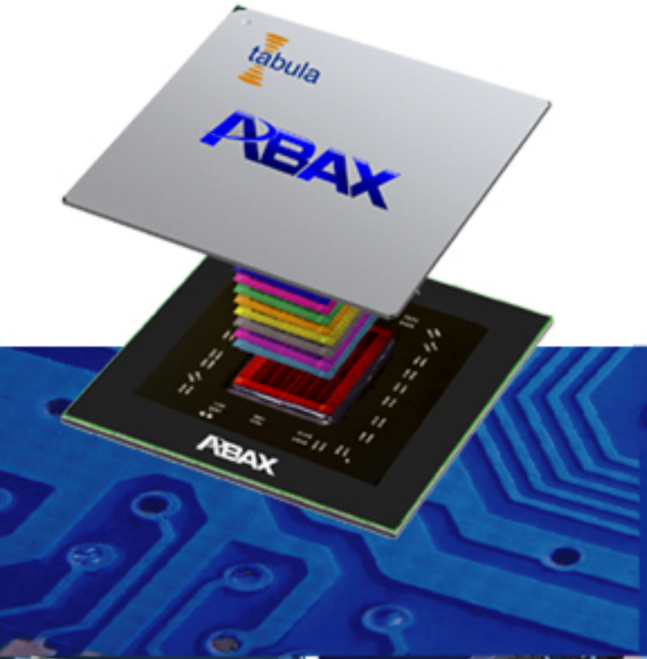
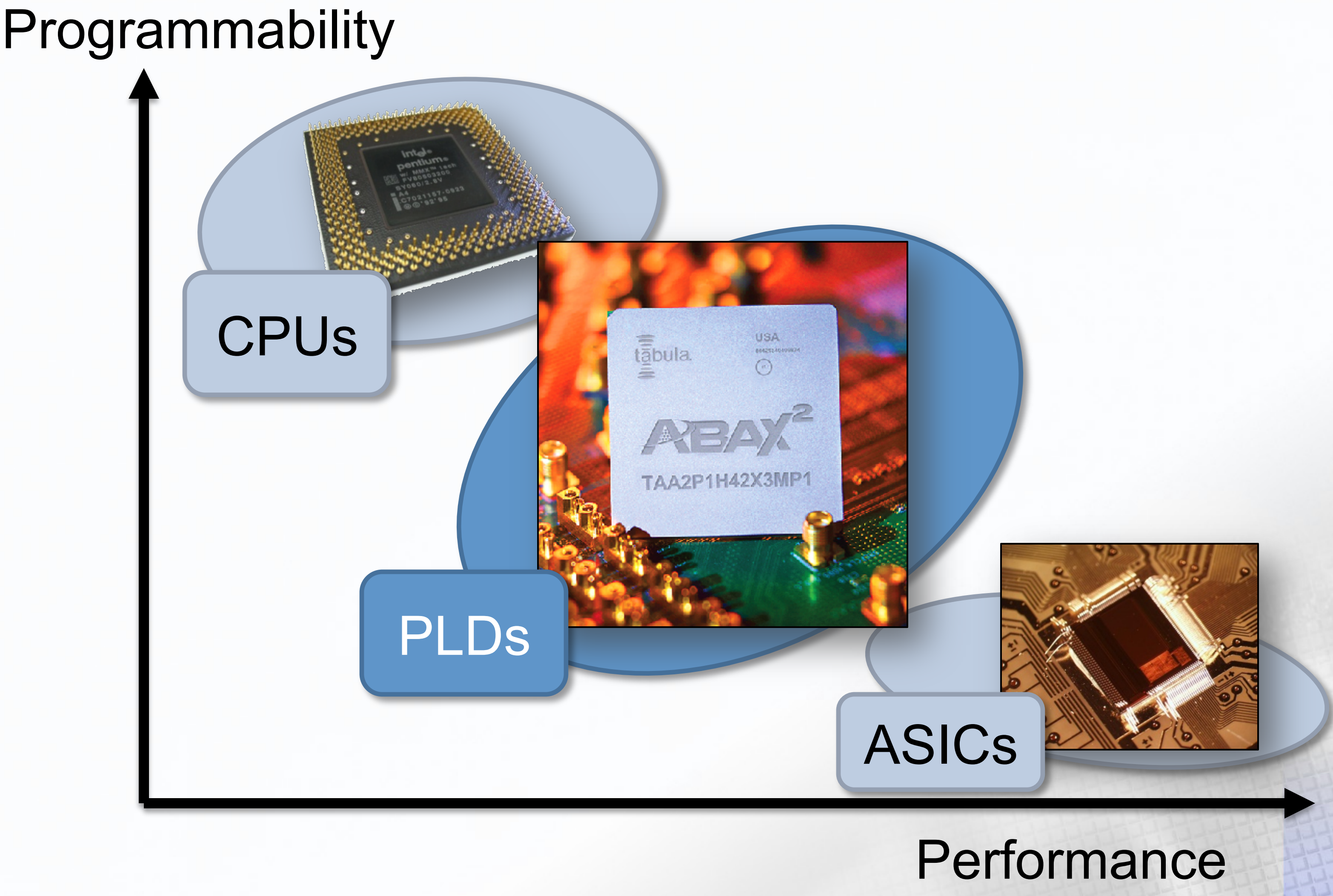
A Constraint Satisfaction Approach for Programmable Logic Detailed Placement



