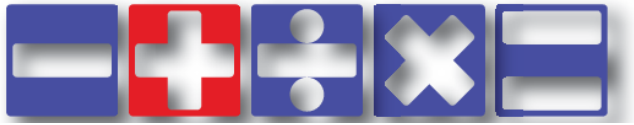
31st IEEE International Symposium on Computer Arithmetic

ARITH 2024



Combining Power and Arithmetic Optimization via Datapath Rewriting

Samuel Coward, Emiliano Morini, Theo Drane, George A. Constantinides

Numerical Hardware Group, Intel Corporation

EEE, Imperial College London

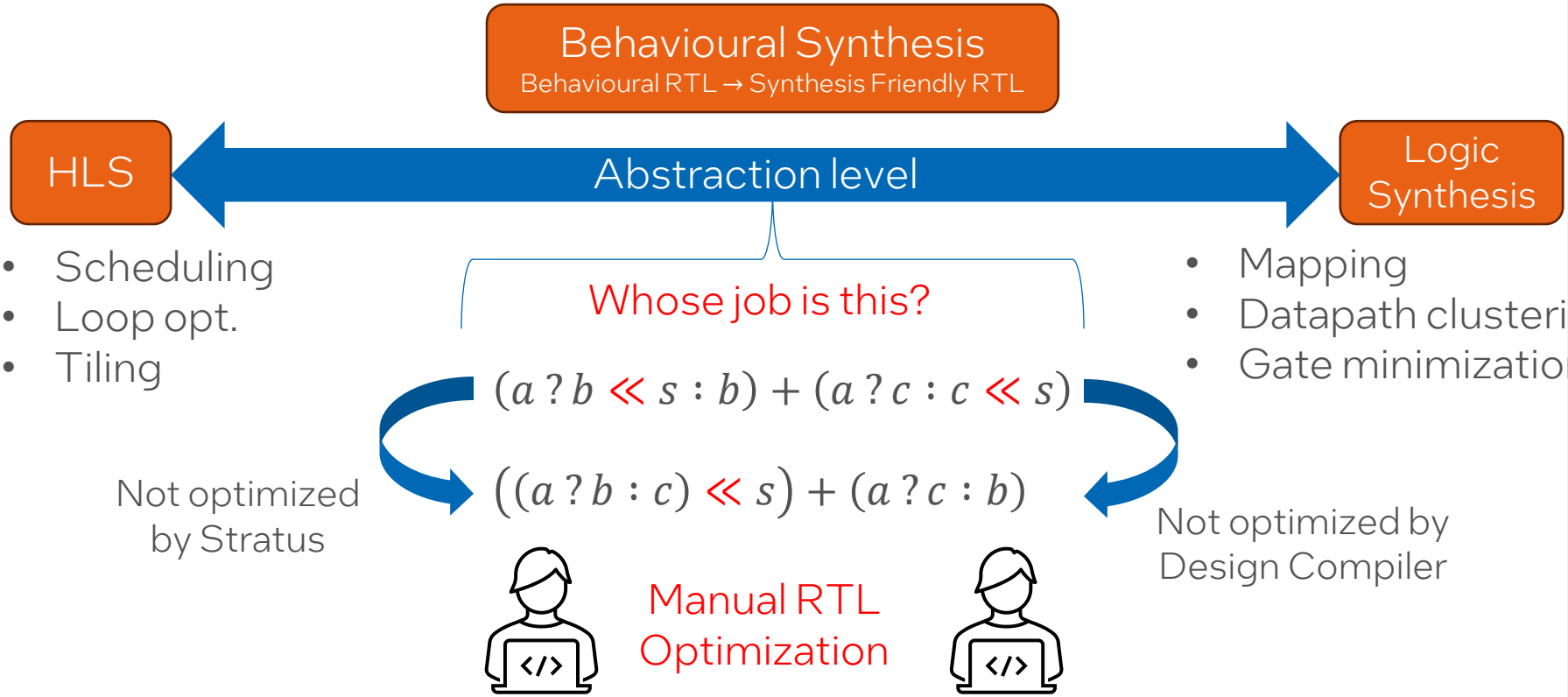
intel

Motivation

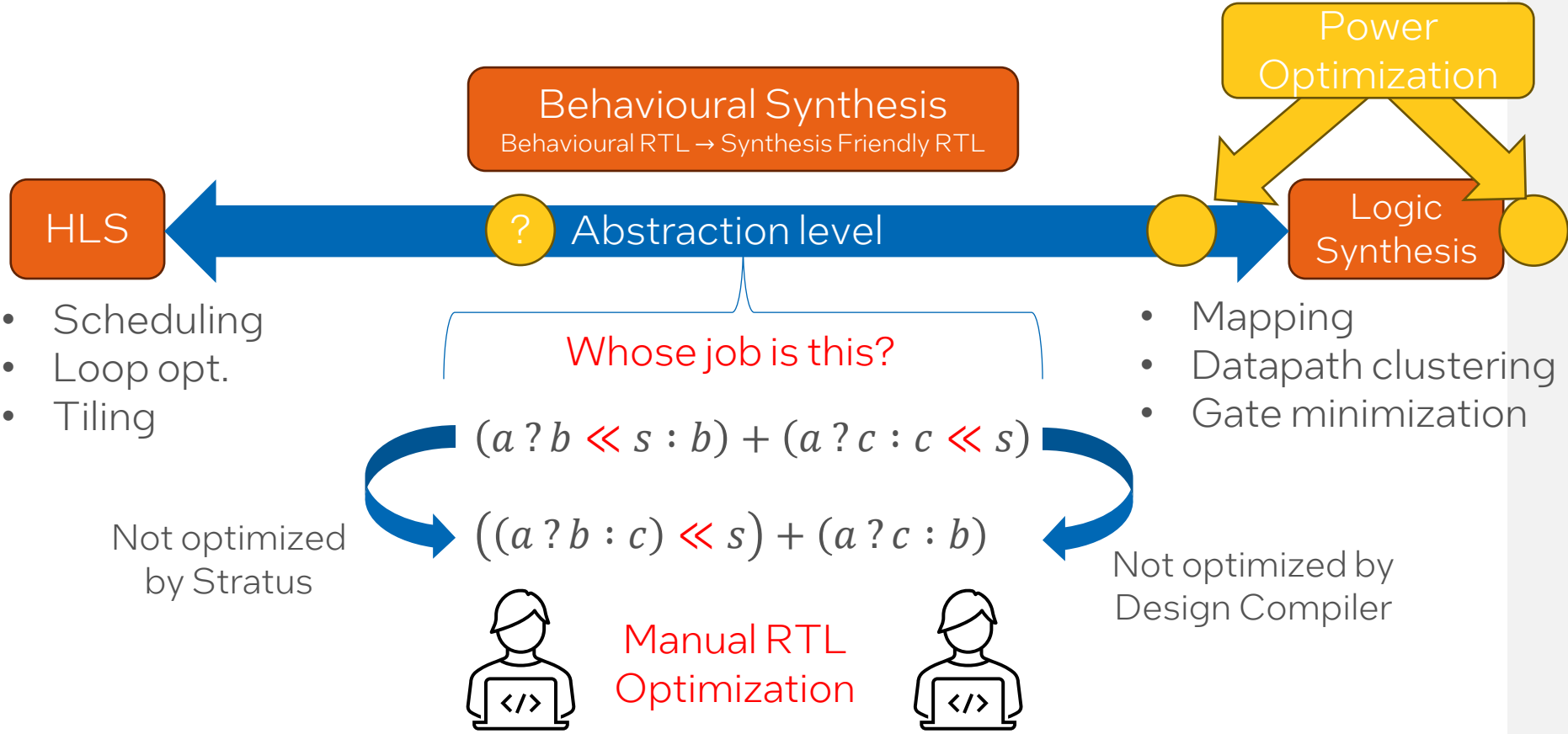
- Scheduling
- Loop opt.
- Tiling

- Mapping
- Datapath clustering
- Gate minimization

Motivation


Behavioural Synthesis
Behavioural RTL → Synthesis Friendly RTL
HLS
Abstraction level
Logic Synthesis
• Scheduling
• Loop opt.
• Tiling
Whose job is this?
• Mapping
• Datapath clustering
• Gate minimization
$(a\ ?\ b \ll s : b) + (a\ ?\ c : c \ll s)$
$((a\ ?\ b : c) \ll s) + (a\ ?\ c : b)$
Not optimized by Stratus
Not optimized by Design Compiler
Manual RTL Optimization

