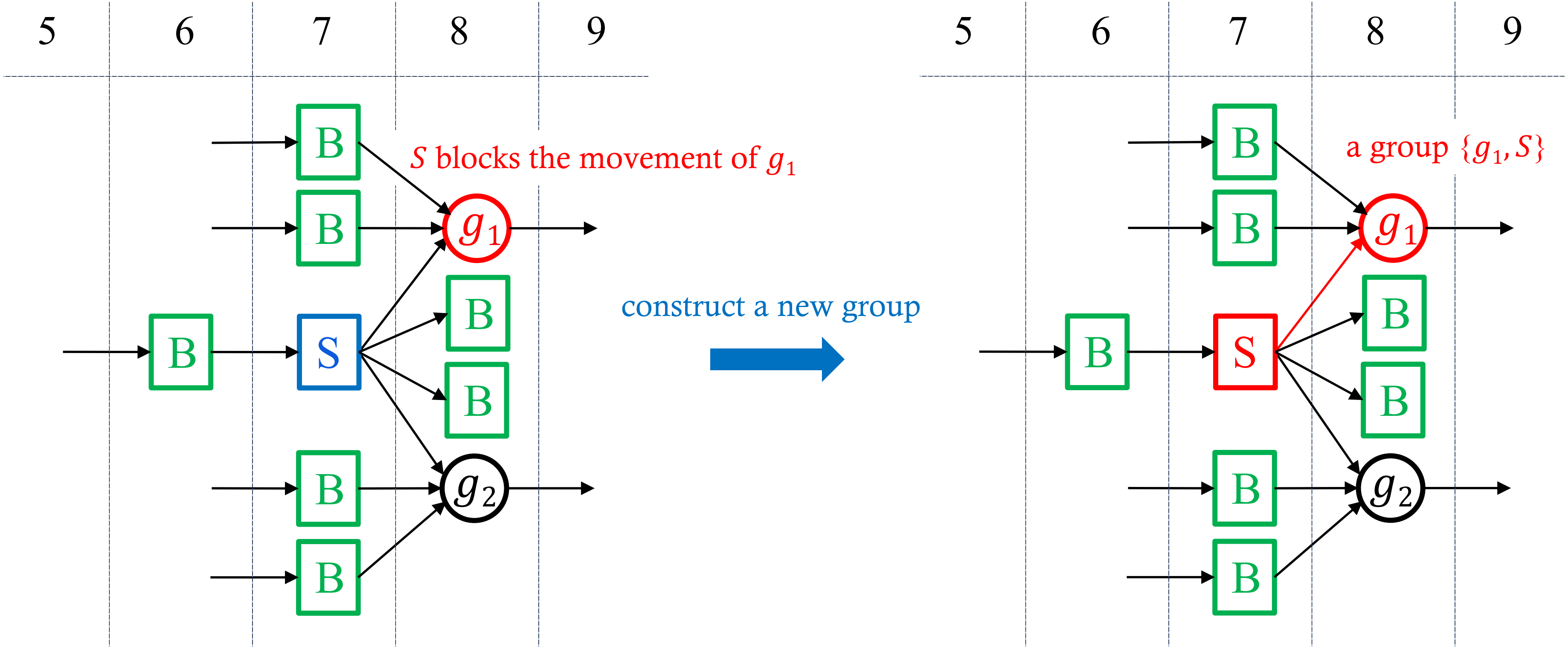


Backward Group Movement

- However, the movement of a gate g may be blocked by other splitters, gates or previously created groups
  - Cluster g and "the gates, the splitters, and the groups blocking g's movement" into a new group, and perform group movement
  - Only the movement which do not increase the number of buffers will be accepted
  - The method for deciding the order of the movements of gates is detailed in the paper