Andrew Mihal and Steve Teig
SAT 2013

THE BEST PATH FROM IDEAS TO PRODUCTION SILICON ®

Programmable Logic Devices



Programmability

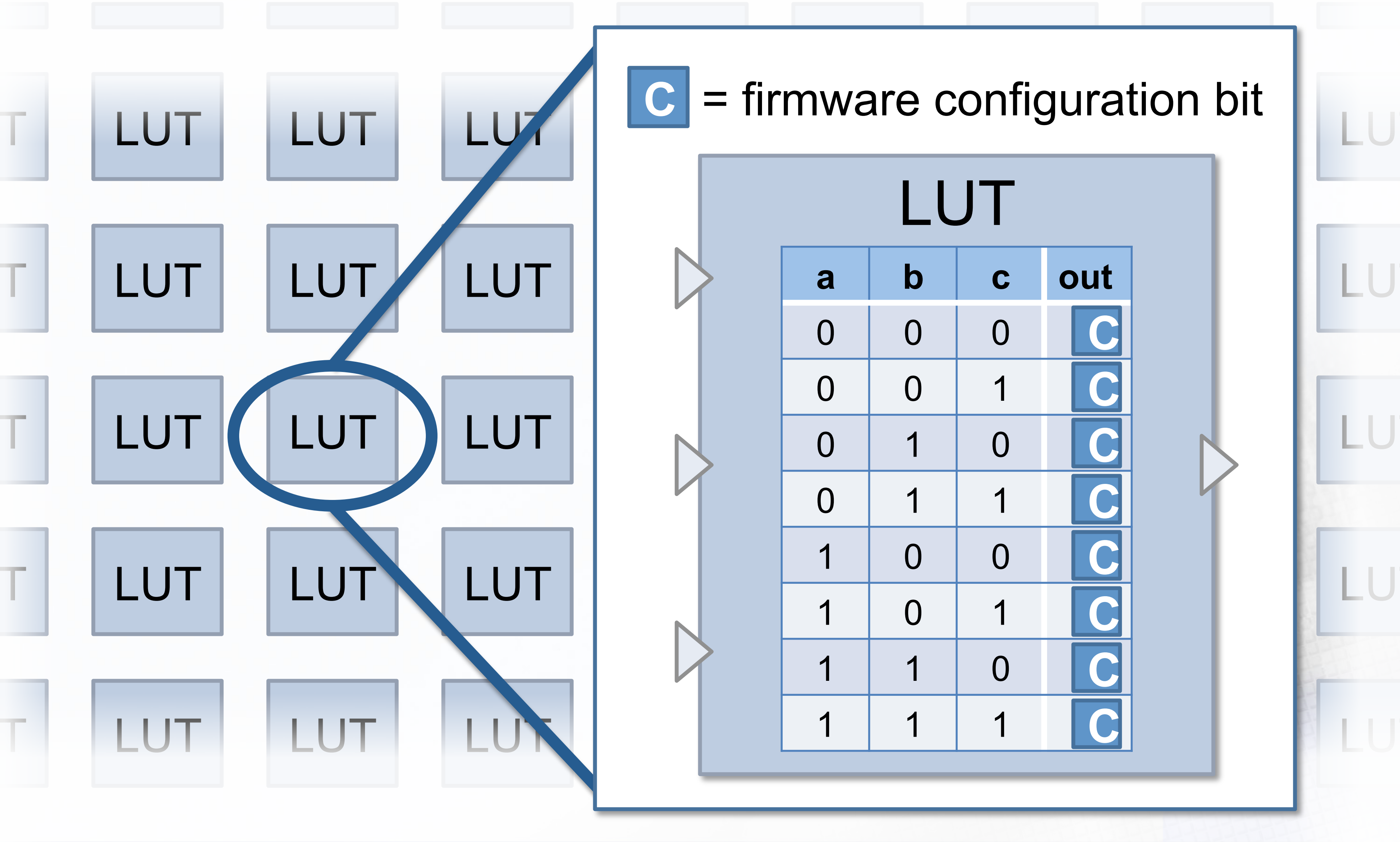
CPUs

PLDs

ASICs

Performance

Lookup Table Fabric

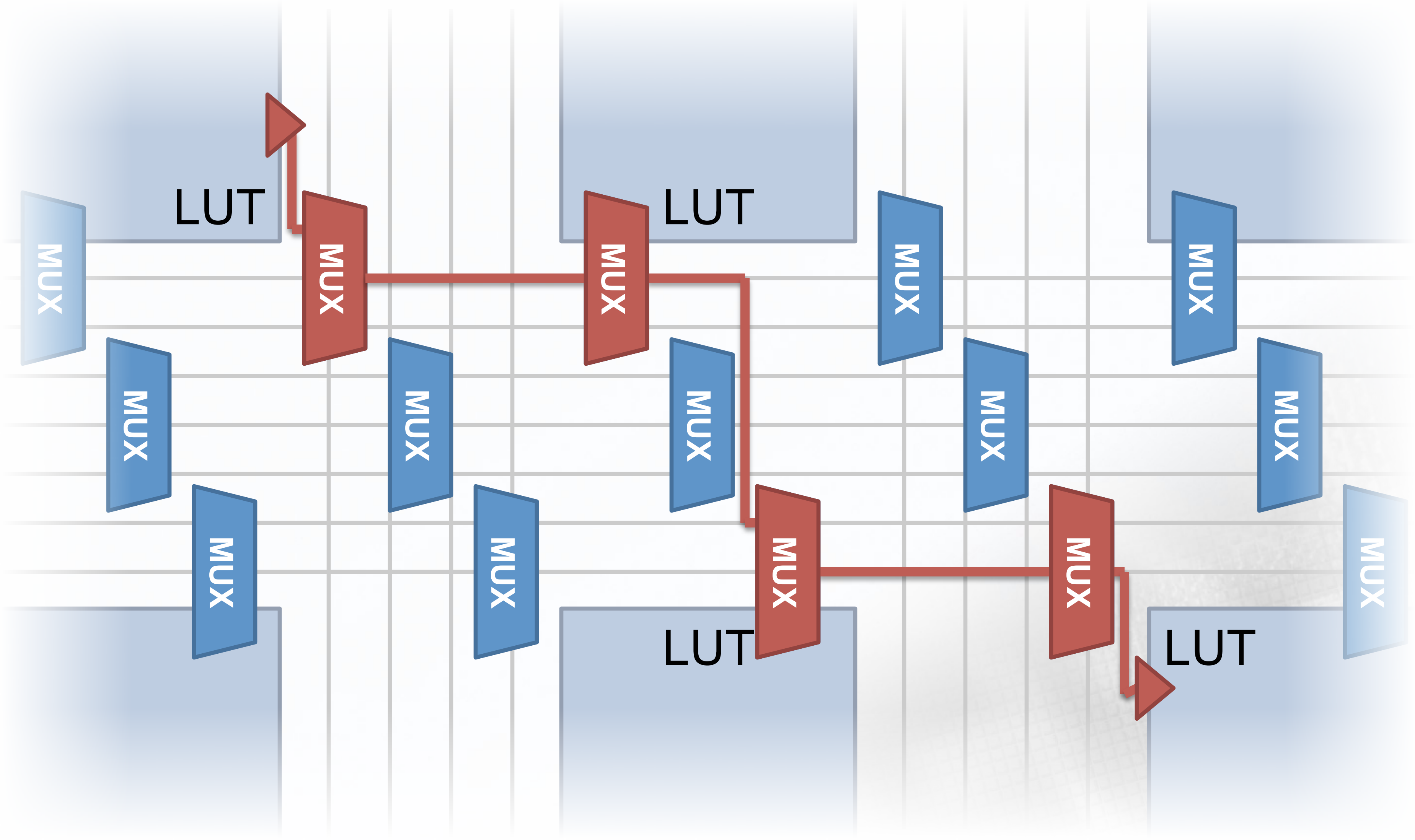


C = firmware configuration bit

LUT

a	b	c	out
0	0	0	C
0	0	1	C
0	1	0	C
0	1	1	C
1	0	0	C
1	0	1	C
1	1	0	C
1	1	1	C

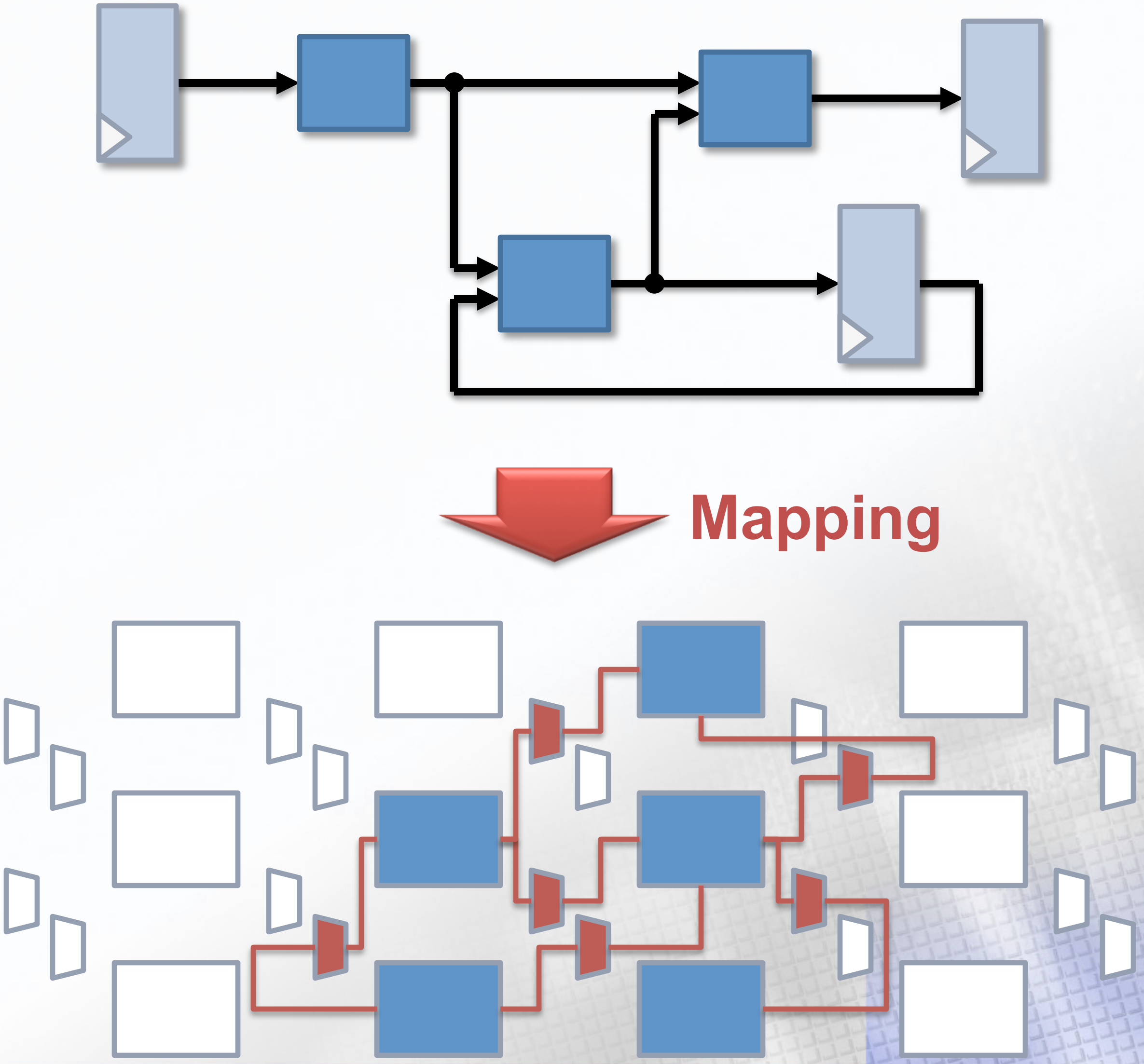
Interconnect

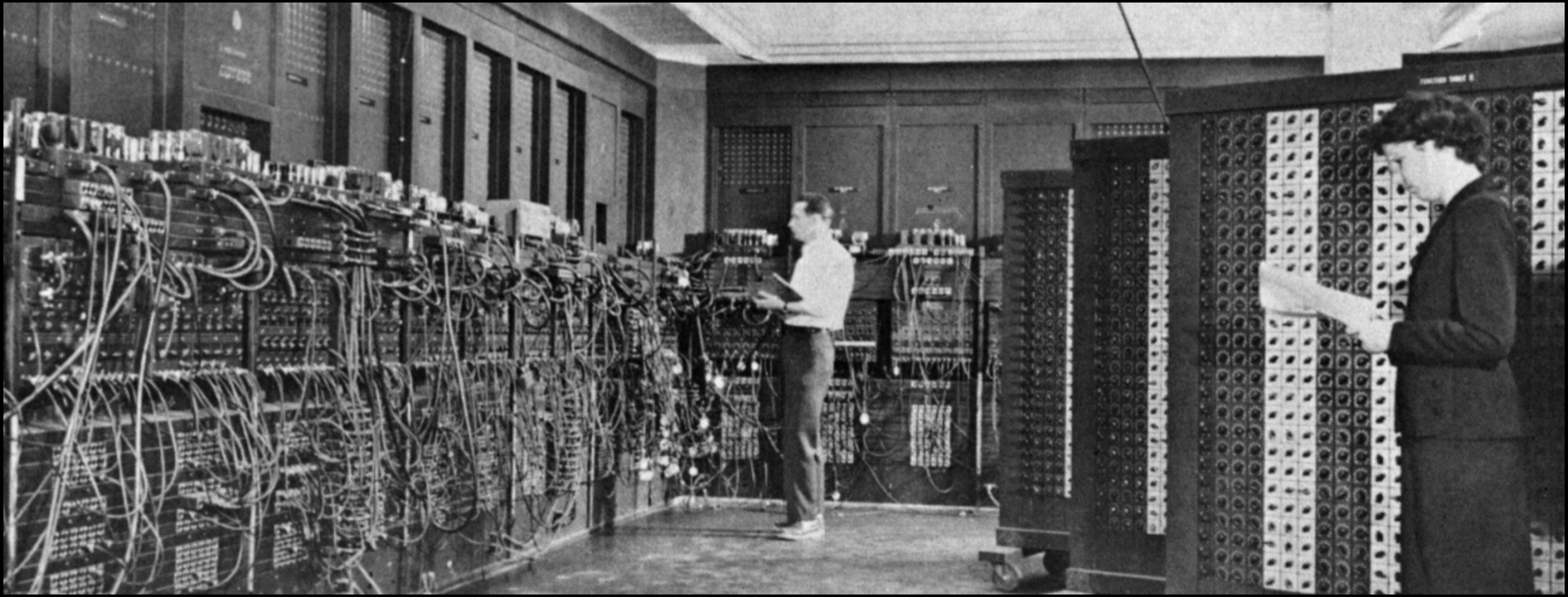


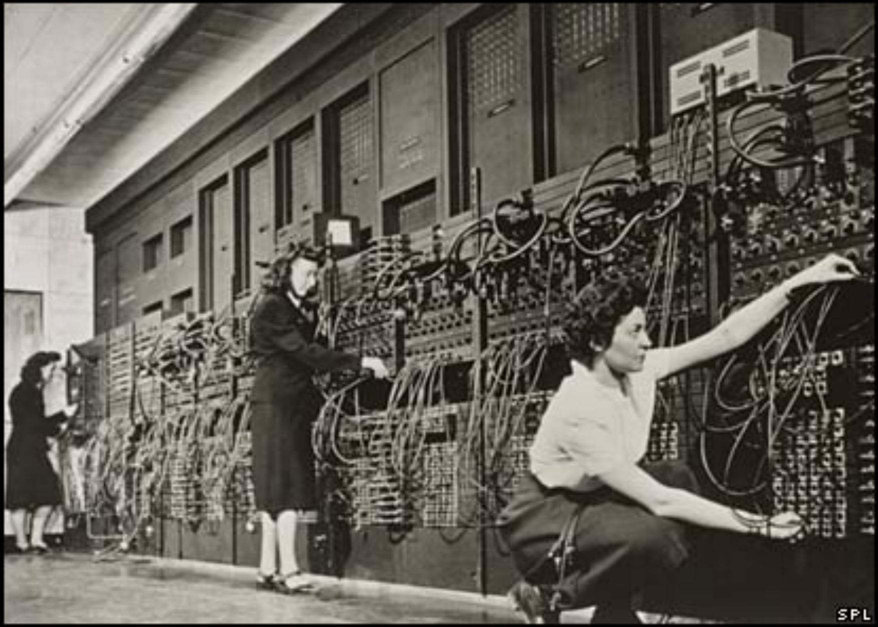
Programming

Input: Circuit Netlist

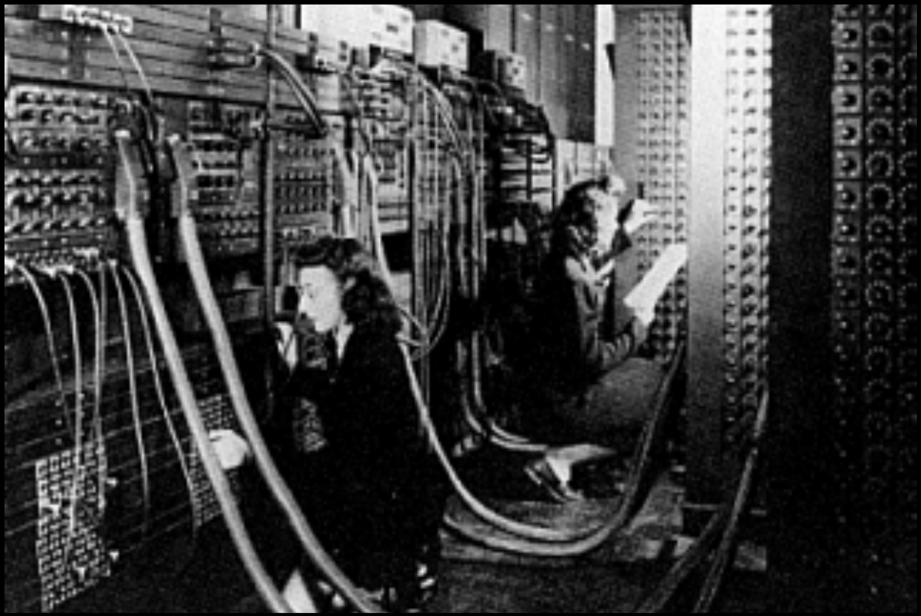