Motivation


Power Optimization
Behavioural Synthesis
Behavioural RTL → Synthesis Friendly RTL
HLS
?
Abstraction level
Logic Synthesis
• Scheduling
• Loop opt.
• Tiling
• Mapping
• Datapath clustering
• Gate minimization
Whose job is this?
(a ? b ≪ s : b) + (a ? c : c ≪ s)
((a ? b : c) ≪ s) + (a ? c : b)
Not optimized by Stratus
Not optimized by Design Compiler
Manual RTL Optimization

E-Graphs

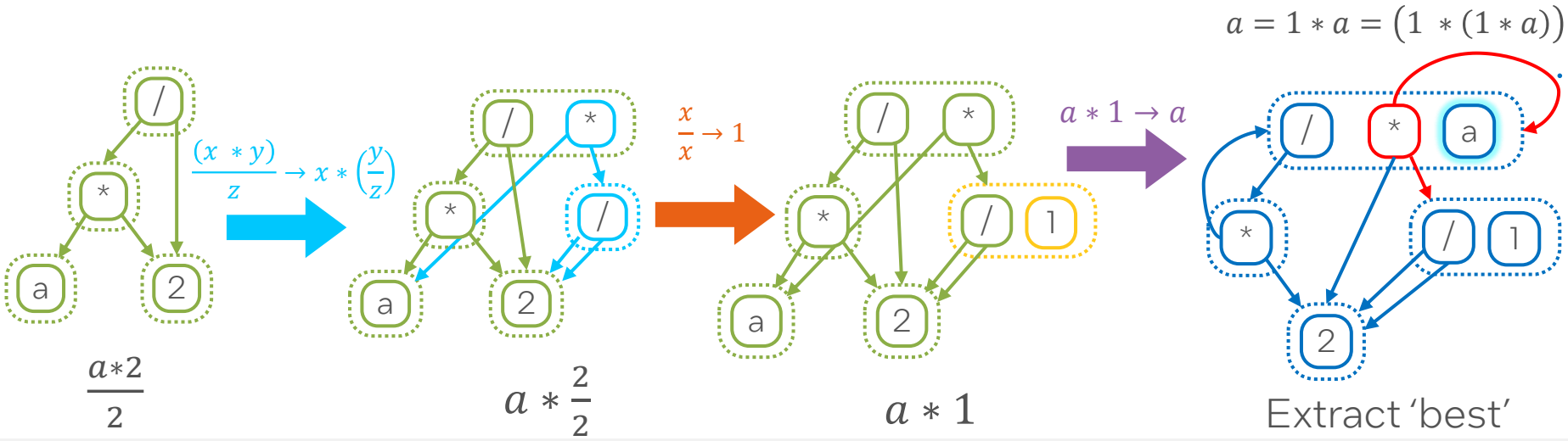
egg: e-graphs good

Z3

CVC5

fastly

- Compact representation of equivalent designs
- Maintains history
- Combine program analysis & rewriting
- Constructive rewrite application – phase ordering



E-Syn: E-Graph Rewriting with Technology-Aware Cost Functions for Logic Synthesis

Chen Chen*
HKUST(GZ)
cchen099@connect.hkust-gz.edu.cn

Guangyu Hu*
HKUST
ghuae@connect.ust.hk

Dongsheng Zuo
HKUST(GZ)
dzuo721@connect.hkust-gz.edu.cn

Equality Saturation for Datapath Synthesis: A Pathway to Pareto Optimality

Ecenur Ustun[1], Cunxi Yu[2], and Zhiru Zhang[1]