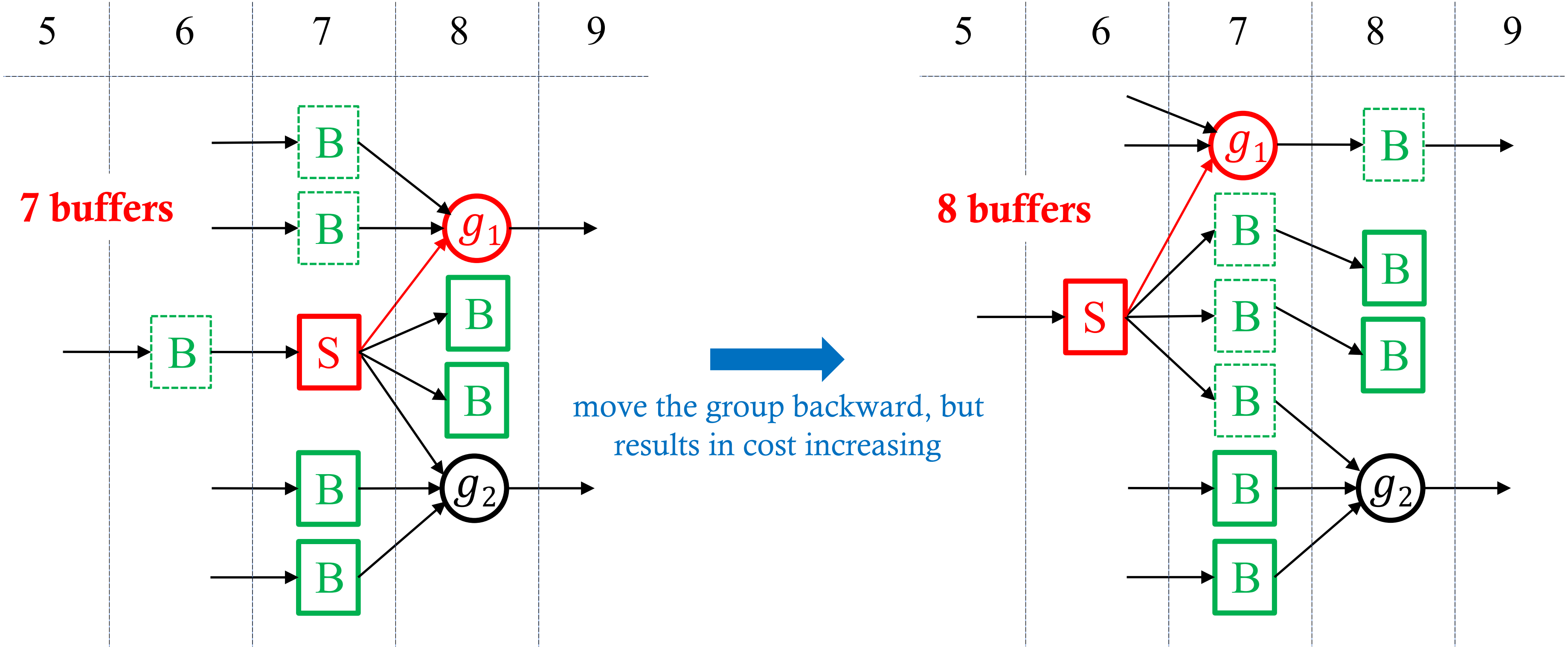
Backward Group Movement

➢ If we first try to move g_1 backward



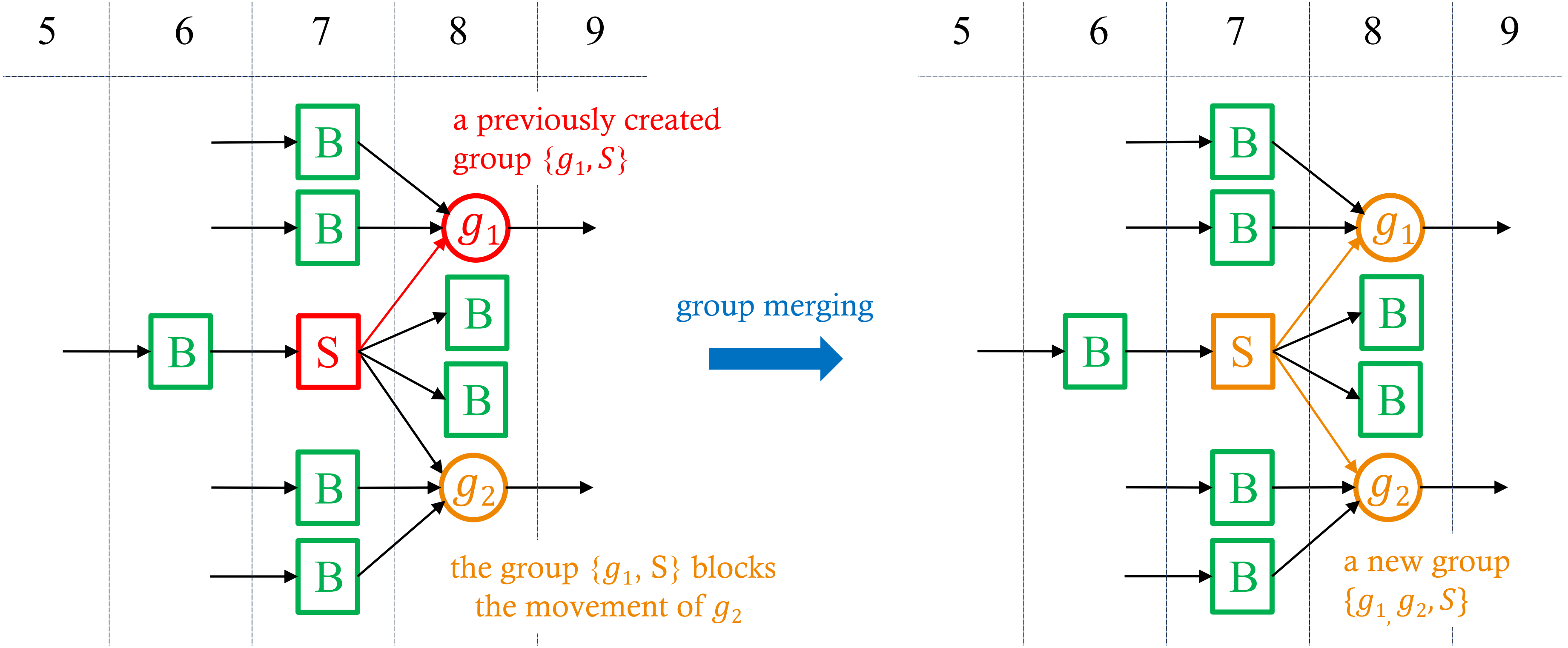
Backward Group Movement

- Try to move the group $\{g_1, S\}$ backward, but the result will NOT be accepted



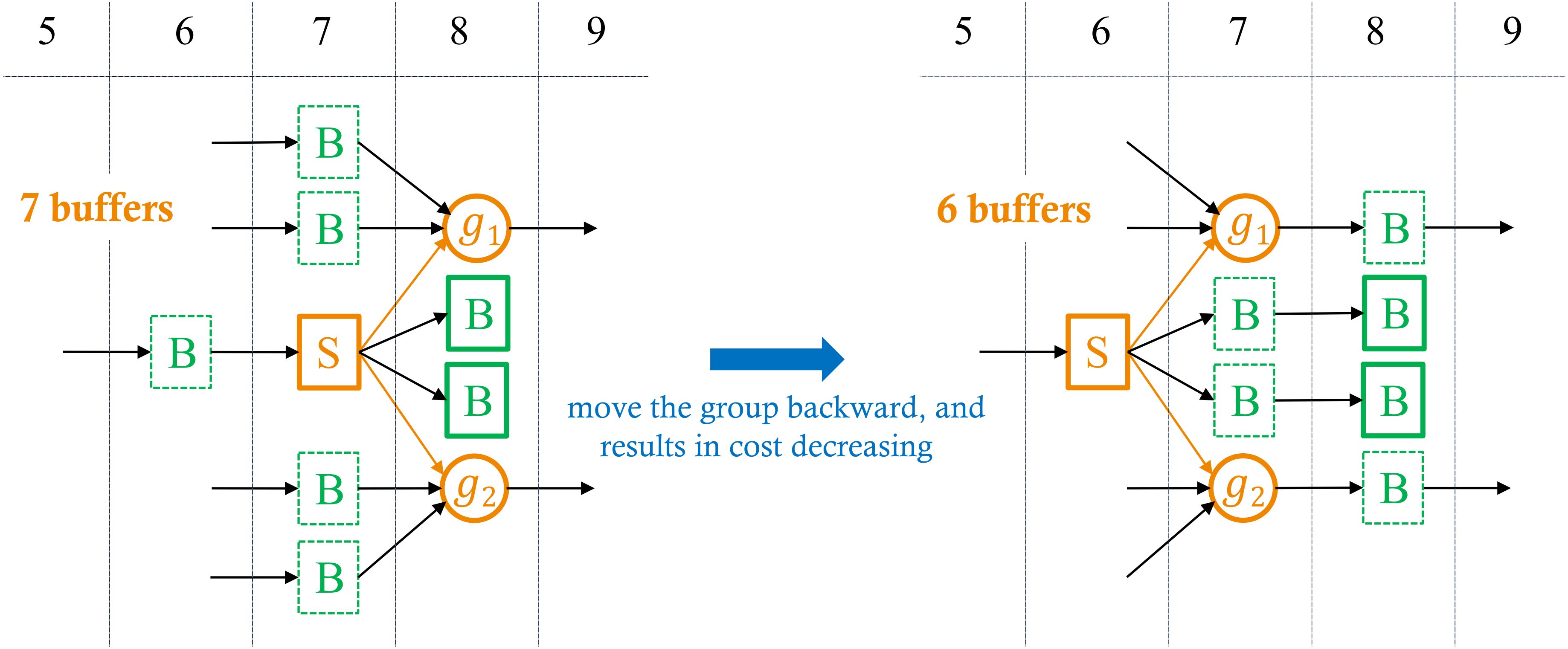
Backward Group Movement

➢ Suppose we try to move g_2 backward next, with the previously created group retained in the circuit



Backward Group Movement

- Try to move the group $\{g_1, g_2, S\}$ backward, and the result will be accepted