Output: Config Bits



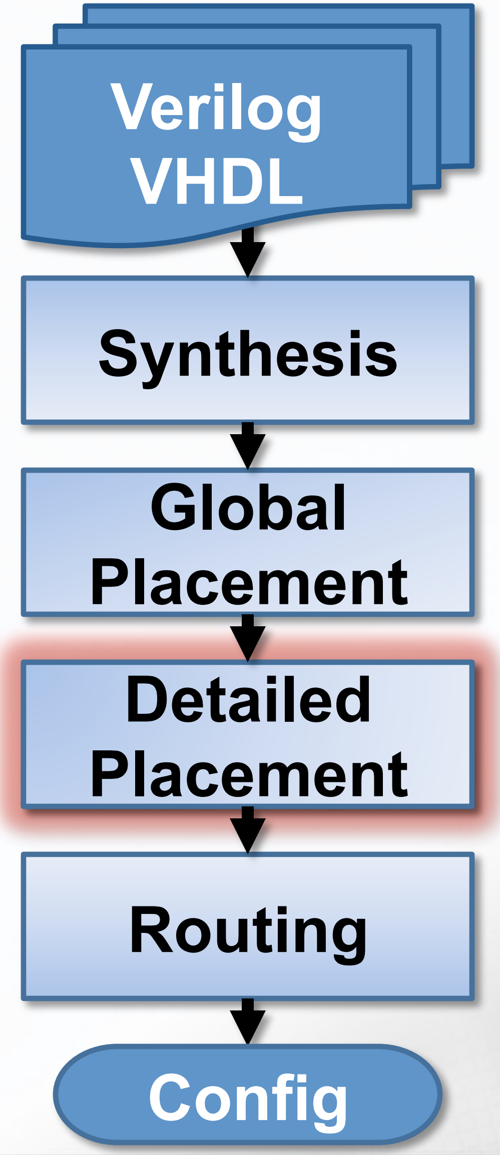




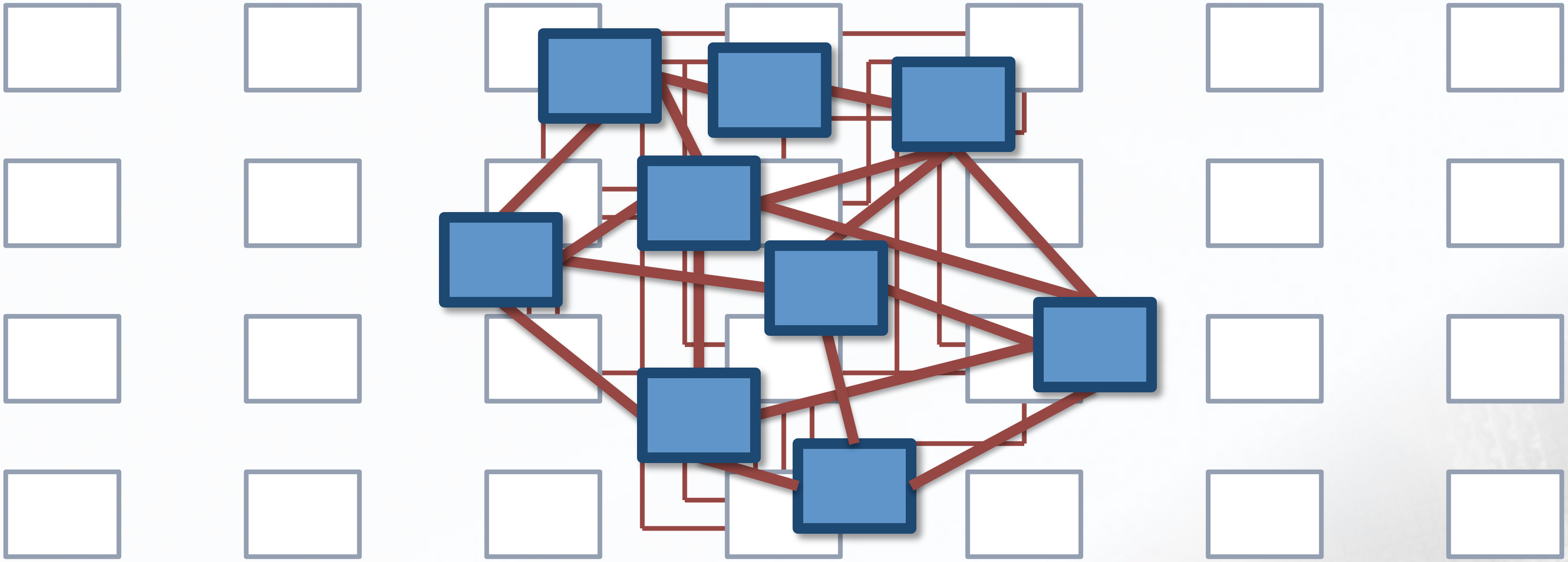
SPL



Programming Toolchain

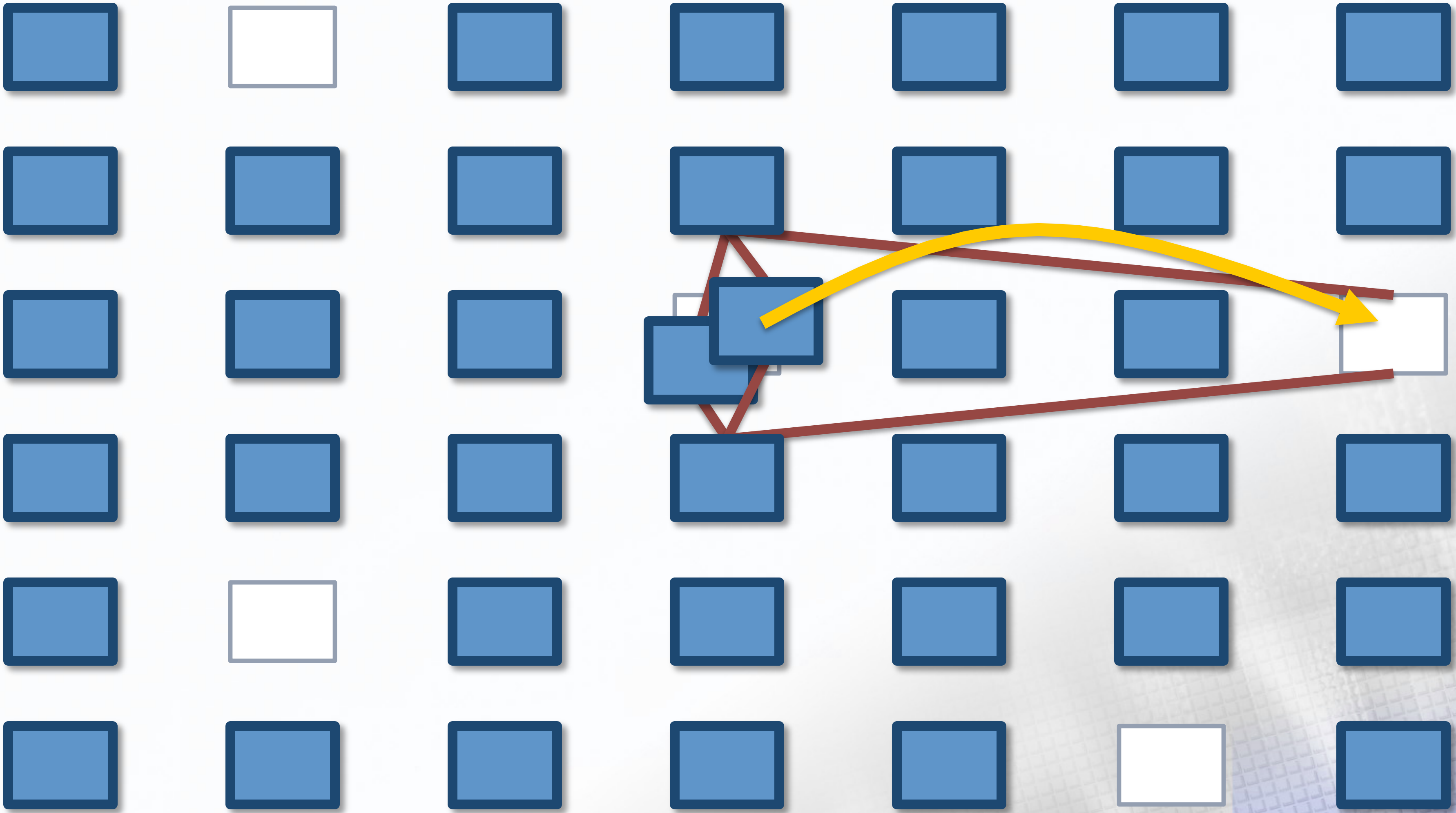


Detailed Placement

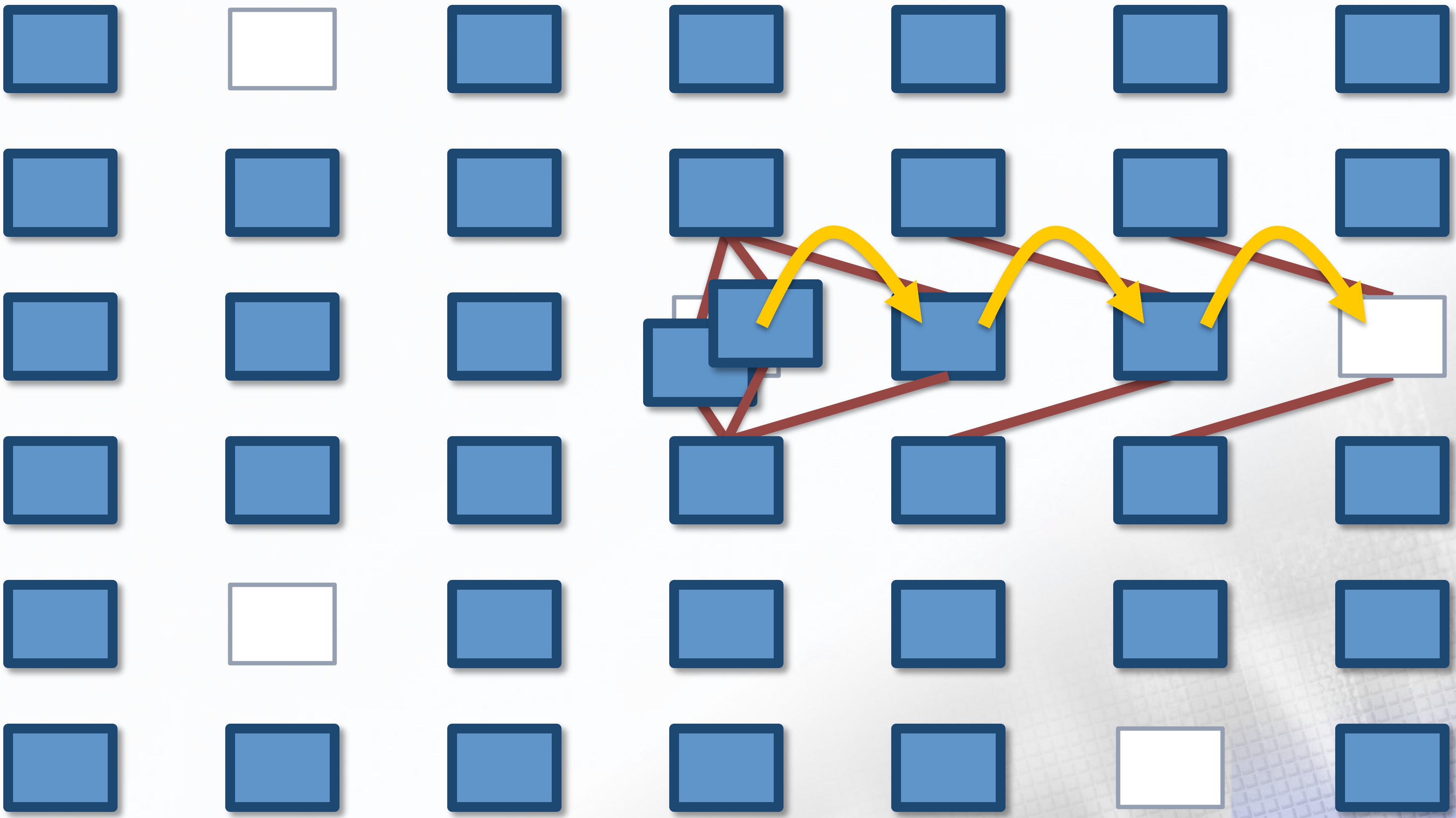


- No Overlaps
- Congestion, Routability
- Timing

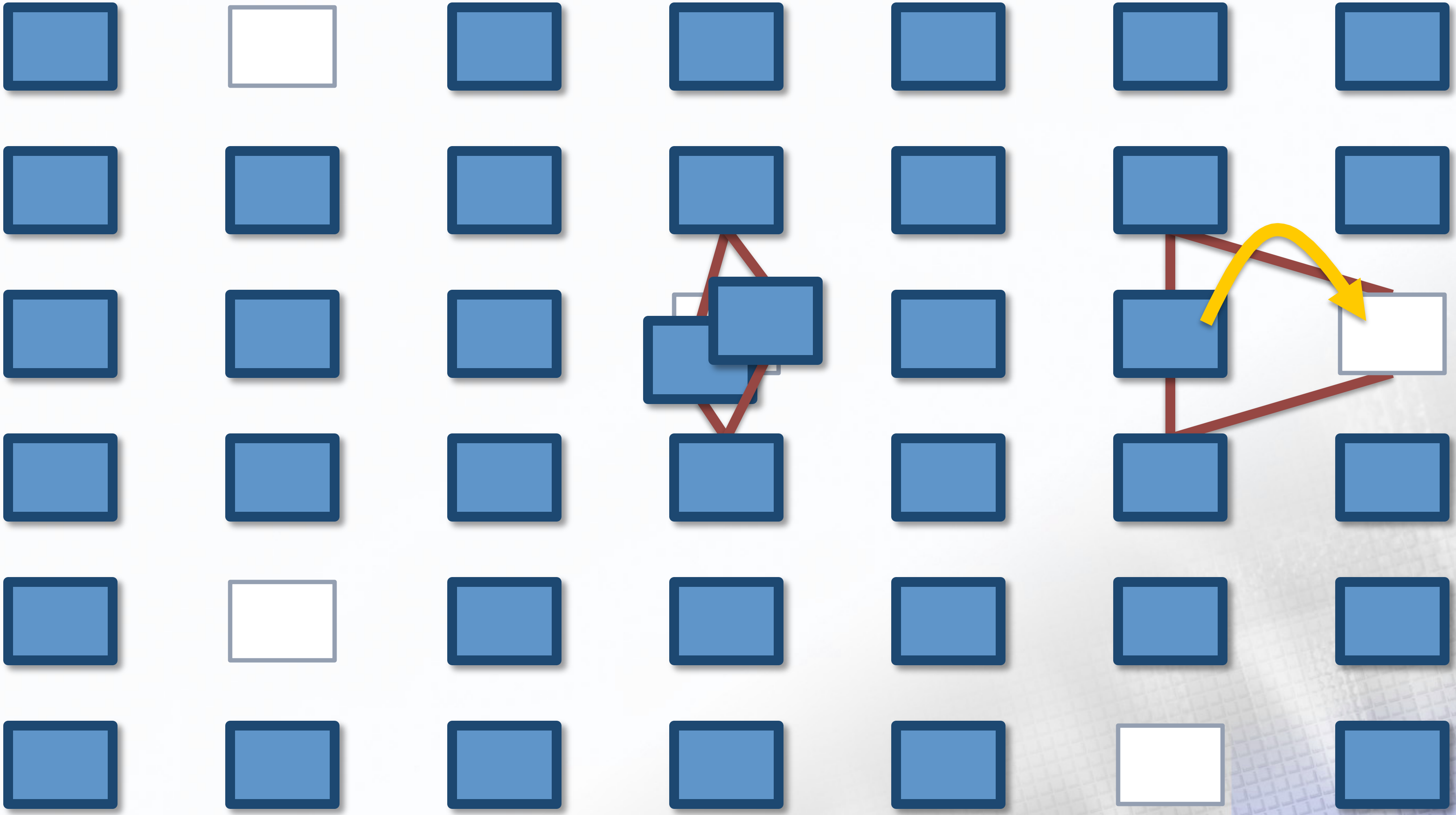
Fixing an Overlap



Everybody Shuffle



Steps Look Bad in Isolation



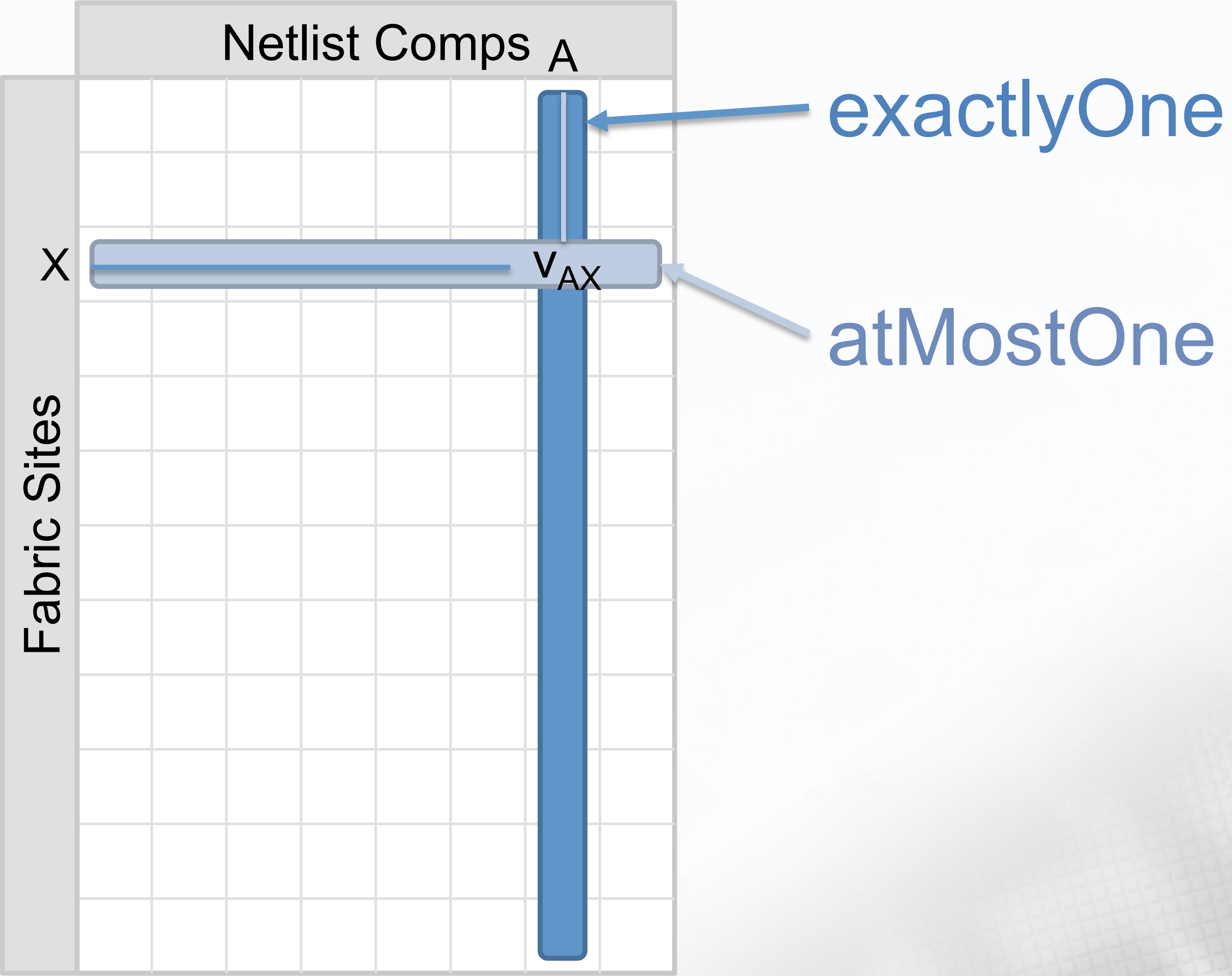
Why not SAT?

- Problem space is too big
  - Components × Sites
  - LUT Permutations
  - Multiple Routing Variations
  - Retiming
  - 3D Fabric with Transparent Latches