ESFO: Equality Saturation for FIRRTL Optimization

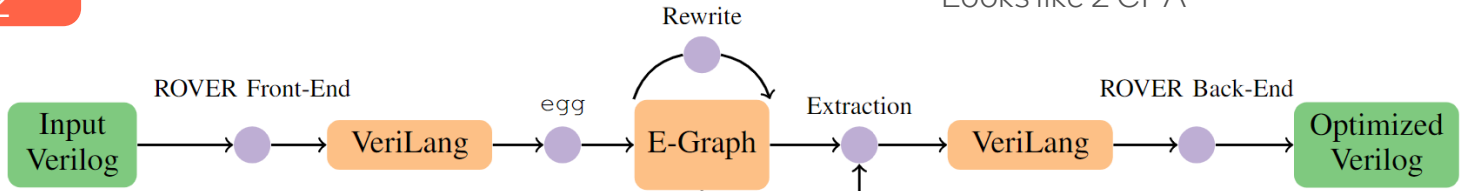
Yan Pi Hongji Zou Tun Li

There and Back Again: A Netlist's Tale with Much Egraphin'

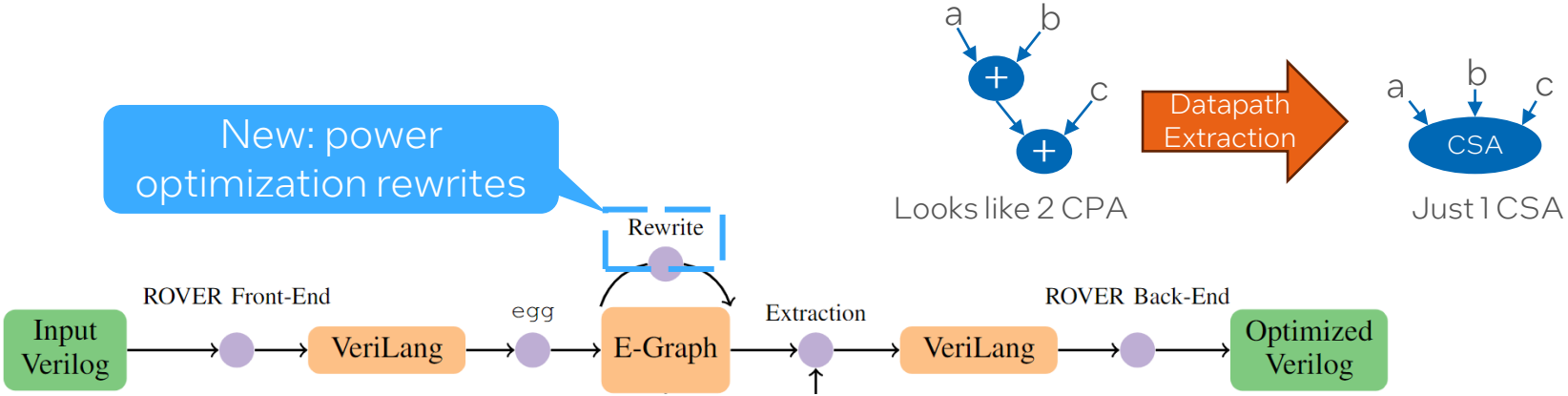
Gus Henry Smith†, Zachary D. Sisco‡, Thanawat Techaumnuaiwit‡, Jingtao Xia‡, Vishal Canumalla†, Andrew Cheung†, Zachary Tatlock†, Chandrakana Nandi§, Jonathan Balkind‡
† University of Washington ‡ University of California, Santa Barbara § Certora, Inc.