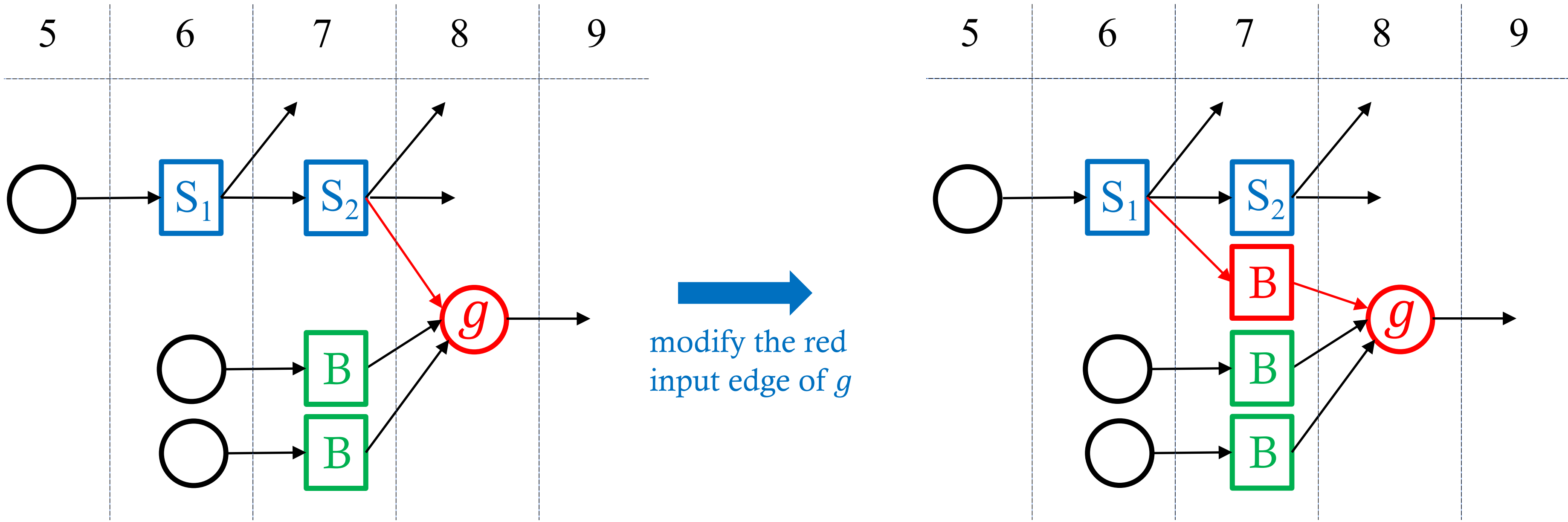


Optimization Methods

- Backward Movement Optimization
  - Backward group movement
  - Input modifying backward movement
- Forward Movement Optimization
  - Forward group movement
  - Buffer integrating forward movement
- Level Perturbation
  - Flexibility-driven branching tree
  - Forced movement
  - Fanout-pair level adjustment

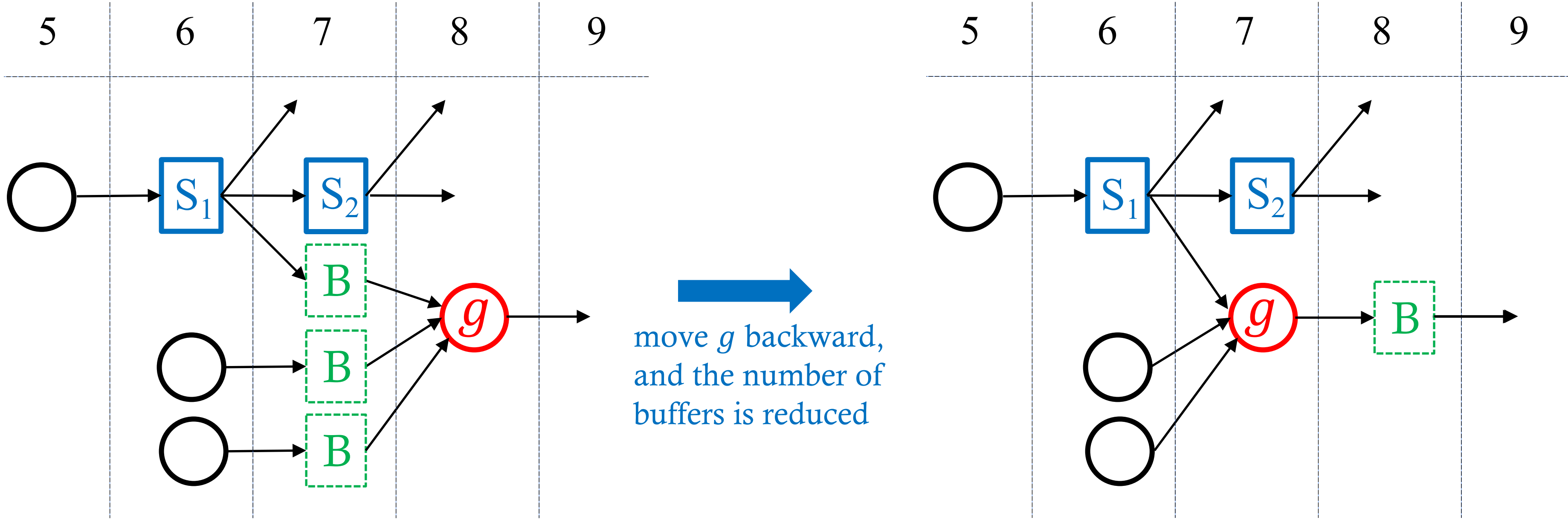
Input Modifying Backward Movement

- Modify the input edge(s) of a gate so we can move it backward easier



Input Modifying Backward Movement

- Modify the input edge(s) of a gate so we can move it backward easier



Optimization Methods

- Backward Movement Optimization
  - Backward Group Movement
  - Input Modifying Backward Movement
- Forward Movement Optimization
  - Forward Group Movement
  - Buffer Integrating Forward Movement
- Level Perturbation
  - Flexibility-driven Branching Tree
  - Forced Movement
  - Fanout-pair Level Adjustment