- Optimization Goals
  - Timing
  - Congestion

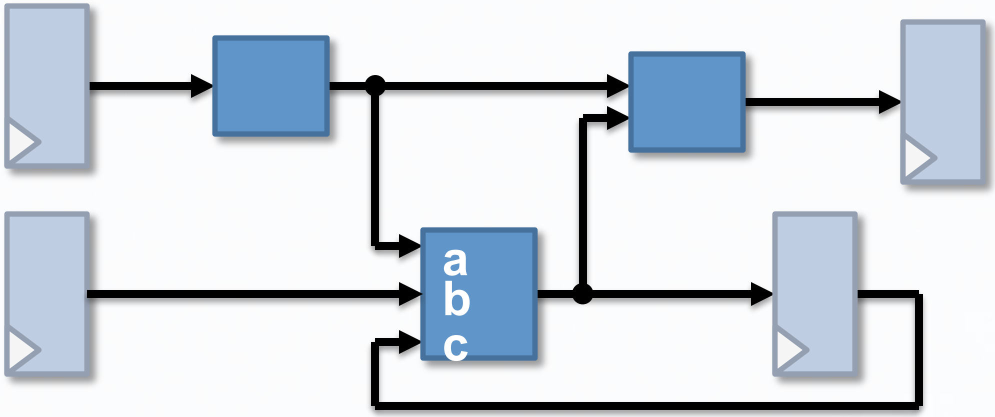
Solutions

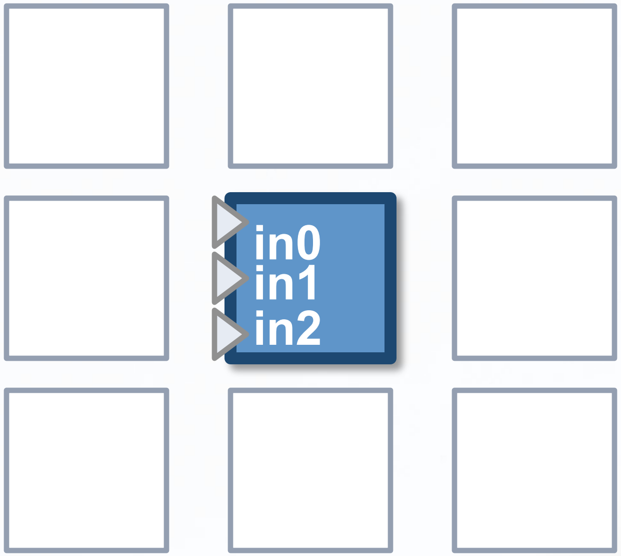
- SAT Formulation
  - Routability Constraints
  - Timing Constraints
- Practicality
  - Dynamic constraint generation
  - Domain-specific variable selection order
  - A*-style future cost clauses
  - Search-and-Repair strategy

Placement Variables



LUT Permutations

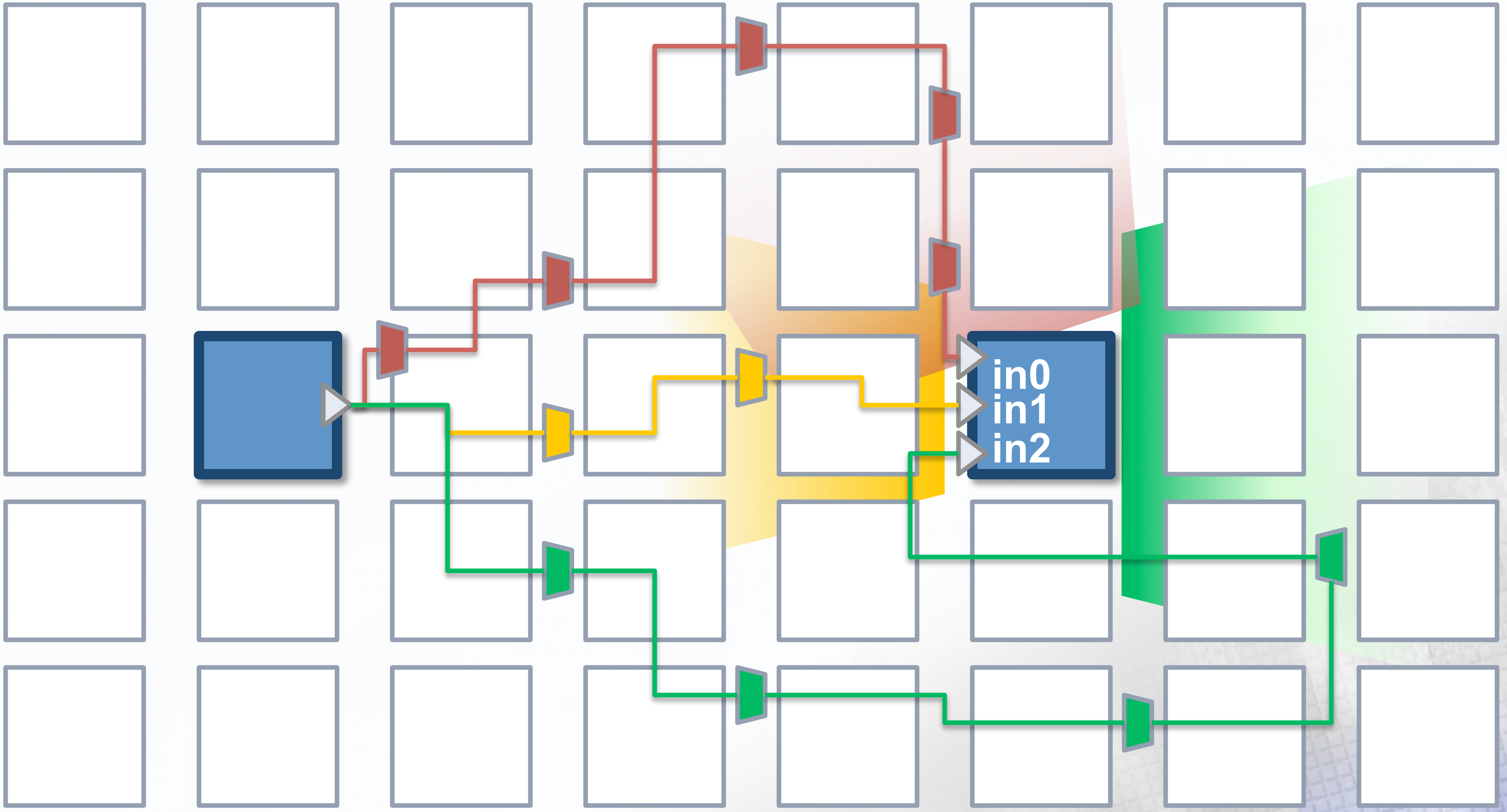




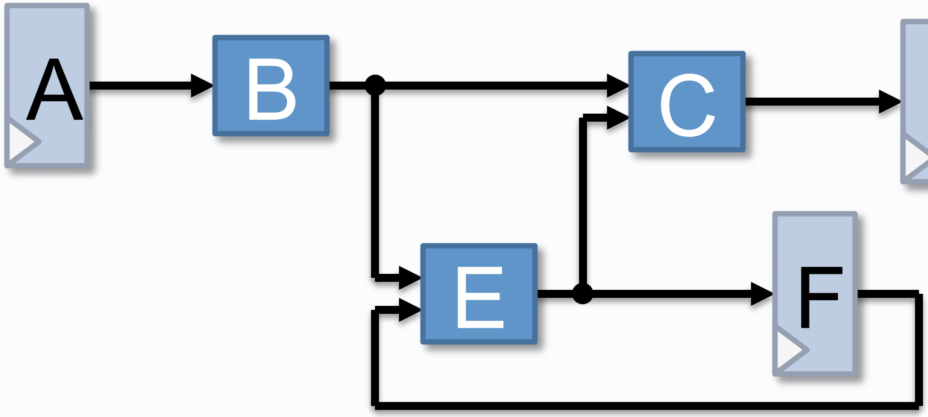
exactlyOne

	Permutation					
Pin	P_0	P_1	P_2	P_3	P_4	P_5
in0	a	a	b	b	c	c
in1	b	c	a	c	a	b
in2	c	b	c	a	b	a