ROVER – RTL Opt. via Verified E-Graph Rewriting

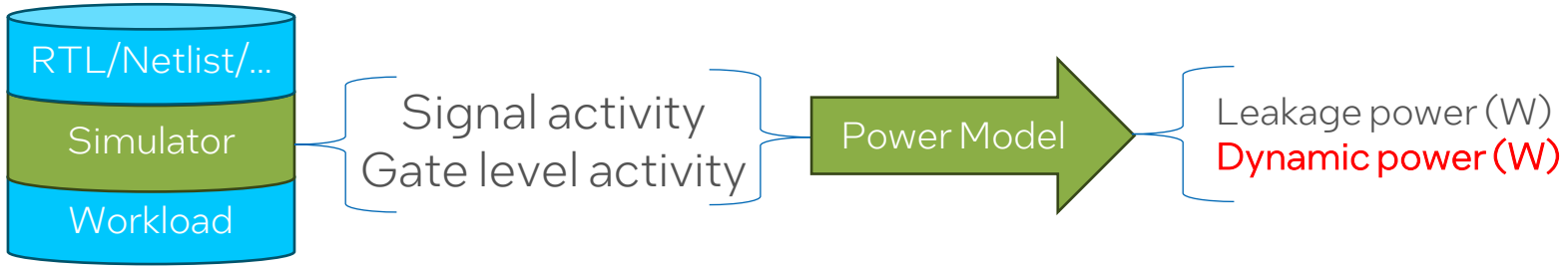
ARITH 2022



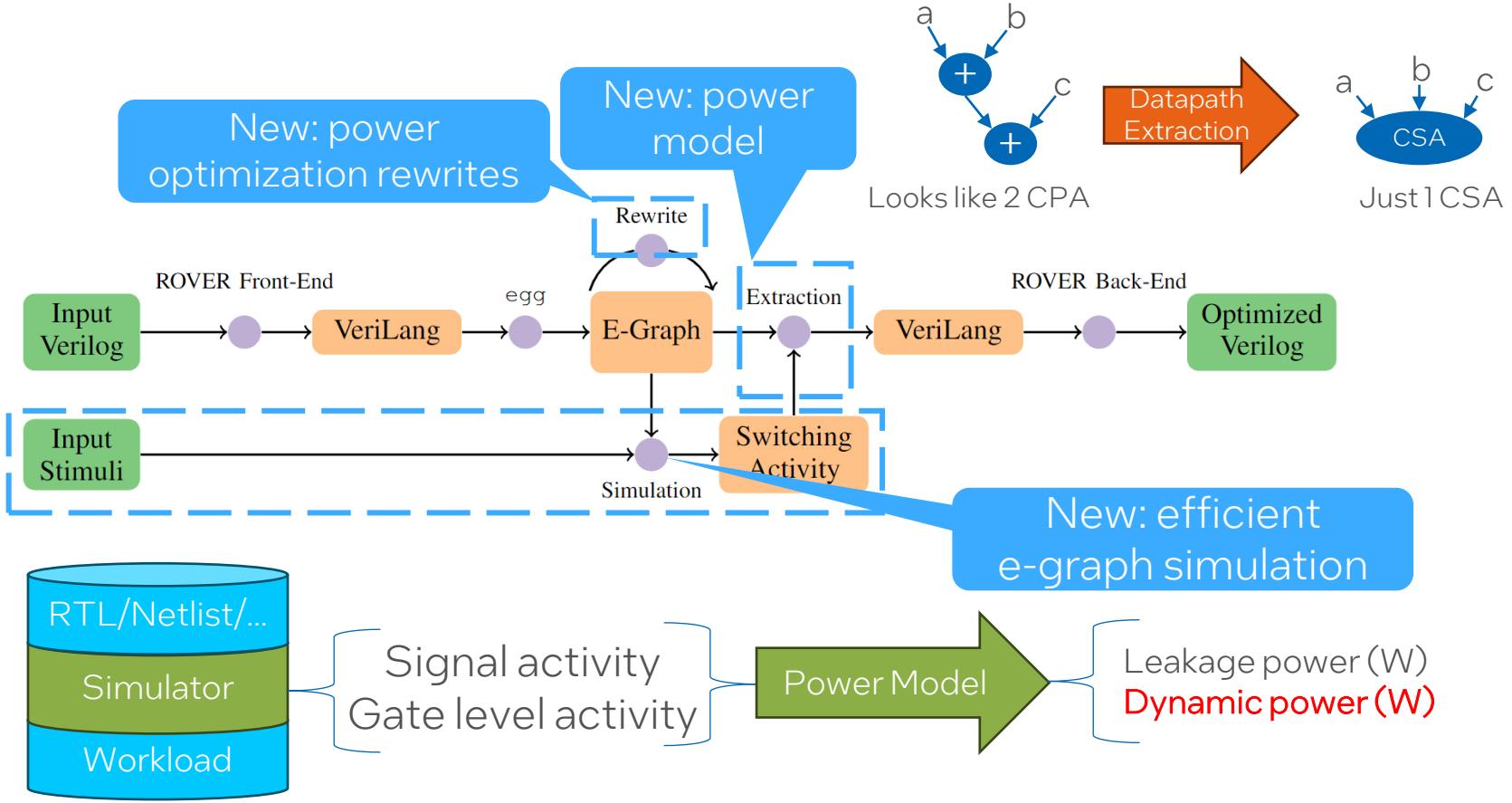
ROVER – RTL Opt. via Verified E-Graph Rewriting