Forward Group Movement

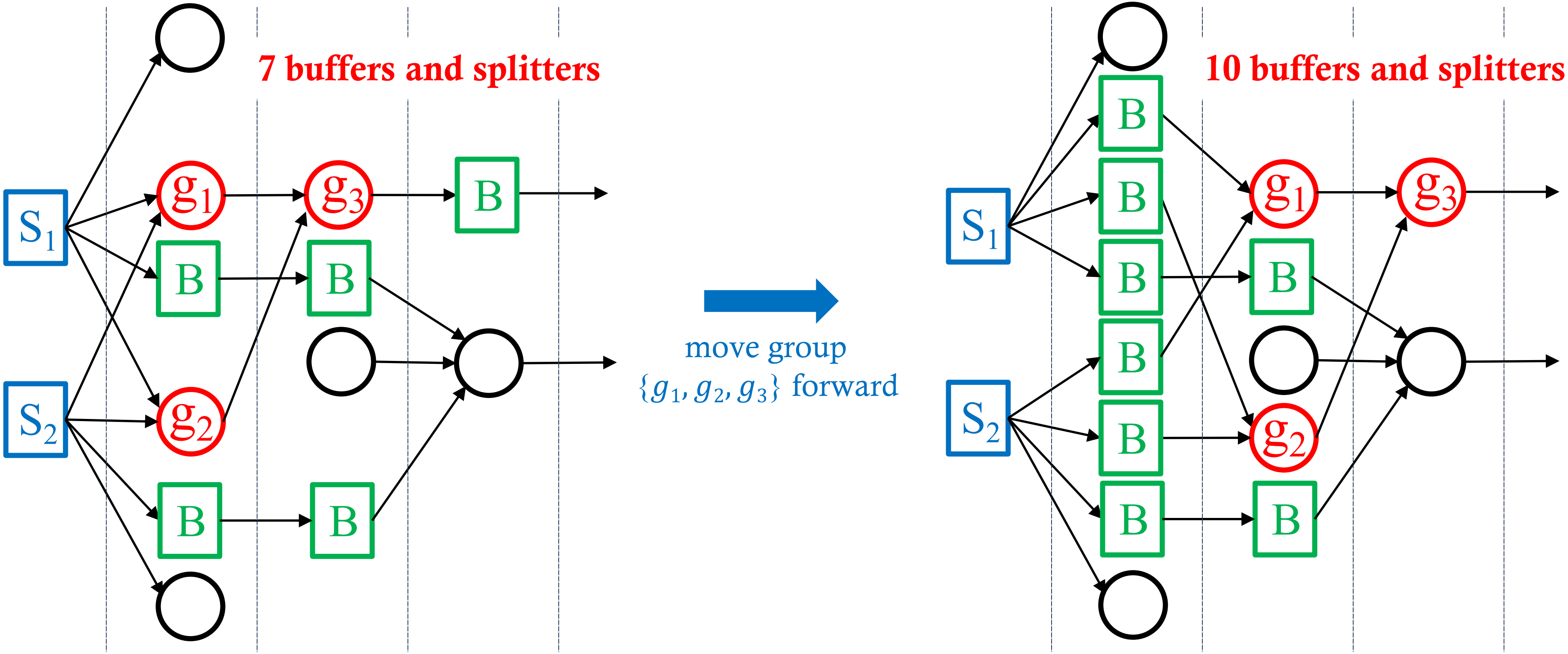
- Use similar rules in “backward group movement” to construct groups and move them forward

Optimization Methods

- Backward Movement Optimization
    - Backward Group Movement
    - Input Modifying Backward Movement
- Forward Movement Optimization
    - Forward Group Movement
    - Buffer Integrating Forward Movement
- Level Perturbation
    - Flexibility-driven Branching Tree
    - Forced Movement
    - Fanout-pair Level Adjustment

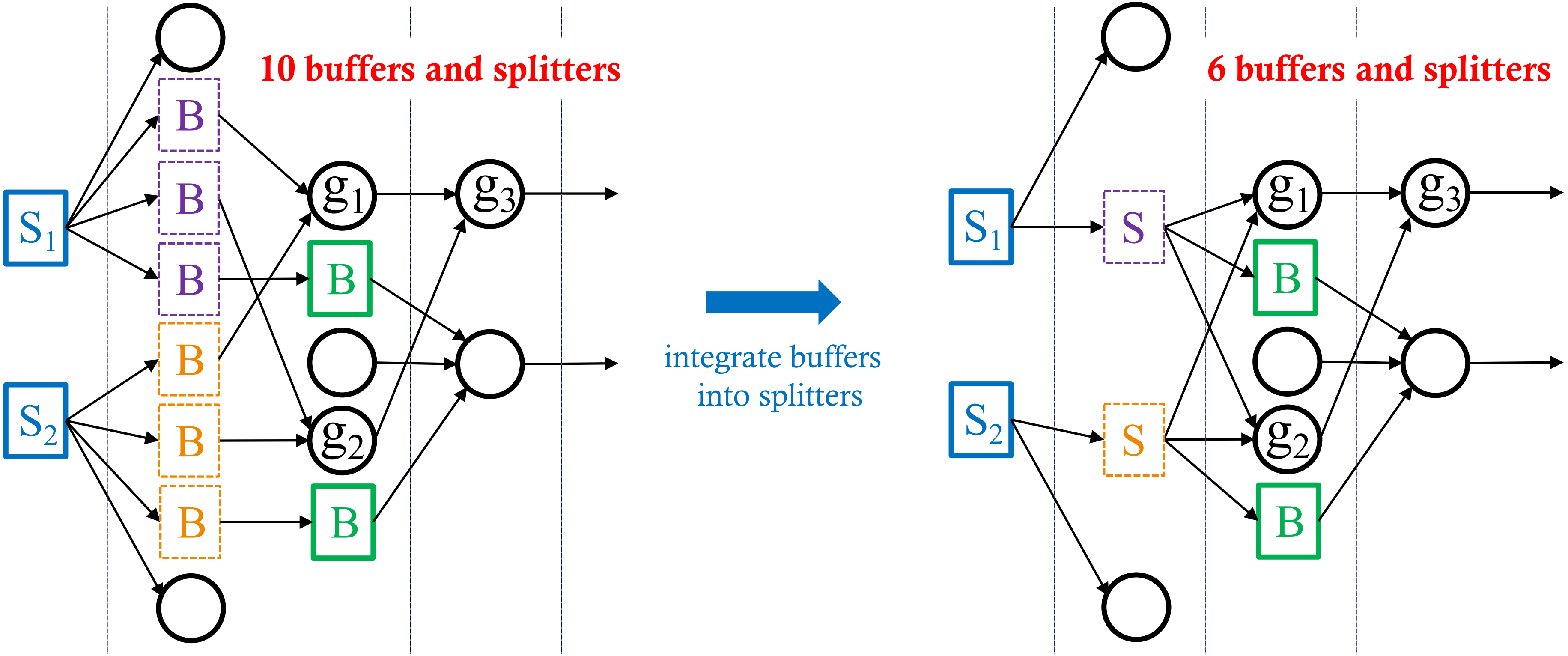
Buffer Integrating Forward Movement

- Move groups without splitters forward, and try to integrate buffers into less splitters if possible



Buffer Integrating Forward Movement

- Move groups without splitters forward, and try to integrate buffers into less splitters if possible