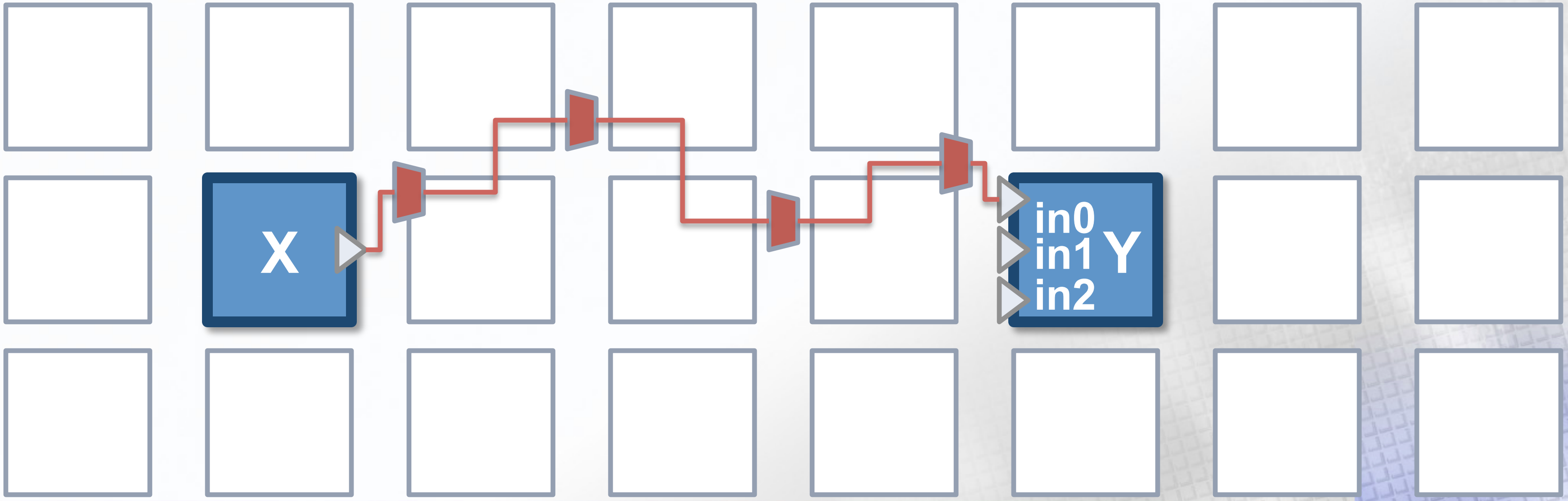
LUT Permutations



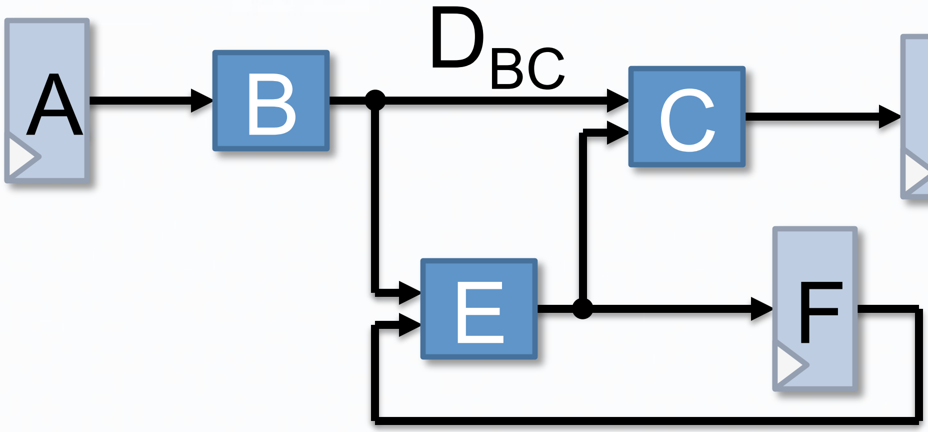
Complete Edge Placement



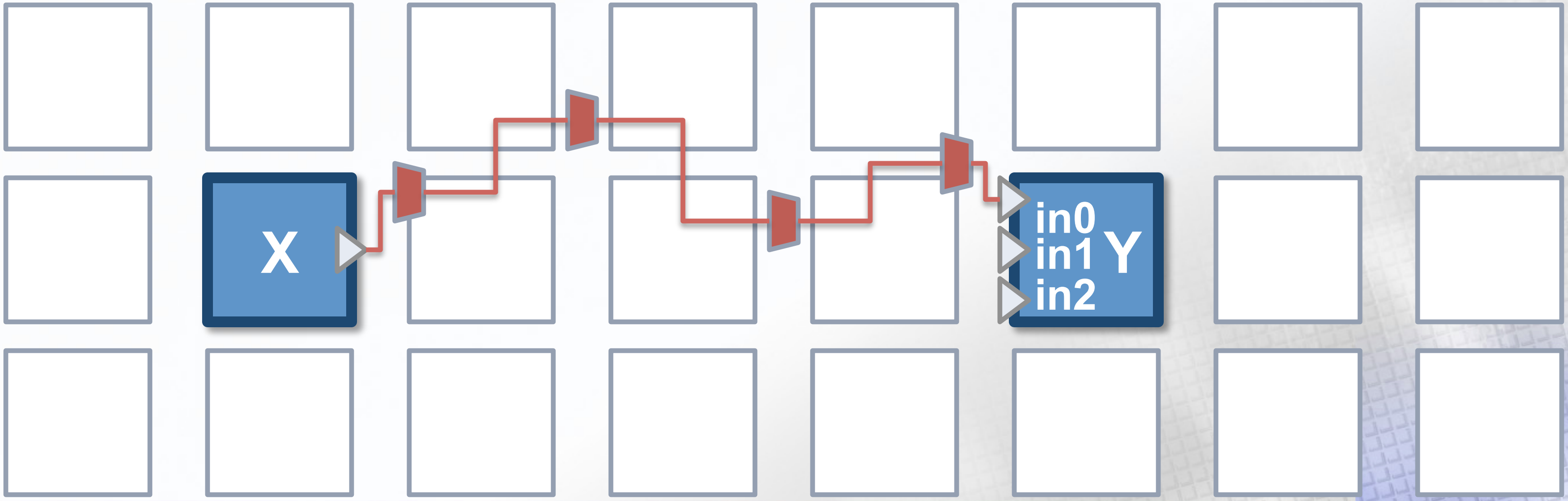
$$(V_{BX} \wedge V_{CY} \wedge P_{C\pi}) \rightarrow C$$



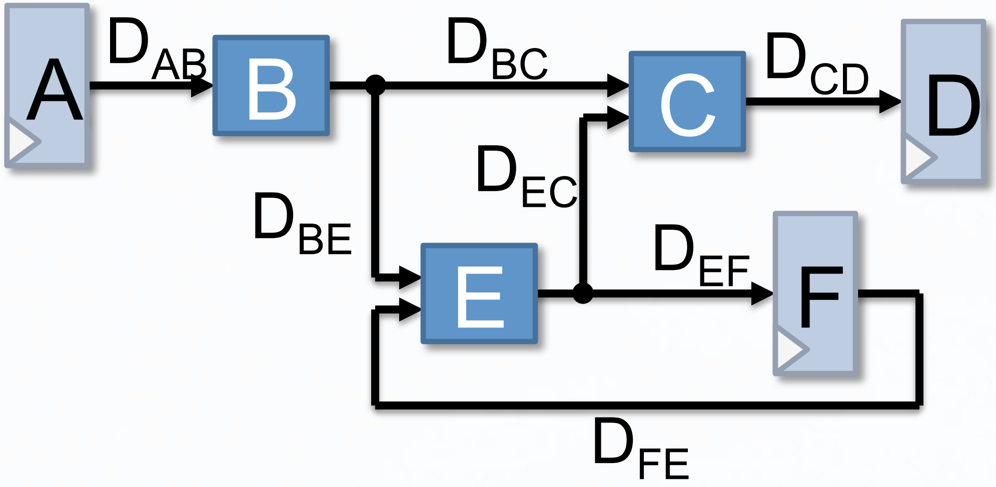
Edge Delays



$$(V_{BX} \wedge V_{CY} \wedge P_{C\pi}) \rightarrow$$
$$D_{BC} = d_{XY[0]}$$

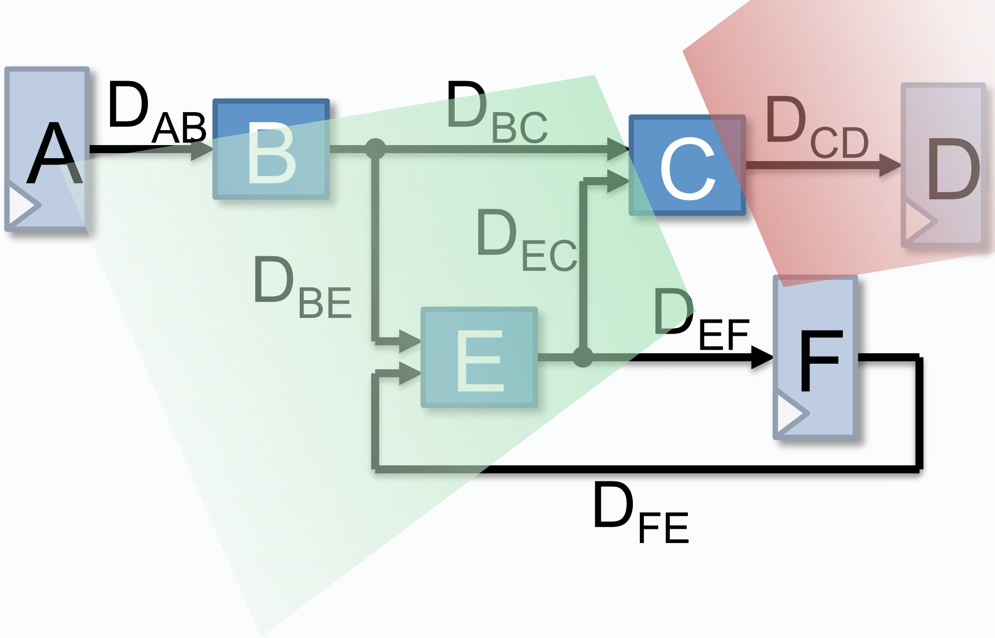


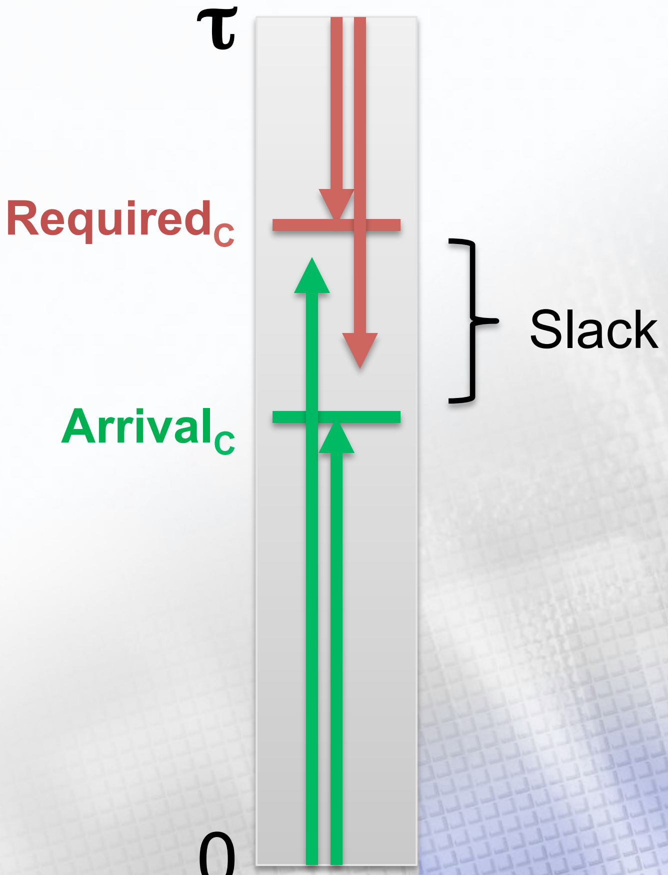
Timing Constraints



Path	Constraint
ABCD	$D_{AB} + D_{BC} + D_{CD} \leq \tau$
ABECD	$D_{AB} + D_{BE} + D_{EC} + D_{CD} \leq \tau$
ABEF	$D_{AB} + D_{BE} + D_{EF} \leq \tau$
FEF	$D_{FE} + D_{EF} \leq \tau$
FECD	$D_{FE} + D_{EC} + D_{CD} \leq \tau$