Measuring Power



ROVER – RTL Opt. via Verified E-Graph Rewriting



Don't Cares – hardware is not software

```
if a:
    f(x)
else:
    g(y)
```

Software – lazy evaluation:
f(x) only computed if a=true
Hardware – eager evaluation:
f(x) and g(y) always evaluated

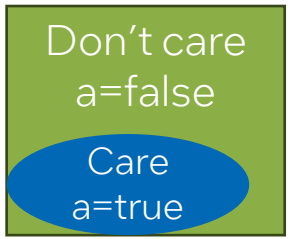
- Muxes introduce potential redundancy
- Redundant computations = wasted power

DAC 2023 – exploit don't care for performance

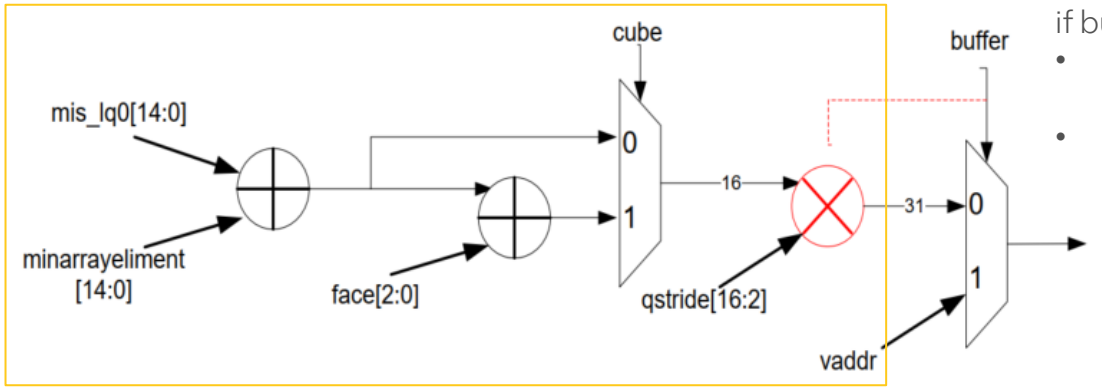
$$a\ ?\ f(x) : g(y) \rightarrow a\ ?\ f'(x) : \mathrm{g(y)}$$

Such that:

$$f'(x) = \begin{cases} f(x) & if\ a \\ don't\ care & else \end{cases}$$



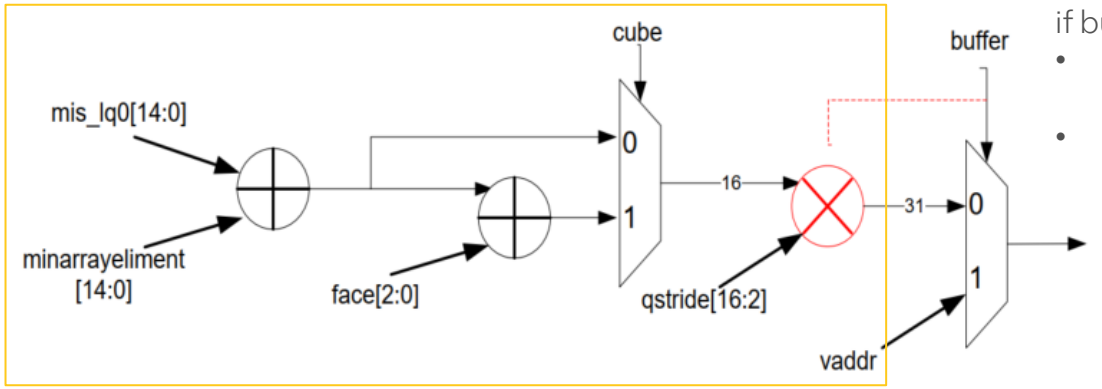
Redundant Computation & Wasted Watts



if buffer=1

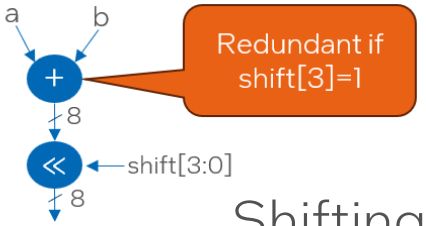
- redundant computation
- wasted power

Redundant Computation & Wasted Watts

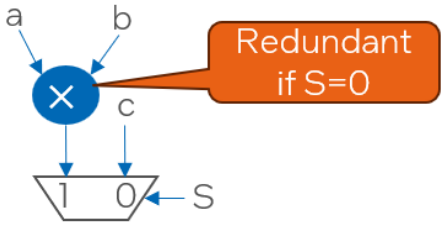


if buffer=1
- redundant computation
- wasted power

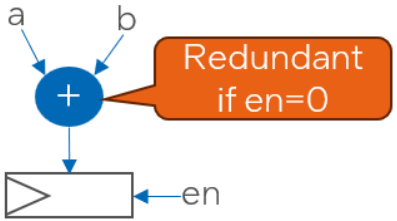
Sources of Redundancy



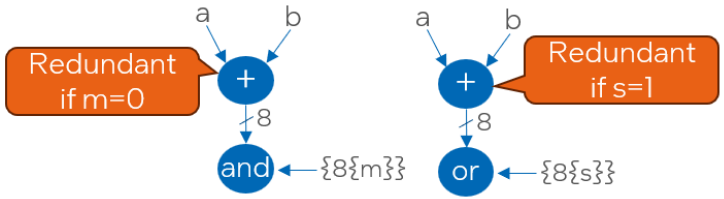
Shifting



Muxes



Enabled Registers

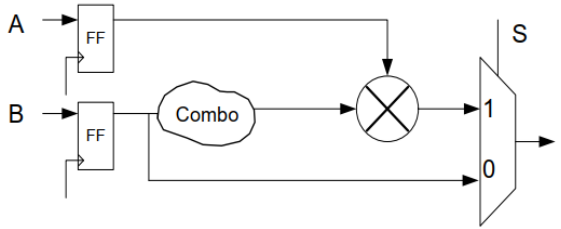


Masking/Saturating

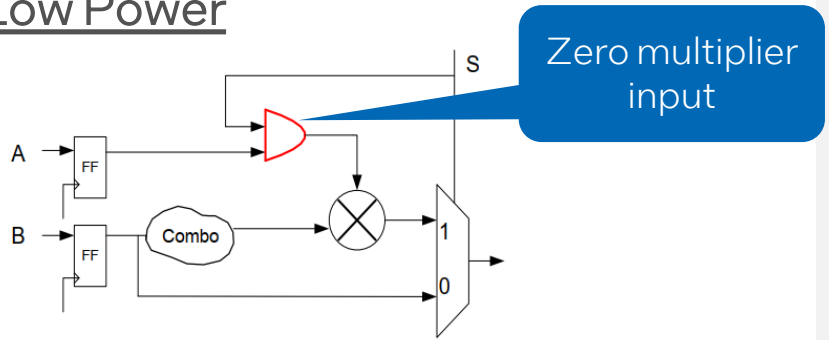
1) Data Gating/Operand Isolation – Zeroing Datapath

- Create a mask from select signal
- Bitwise AND with datapath inputs

Original



Low Power

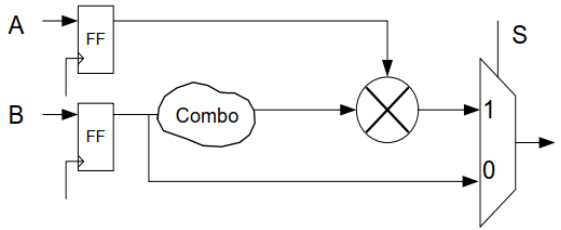


1) Data Gating/Operand Isolation – Zeroing Datapath

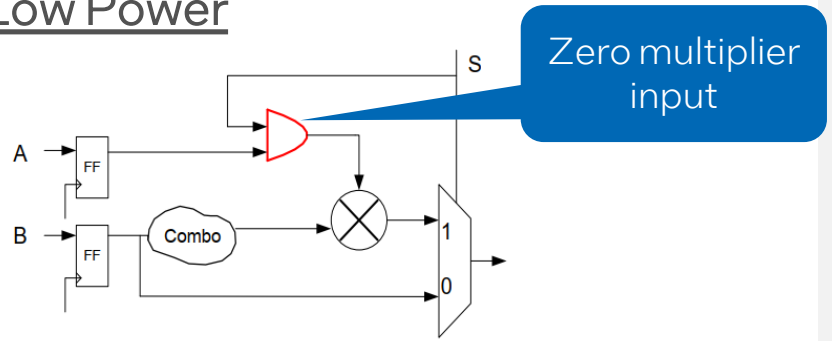
- Create a mask from select signal
- Bitwise AND with datapath inputs