Optimization Methods

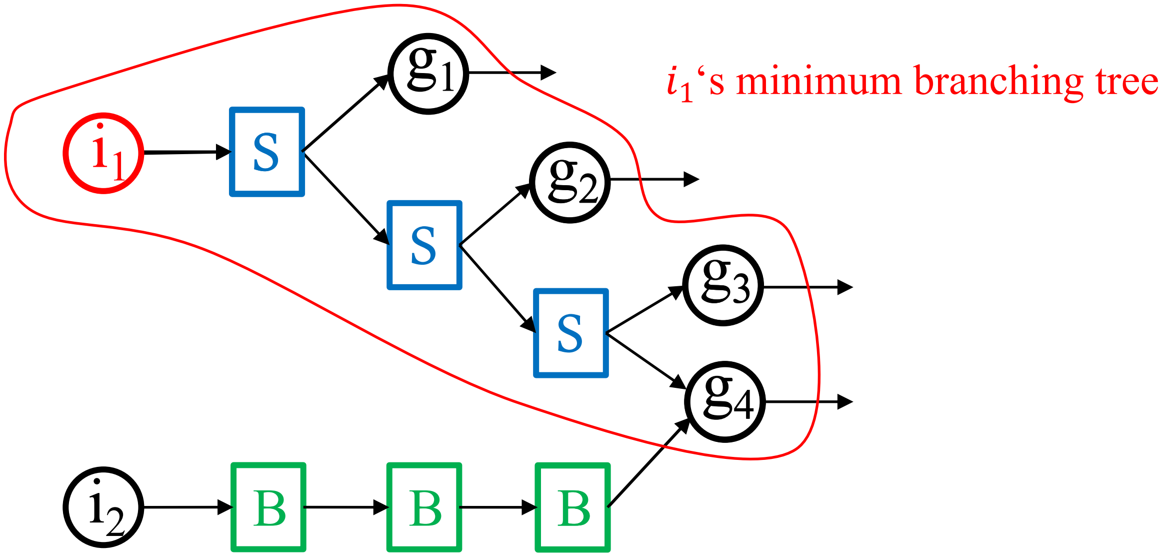
- Backward Movement Optimization
  - Backward Group Movement
  - Input Modifying Backward Movement
- Forward Movement Optimization
  - Forward Group Movement
  - Buffer Integrating Forward Movement
- Level Perturbation
  - Flexibility-driven Branching Tree
  - Forced Movement
  - Fanout-pair Level Adjustment

Flexibility-driven Branching Tree

- A branching tree of a gate or PI g is a set of nodes containing
  1) g
  2) the gates and POs that g passes its output signal to
  3) the buffers and splitters between (1) and (2)
- Minimum branching tree
  - Advantage: requires the least number of buffers and splitters
  - Disadvantage: the movement flexibility of gates is lower

Flexibility-driven Branching Tree

- The minimum branching tree of i_1
  - The movement of g_4 is blocked by a splitter

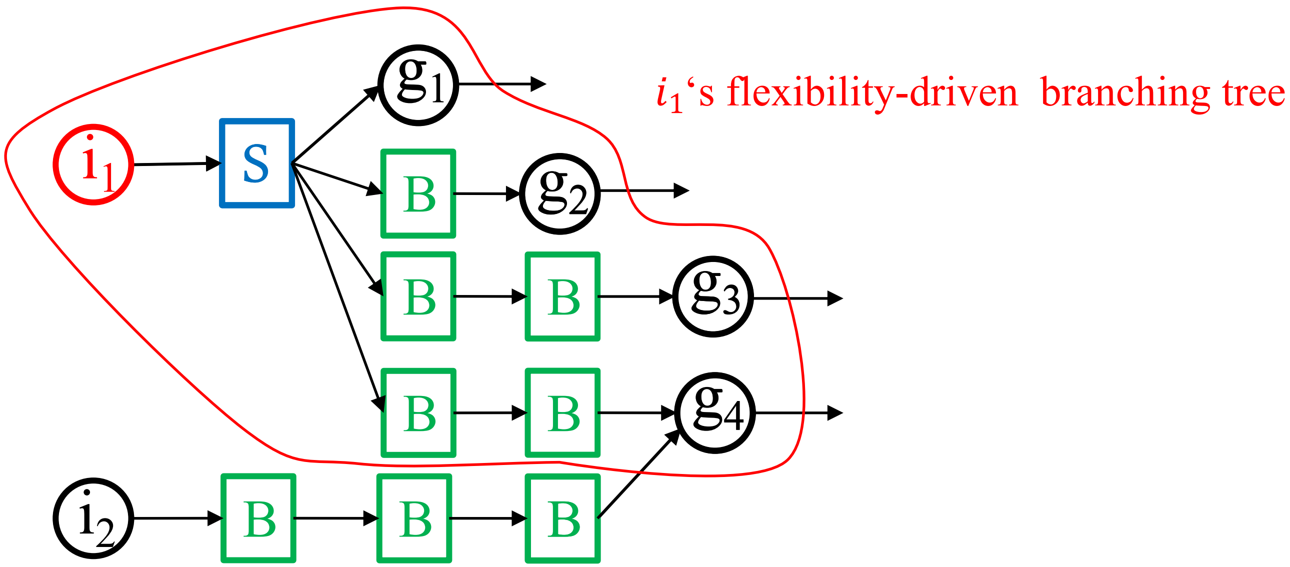


Flexibility-driven Branching Tree

- Flexibility-driven branching tree
  - Advantage: the movement flexibility of gates is higher
  - Disadvantage: use as many buffers as possible (higher cost)

Flexibility-driven Branching Tree

- The flexibility-driven branching tree of i_1
  - The movement of g_4 is NOT blocked by any gate or splitter

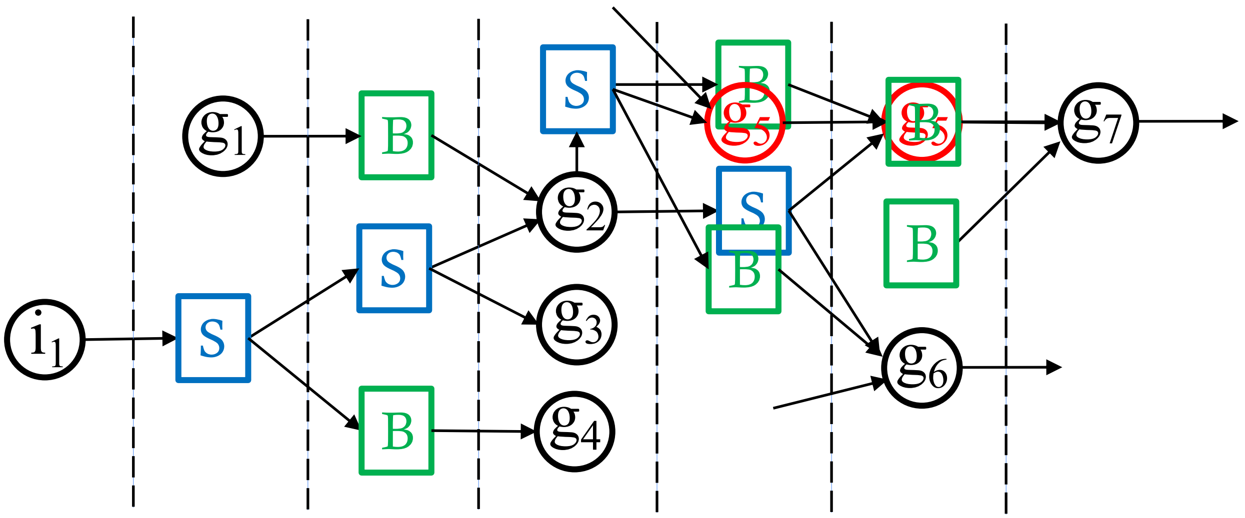


Optimization Methods

- Backward Movement Optimization
  - Backward Group Movement
  - Input Modifying Backward Movement
- Forward Movement Optimization
  - Forward Group Movement
  - Buffer Integrating Forward Movement
- Level Perturbation
  - Flexibility-driven Branching Tree
  - Forced Movement
  - Fanout-pair Level Adjustment