Static Timing Analysis

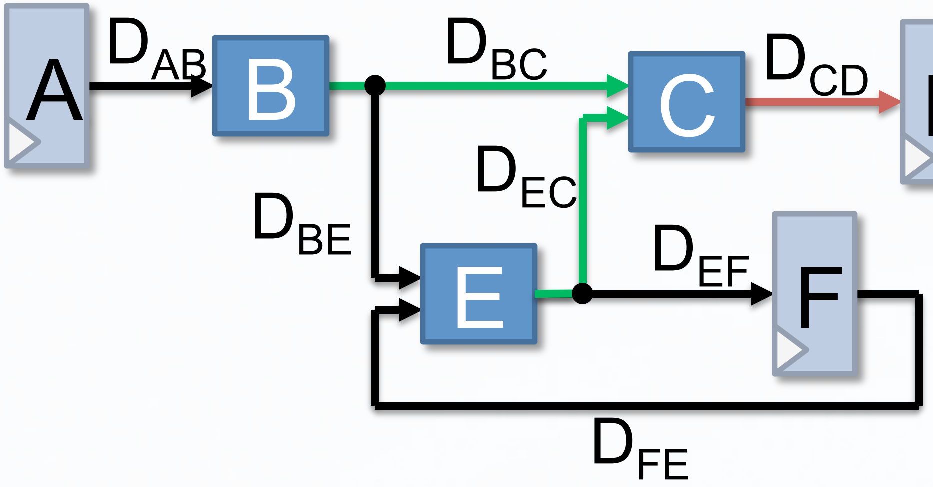




$$\text{Arrival}_C = \max(D_{AB} + D_{BC},\ D_{AB} + D_{BE} + D_{EC},\ D_{FE} + D_{EC})$$

$$\text{Required}_C = \min(\tau - D_{CD},\ \ldots)$$

Arrival and Required Equations



$$Arr_C = \max(Arr_B + D_{BC}, Arr_E + D_{EC})$$

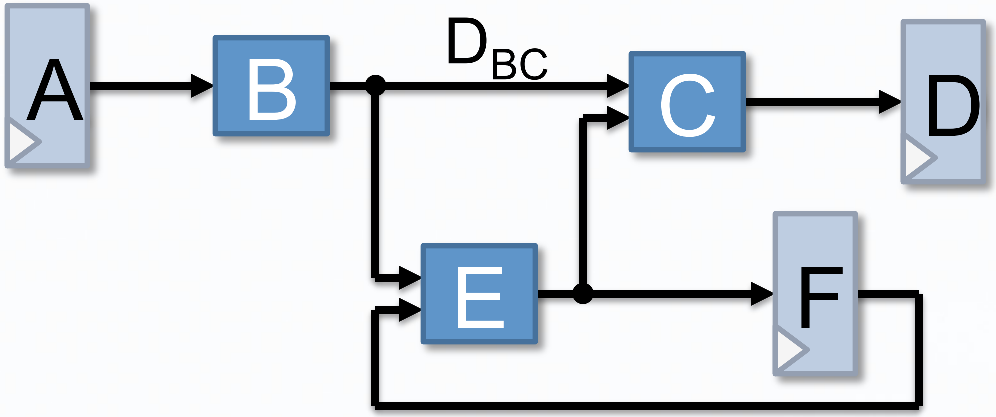
$$Req_C = \min(Req_D - D_{CD})$$

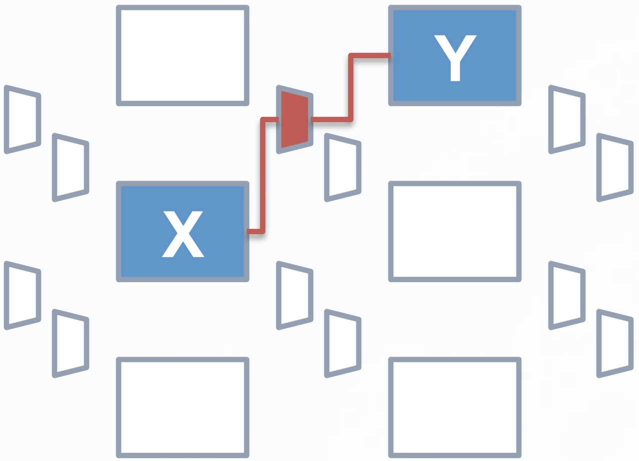
$$Arr_C \geq Arr_B + D_{BC}$$

$$Arr_C \geq Arr_E + D_{EC}$$

$$Req_C \leq Req_D - D_{CD}$$

Difference Logic Formulation

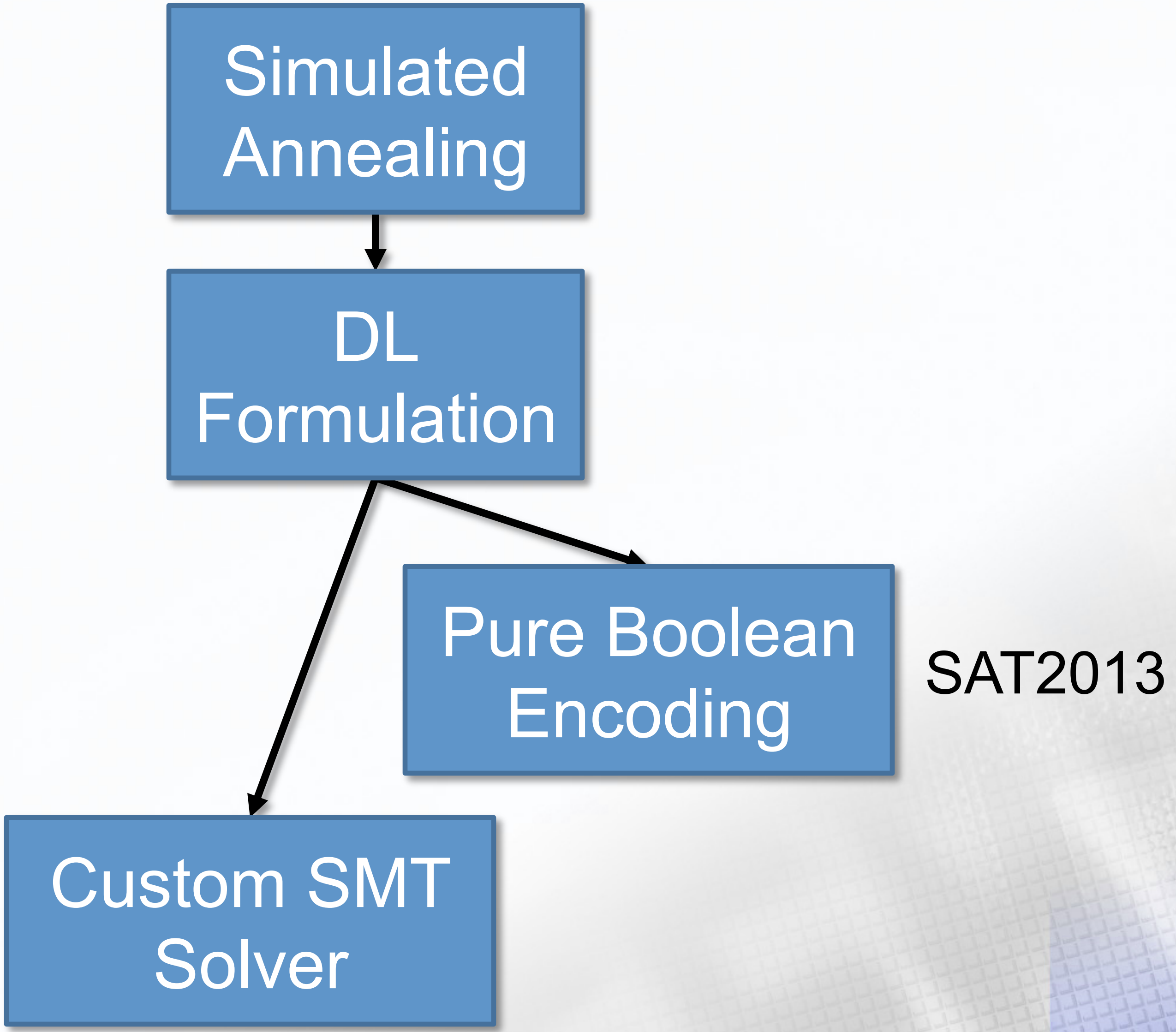




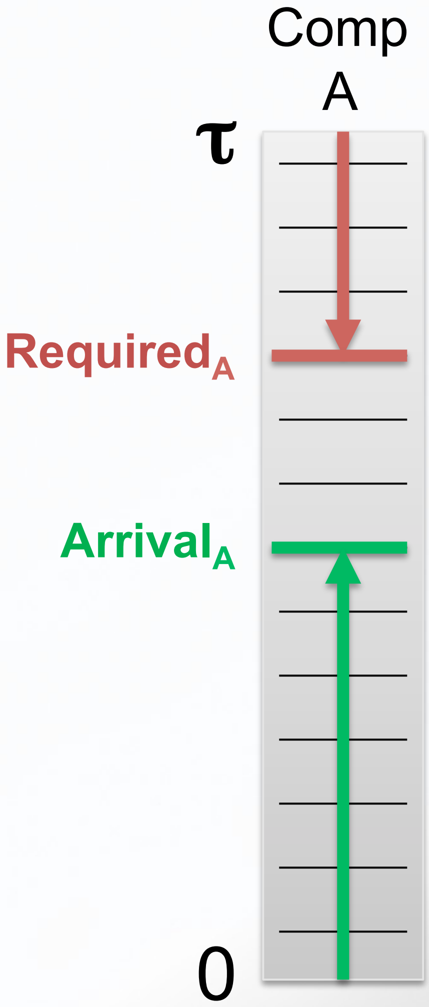
$$\boxed{Arr_N \le Req_N}$$

$$\boxed{\begin{aligned}(V_{BX} \wedge V_{CY} \wedge P_{C\pi}) \rightarrow \\ (Arr_C - Arr_B \ge d_{XY}) \\ \wedge\ (Req_B - Req_C \le -d_{XY})\end{aligned}}$$

Timeline


Simulated Annealing
DL Formulation
Pure Boolean Encoding
SAT2013
SMT2013
Custom SMT Solver

Quantized Arr/Req Times



$\vdots$

$E_{At+2} = T \Leftrightarrow \text{Req}_A \leq t_{+2}$

E_{At+1}

E_{At}

$E_{At-1} = F \Leftrightarrow \text{Arr}_A > t_{-1}$

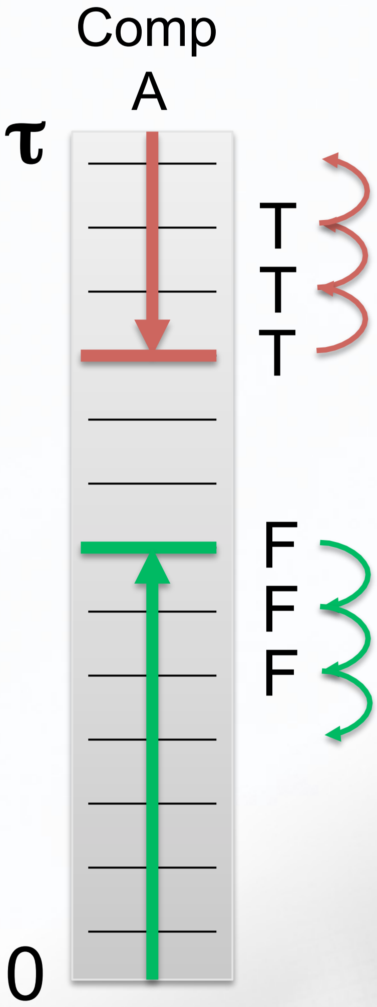
E_{At-2}

$\vdots$

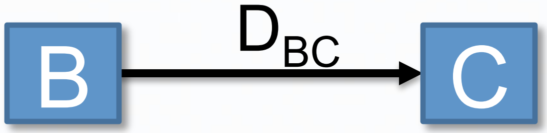
Ohrimenko, Stuckey, Codish
Constraints 2009