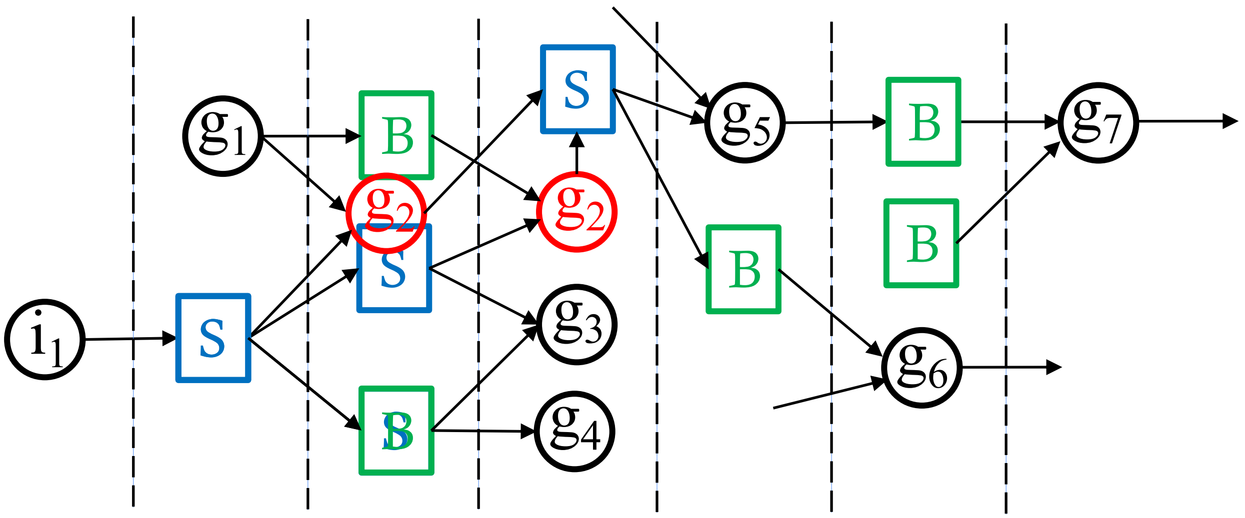
Forced Movement

- Force a gate to be moved backward one level, and resolve the created violations



Forced Movement

- Force a gate to be moved backward one level, and resolve the created violations

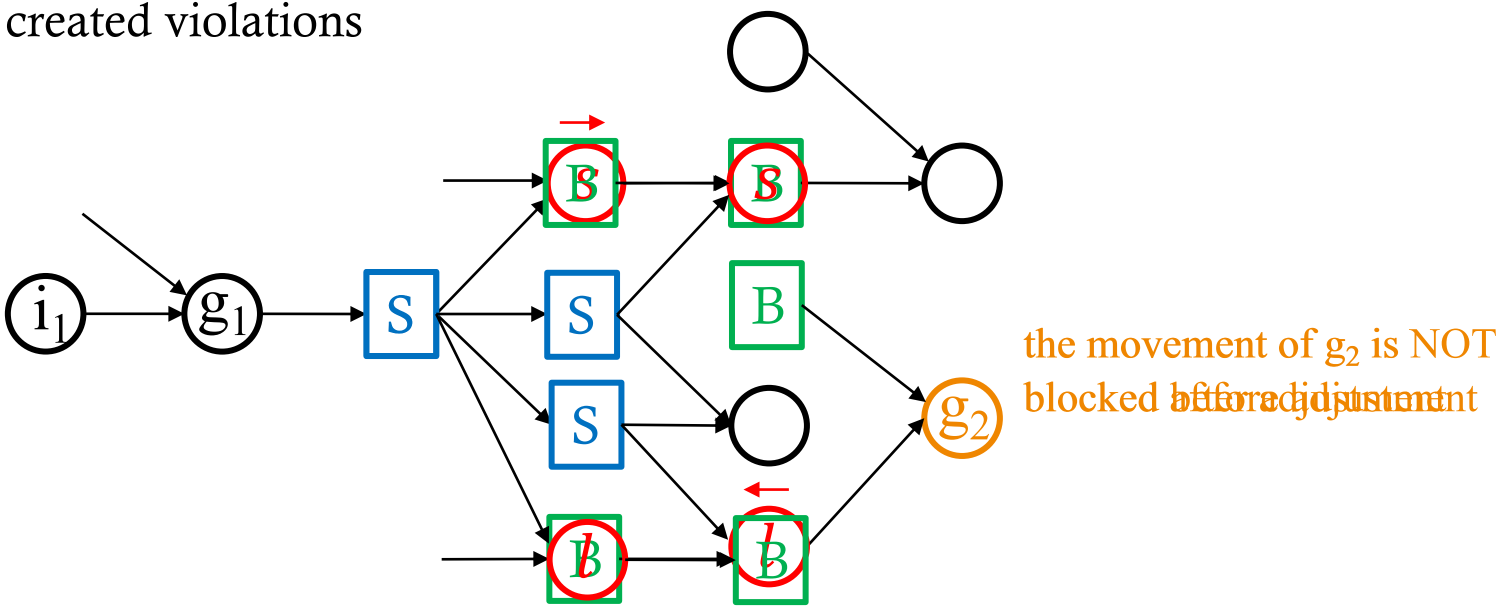


Optimization Methods

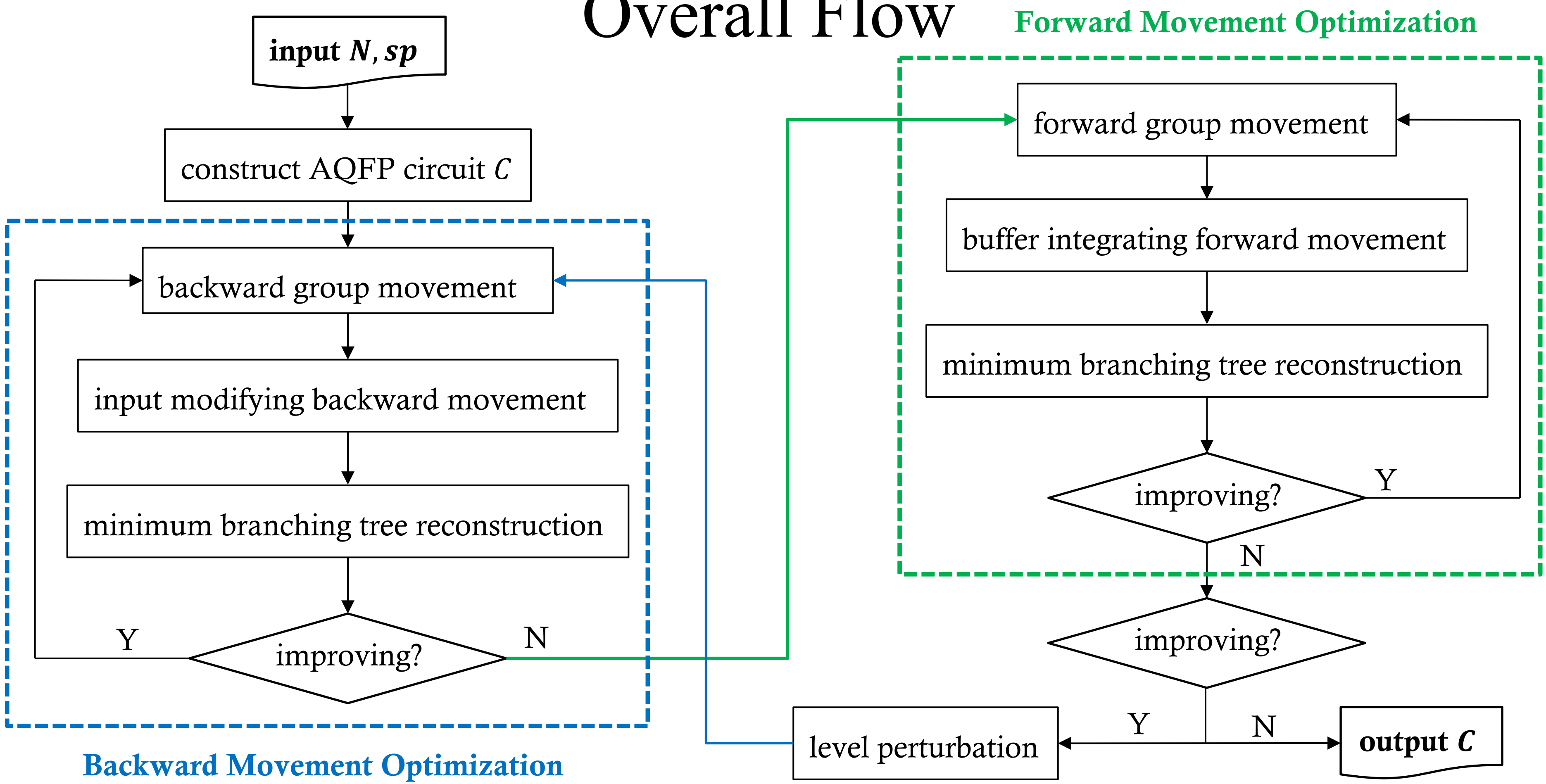
- Backward Movement Optimization
  - Backward Group Movement
  - Input Modifying Backward Movement
- Forward Movement Optimization
  - Forward Group Movement
  - Buffer Integrating Forward Movement
- Level Perturbation
  - Flexibility-driven Branching Tree
  - Forced Movement
  - Fanout-pair Level Adjustment

Fanout-pair Level Adjustment

- For gates *s* and *l* in the same branching tree (assume *s* is at a smaller level than that of *l*), move *s* forward one level and move *l* backward one level, then resolve the created violations



Overall Flow



Experimental Results

- Compare to state-of-the-art methods
  - A Global Optimization Algorithm for Buffer and Splitter Insertion in Adiabatic Quantum-Flux-Parametron Circuits **[3]**
    1) Use integer linear programming-based method to decide logic gates levels
    2) Use dynamic programming-based method to insert buffers and splitters
  - Depth-Optimal Buffer and Splitter Insertion and Optimization in AQFP Circuits **[2]**
    1) Obtained depth-optimal circuits
    2) Use retiming-based method to move gates and splitters for cost reduction

Experimental Results

- Optimal solution

Benchmark	Original Circuit			Opt from [1]		[3]		[2]			Our method		
	Gates	max|LO|	D	#B/S	D’	#B/S	D’	#B/S	D’	Time(s)	#B/S	D’	Time(s)
adder1	7	2	4	**16***	8	-	-	**16***	8	<0.01	**16***	8	<0.01
adder8	77	3	17	**371***	33	-	-	**371***	33	<0.01	**371***	33	<0.01
counter16	29	4	9	**65***	17	66	17	**65***	17	<0.01	**65***	17	<0.01
counter32	82	4	13	**154***	23	156	23	**154***	23	<0.01	**154***	23	<0.01
counter64	195	4	17	**347***	30	351	30	**347***	30	<0.01	**347***	30	<0.01
counter128	428	4	22	**747***	38	755	38	**747***	38	0.01	**747***	38	0.01
sorter32	480	2	15	**480***	30	-	-	**480***	30	<0.01	**480***	30	<0.01
sorter48	880	3	20	**880***	35	-	-	**880***	35	0.01	**880***	35	0.01
mult8	439	9	35	-	-	1681	70	1709	70	0.05	**1680**	70	0.04
c17	6	2	3	**12***	5	-	-	**12***	5	<0.01	**12***	5	<0.01
c432	121	10	26	-	-	**829**	37	839	37	0.01	**829**	37	<0.01
c499	387	8	18	-	-	**1173**	29	**1173**	29	0.03	1177	29	0.01
c880	306	9	27	-	-	1536	40	**1511**	40	0.07	1517	40	0.03
c1355	389	9	18	-	-	1186	29	1184	29	0.03	**1182**	29	0.02
c1908	289	14	21	-	-	1253	34	**1236**	34	0.04	**1236**	34	0.01
c2670	368	32	21	-	-	**1869**	28	1940	28	0.07	1914	28	0.03
c3540	794	38	32	-	-	**1963**	52	1966	52	0.11	1993	52	0.11
c5315	1302	41	26	-	-	**5505**	40	5635	40	0.32	5584	40	0.16
c6288	1870	17	89	-	-	8832	179	9009	179	0.2	**8632**	179	0.18
c7552	1394	170	33	-	-	6768	58	7832	56	0.66	**6614**	56	0.42
alu32	1513	128	100	-	-	13976	169	13842	169	0.59	**13804**	169	0.65

Experimental Results

- Compare to [3] (11 better, 4 worse)

Benchmark	Original Circuit			Opt from [1]		[3]		[2]			Our method		
	Gates	$\max\lvert LO\rvert$	D	#B/S	D'	#B/S	D'	#B/S	D'	Time(s)	#B/S	D'	Time(s)
adder1	7	2	4	$\mathbf{16}^*$	8	-	-	$\mathbf{16}^*$	8	<0.01	$\mathbf{16}^*$	8	<0.01
adder8	77	3	17	$\mathbf{371}^*$	33	-	-	$\mathbf{371}^*$	33	<0.01	$\mathbf{371}^*$	33	<0.01
counter16	29	4	9	$\mathbf{65}^*$	17	66	17	$\mathbf{65}^*$	17	<0.01	$\mathbf{65}^*$	17	<0.01
counter32	82	4	13	$\mathbf{154}^*$	23	156	23	$\mathbf{154}^*$	23	<0.01	$\mathbf{154}^*$	23	<0.01
counter64	195	4	17	$\mathbf{347}^*$	30	351	30	$\mathbf{347}^*$	30	<0.01	$\mathbf{347}^*$	30	<0.01
counter128	428	4	22	$\mathbf{747}^*$	38	755	38	$\mathbf{747}^*$	38	0.01	$\mathbf{747}^*$	38	0.01
sorter32	480	2	15	$\mathbf{480}^*$	30	-	-	$\mathbf{480}^*$	30	<0.01	$\mathbf{480}^*$	30	<0.01
sorter48	880	3	20	$\mathbf{880}^*$	35	-	-	$\mathbf{880}^*$	35	0.01	$\mathbf{880}^*$	35	0.01
mult8	439	9	35	-	-	1681	70	1709	70	0.05	**1680**	70	0.04
c17	6	2	3	$\mathbf{12}^*$	5	-	-	$\mathbf{12}^*$	5	<0.01	$\mathbf{12}^*$	5	<0.01
c432	121	10	26	-	-	**829**	37	839	37	0.01	**829**	37	<0.01
c499	387	8	18	-	-	**1173**	29	**1173**	29	0.03	1177	29	0.01
c880	306	9	27	-	-	1536	40	**1511**	40	0.07	1517	40	0.03
c1355	389	9	18	-	-	1186	29	1184	29	0.03	**1182**	29	0.02
c1908	289	14	21	-	-	1253	34	**1236**	34	0.04	**1236**	34	0.01
c2670	368	32	21	-	-	**1869**	28	1940	28	0.07	1914	28	0.03
c3540	794	38	32	-	-	**1963**	52	1966	52	0.11	1993	52	0.11
c5315	1302	41	26	-	-	**5505**	40	5635	40	0.32	5584	40	0.16
c6288	1870	17	89	-	-	8832	179	9009	179	0.2	**8632**	179	0.18
c7552	1394	170	33	-	-	6768	58	7832	56	0.66	**6614**	56	0.42
alu32	1513	128	100	-	-	13976	169	13842	169	0.59	**13804**	169	0.65