Vertical Chain Clauses

$(\neg E_{An} + E_{An+1})$

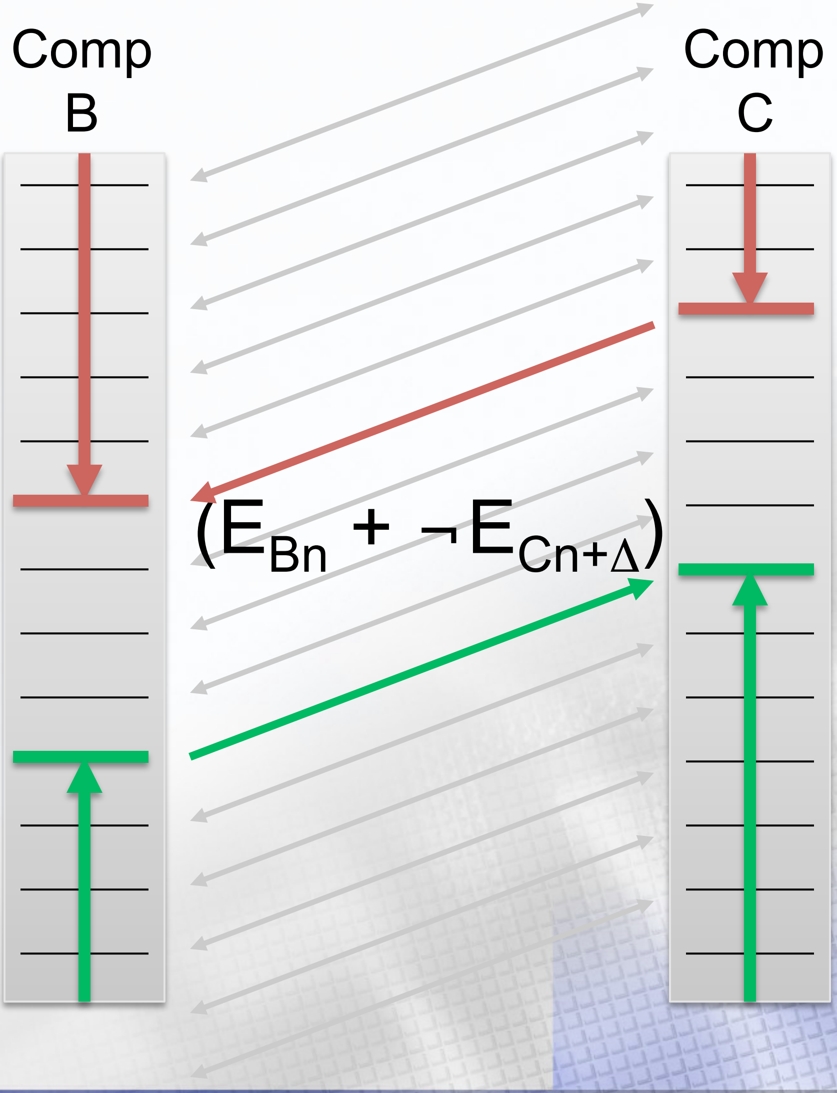


Horizontal Chain Clauses

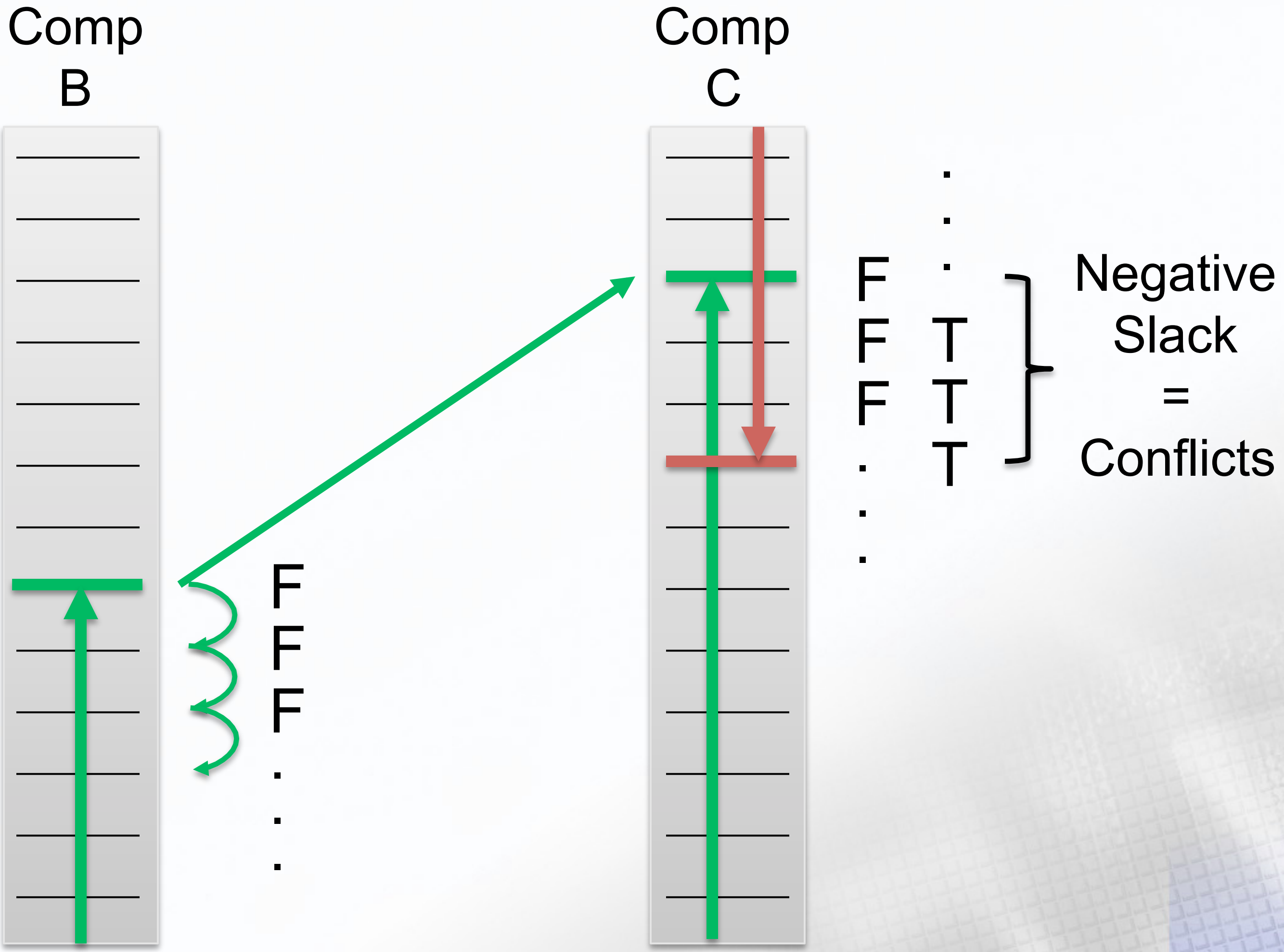


$$(V_{BX} \wedge V_{CY} \wedge P_{C\pi}) \rightarrow$$
$$(Arr_C - Arr_B \geq d_{XY})$$
$$\wedge \ (Req_B - Req_C \leq -d_{XY})$$

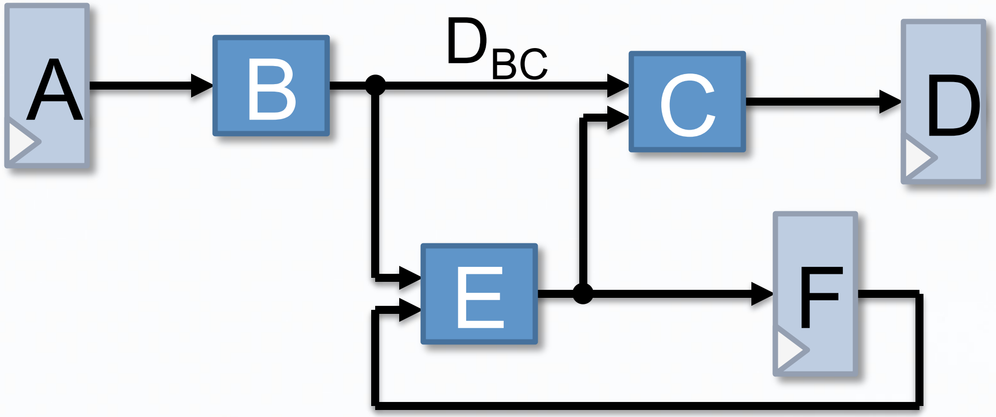
$$(V_{BX} \wedge V_{CY} \wedge P_{C\pi}) \rightarrow$$
$$\bigwedge_n (E_{Bn} + \neg E_{Cn+\Delta})$$



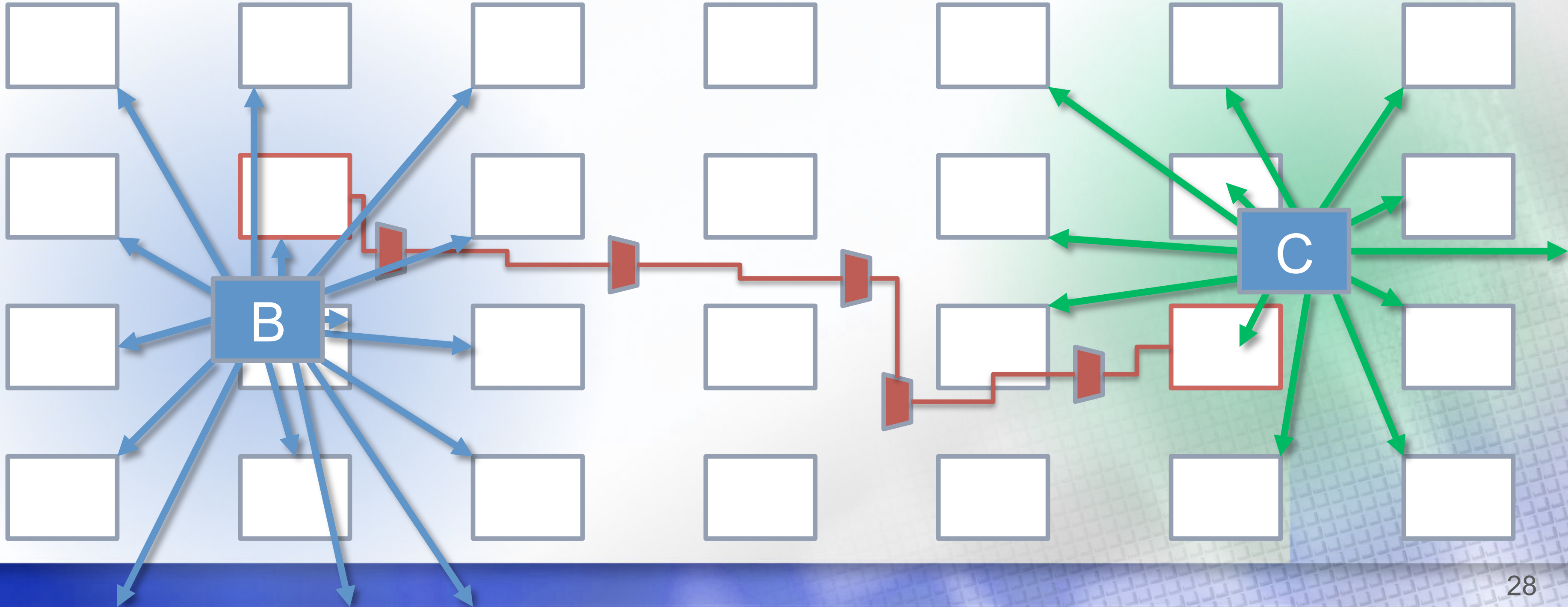
Timing Violation Example



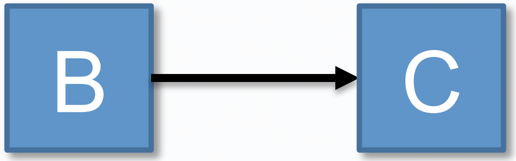
Edge Placement Options



$$(v_{BX} \wedge v_{CY} \wedge P_{C\pi}) \rightarrow (\ldots)$$

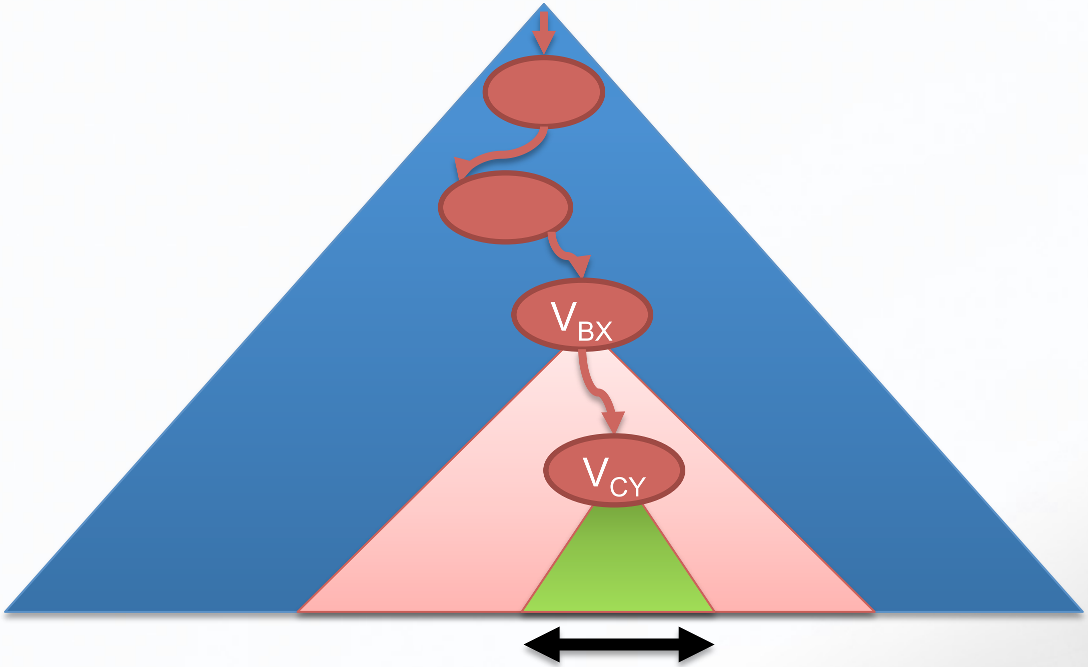


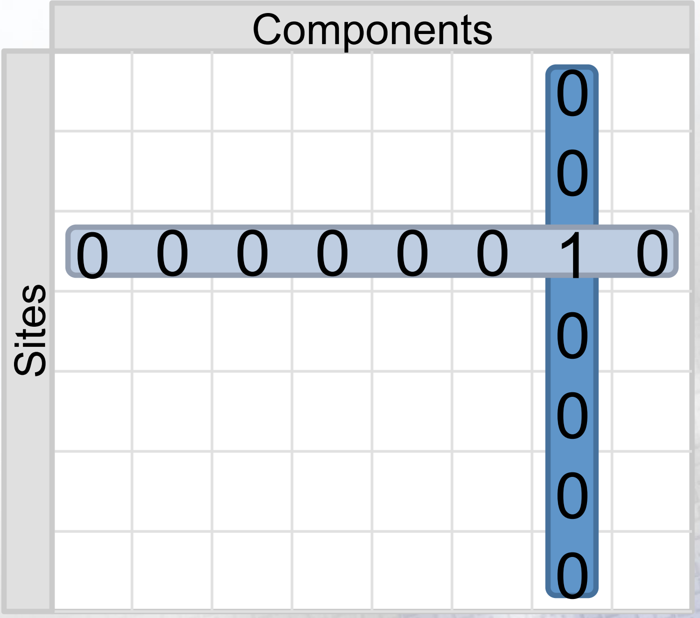
Dynamic Constraint Generation



$$(v_{BX} \wedge v_{CY} \wedge P_{C\pi}) \rightarrow (\ldots)$$

$$(\neg v_{BX} + \neg v_{CY} + \ldots)$$





subspace where (¬V_{BX} + ¬V_{CY} + ...) is relevant

Domain-Specific Variable Selection Order

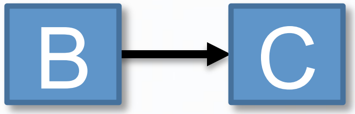
- Choose LUT placements V_{AX} before permutations $P_{A\pi}$