Experimental Results

- Compare to [2] (8 better, 3 worse)

Benchmark	Original Circuit			Opt from [1]		[3]		[2]			Our method		
	Gates	max|LO|	D	#B/S	D'	#B/S	D'	#B/S	D'	Time(s)	#B/S	D'	Time(s)
adder1	7	2	4	$\mathbf{16}^*$	8	-	-	$\mathbf{16}^*$	8	<0.01	$\mathbf{16}^*$	8	<0.01
adder8	77	3	17	$\mathbf{371}^*$	33	-	-	$\mathbf{371}^*$	33	<0.01	$\mathbf{371}^*$	33	<0.01
counter16	29	4	9	$\mathbf{65}^*$	17	66	17	$\mathbf{65}^*$	17	<0.01	$\mathbf{65}^*$	17	<0.01
counter32	82	4	13	$\mathbf{154}^*$	23	156	23	$\mathbf{154}^*$	23	<0.01	$\mathbf{154}^*$	23	<0.01
counter64	195	4	17	$\mathbf{347}^*$	30	351	30	$\mathbf{347}^*$	30	<0.01	$\mathbf{347}^*$	30	<0.01
counter128	428	4	22	$\mathbf{747}^*$	38	755	38	$\mathbf{747}^*$	38	0.01	$\mathbf{747}^*$	38	0.01
sorter32	480	2	15	$\mathbf{480}^*$	30	-	-	$\mathbf{480}^*$	30	<0.01	$\mathbf{480}^*$	30	<0.01
sorter48	880	3	20	$\mathbf{880}^*$	35	-	-	$\mathbf{880}^*$	35	0.01	$\mathbf{880}^*$	35	0.01
mult8	439	9	35	-	-	1681	70	1709	70	0.05	**1680**	70	0.04
c17	6	2	3	$\mathbf{12}^*$	5	-	-	$\mathbf{12}^*$	5	<0.01	$\mathbf{12}^*$	5	<0.01
c432	121	10	26	-	-	**829**	37	839	37	0.01	**829**	37	<0.01
c499	387	8	18	-	-	**1173**	29	**1173**	29	0.03	1177	29	0.01
c880	306	9	27	-	-	1536	40	**1511**	40	0.07	1517	40	0.03
c1355	389	9	18	-	-	1186	29	1184	29	0.03	**1182**	29	0.02
c1908	289	14	21	-	-	1253	34	**1236**	34	0.04	**1236**	34	0.01
c2670	368	32	21	-	-	**1869**	28	1940	28	0.07	1914	28	0.03
c3540	794	38	32	-	-	**1963**	52	1966	52	0.11	1993	52	0.11
c5315	1302	41	26	-	-	**5505**	40	5635	40	0.32	5584	40	0.16
c6288	1870	17	89	-	-	8832	179	9009	179	0.2	**8632**	179	0.18
c7552	1394	170	33	-	-	6768	58	7832	56	0.66	**6614**	56	0.42
alu32	1513	128	100	-	-	13976	169	13842	169	0.59	**13804**	169	0.65

Conclusion

- For optimizing the buffer and splitter insertion problem in AQFP circuits, we proposed
  1) Movement methods
  2) Level perturbation methods
- Compared to state of the art [2] and [3], our method obtained better results with less runtime in most of the benchmarks

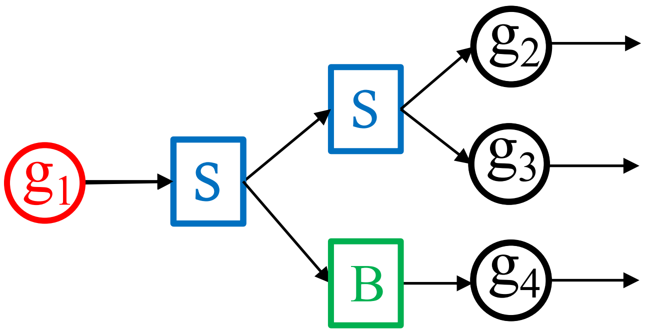
Experimental Results in Large Benchmarks

- Benchmarks with number of logic gates range from 4000 ~ 130000

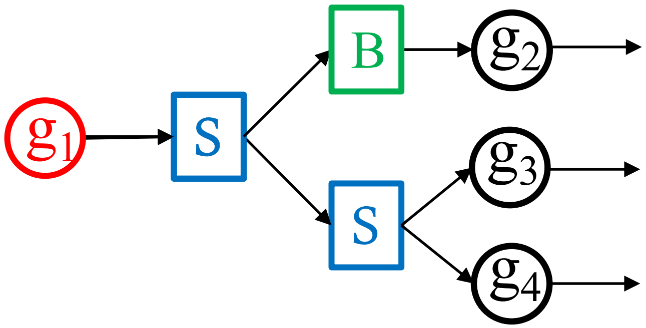
	Initial			[2]			Our Method			Improvement	
Benchmark	#gates	depth		#B/S	depth	time (s)	#B/S	depth	time (s)	#B/S	time
sin	4303	110		14783	188	2.43	14685	188	1.02	0.66%	58.02%
arbiter	7000	59		25725	63	0.75	25731	63	2	-0.02%	-166.67%
voter	7860	47		15736	86	0.5	15810	86	0.67	-0.47%	-34.00%
square	12180	126		63087	251	10.6	65602	251	6.8	-3.99%	35.85%
multiplier	19710	133		61714	264	43.76	65420	264	10.99	-6.01%	74.89%
log2	24456	200		84440	379	48.76	84215	379	32.83	0.27%	32.67%
mem_ctrl	42758	73		213463	114	95.78	185443	114	38.68	13.13%	59.62%
sqrt	23238	3366		1323854	6628	890.73	1317608	6628	34.03	0.47%	96.18%
div	57300	2217		1617032	4371	2190.5	1373661	4371	102.68	15.05%	95.31%
hyp	136109	8762		5596808	17246	2610	5479479	17246	402.91	2.10%	84.56%
			AVG							2.12%	33.64%

Different Minimum Branching Trees

- Hard to predict which one is more helpful for movements



minimum branching tree A



minimum branching tree B

Buffer Integration in Backward Movement

- Using more splitters would decrease the movability of gates