Placement VSIDS Heap: V_{AX} | V_{BX} | V_{CY} | …

Permutation VSIDS Heap: P_{A1} | P_{B0} | P_{B2} | …

- Initialize variable activities
  - Favor placing components on original sites
  - Favor leaving permutations unchanged
  - Favor placing components on deep critical paths first

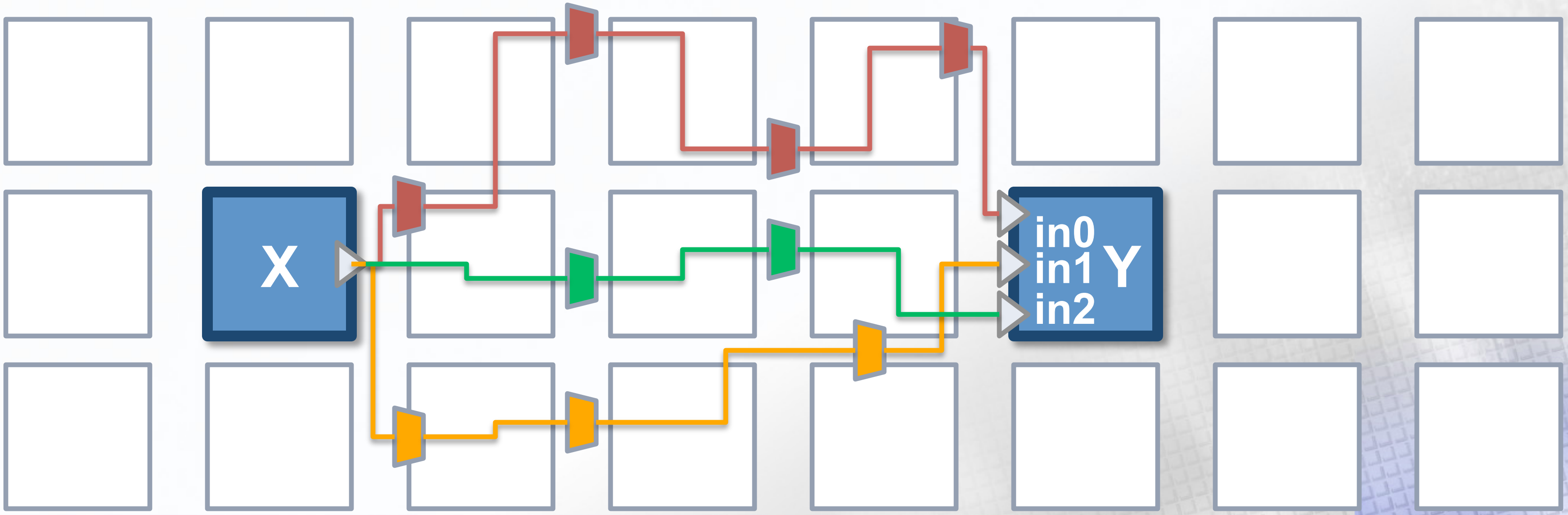
Future Cost Clauses



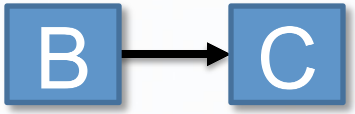
$$(V_{BX} \wedge V_{CY} \wedge P_{C0}) \rightarrow D_{BC} = 500\text{ps}$$

$$(V_{BX} \wedge V_{CY} \wedge P_{C1}) \rightarrow D_{BC} = 455\text{ps}$$

$$(V_{BX} \wedge V_{CY} \wedge P_{C2}) \rightarrow D_{BC} = 430\text{ps}$$



Future Cost Clauses

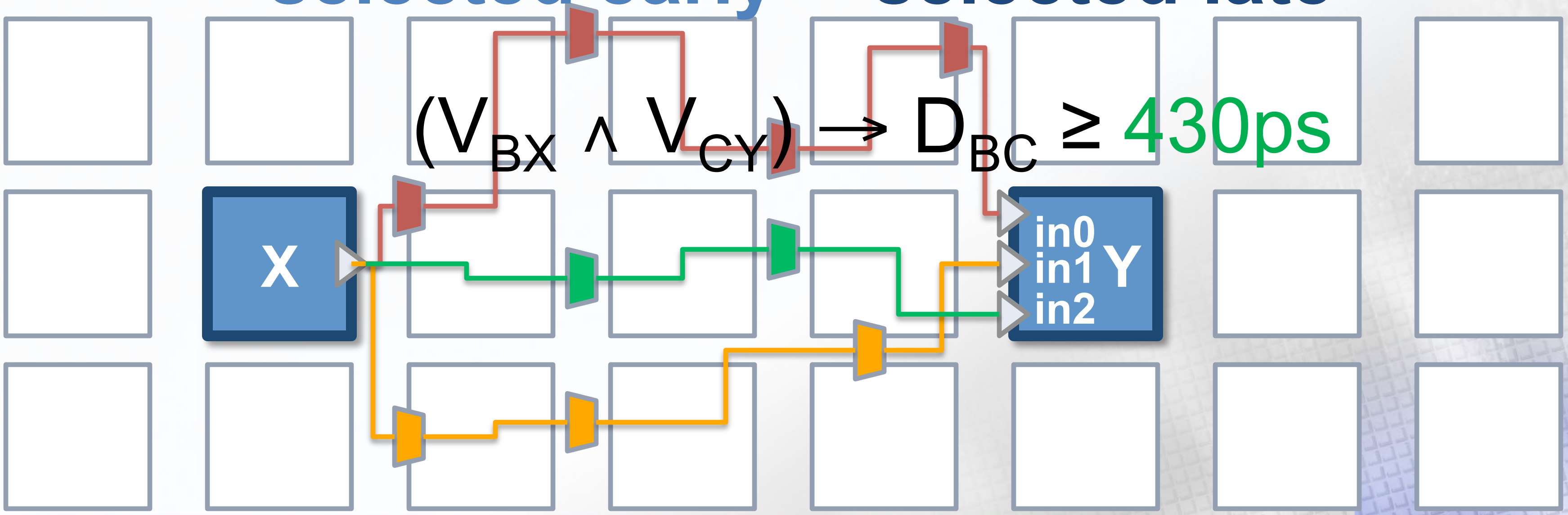


exactlyOne

$$(V_{BX} \wedge V_{CY} \wedge P_{C0}) \rightarrow D_{BC} = 500\text{ps}$$
$$(V_{BX} \wedge V_{CY} \wedge P_{C1}) \rightarrow D_{BC} = 455\text{ps}$$
$$(V_{BX} \wedge V_{CY} \wedge P_{C2}) \rightarrow D_{BC} = 430\text{ps}$$

selected early

selected late



$$(V_{BX} \wedge V_{CY}) \rightarrow D_{BC} \geq 430\text{ps}$$

Results: Annealer vs. SAT

		Annealing		SAT	
Design	LUTs	Runtime	Freq	Runtime	Freq
camellia256	89341	1.0	1.0	0.054	1.14
sudoku	17784	1.0	1.0	0.266	1.49
dct	17199	1.0	1.0	2.526	1.52
wishbone	12775	1.0	1.0	0.028	1.03
fpudouble	12660	1.0	1.0	0.346	1.39
aes	5818	1.0	1.0	2.179	1.20
r2000sc	5095	1.0	1.0	0.788	1.28
sha256	3368	1.0	1.0	0.106	1.14

Results: Dynamic Constraint Generation

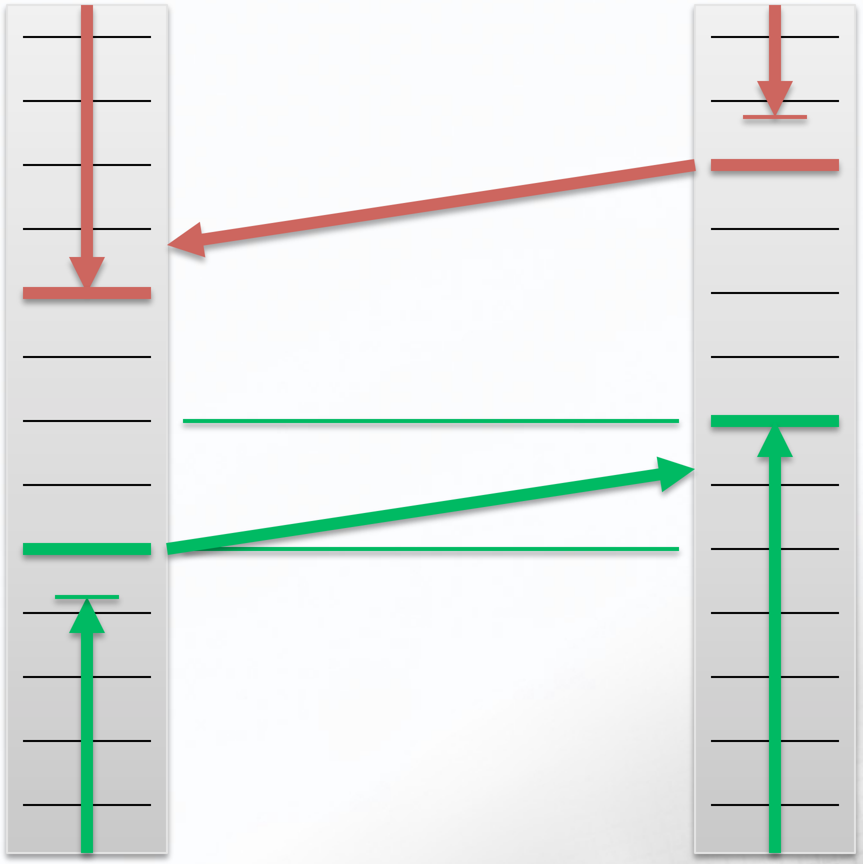
		Dynamic			No Dynamic	
Design	**LUTs**	**Runtime**	**Vars**	**Clauses**	**Runtime**	**Clauses**
camellia256	89341	1.0	751k	30k	30.29	8099k
sudoku	17784	1.0	758k	31k	7.40	743k
dct	17199	1.0	1512k	116k	4.49	8153k
wishbone	12775	1.0	1081k	25k	25.53	17449k
fpudouble	12660	1.0	1598k	74k	4.06	3730k
aes	5818	1.0	908k	24k	2.71	5070k
r2000sc	5095	1.0	1404k	40k	7.95	22285k
sha256	3368	1.0	2100k	39k	11.68	16908k

Results: Variable Order and A* Clauses

		No Var Order		No A* Clauses	
Design	**LUTs**	**Runtime**	**Freq**	**Runtime**	**Freq**
camellia256	89341	5.31	0.89	1.18	1.00
sudoku	17784	5.25	0.97	1.08	1.00
dct	17199	8.37	0.94	1.65	1.06
wishbone	12775	5.00	0.99	1.45	1.00
fpudouble	12660	6.73	0.82	2.41	1.09
aes	5818	7.45	0.99	1.17	1.00
r2000sc	5095	8.71	0.99	1.84	1.00
sha256	3368	10.96	1.00	1.19	1.00

Switching from SAT to SMT

- Quantized arr/req times and delays overconstrain the problem



Summary

- First use of SAT for Detailed Placement
- Encoding is not enough to make it practical
- Exploit natural subdivisions in the problem space
- SMT version currently deployed in a production placement tool at Tabula