Left	Right
$s\ ?\ b : c$	$s\ ?\ (b\ \&\ \{w_b\{s\}\}) : (c\ \&\ \{w_c\{\bar{s}\}\})$
$(a\ op\ b)\ \&\ \{w_o\{s\}\}$	$(a\ \&\ \{w_o\{s\}\})\ op\ (b\ \&\ \{w_b\{s\}\})$
$(a\ \&\{w_a\{s_1\}\})\ \&\ \{w_a\{s_2\}\}$	$(a\ \&\{w_a\{s_1\&s_2\}\})$

Original



Low Power

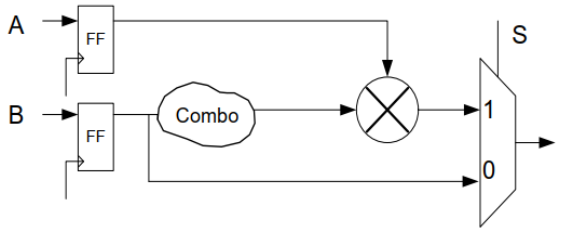


1) Data Gating/Operand Isolation – Zeroing Datapath

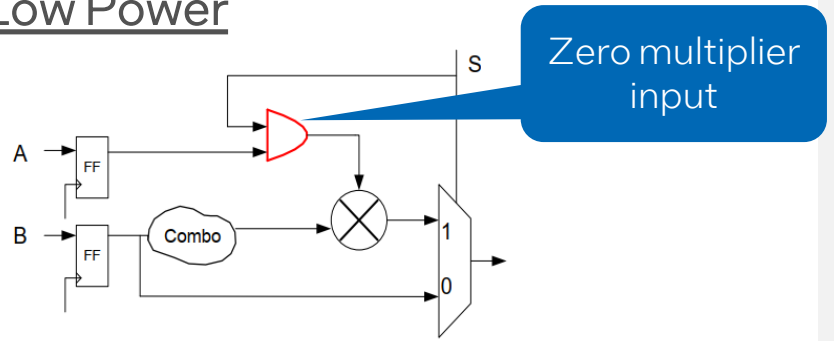
- Create a mask from select signal
- Bitwise AND with datapath inputs

Left	Right
$s\ ?\ b : c$	$s\ ?\ (b\ \&\ \{w_b\{s\}\}) : (c\ \&\ \{w_c\{\bar{s}\}\})$
$(a\ op\ b)\ \&\ \{w_o\{s\}\}$	$(a\ \&\ \{w_o\{s\}\})\ op\ (b\ \&\ \{w_b\{s\}\})$
$(a\ \&\{w_a\{s_1\}\})\ \&\ \{w_a\{s_2\}\}$	$(a\ \&\{w_a\{s_1\&s_2\}\})$

Original



Low Power

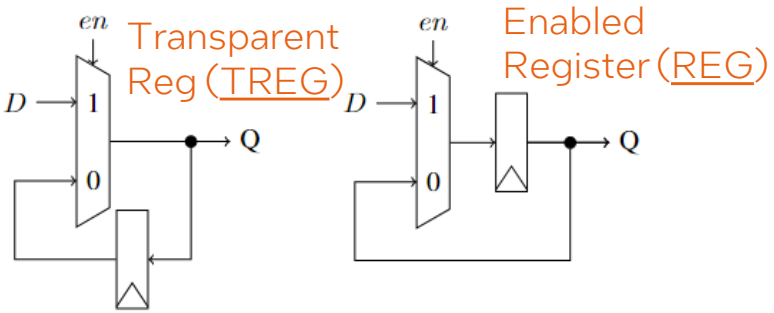


Considerations:

- Toggle rate of select signal S
- Toggle rate of data signals A,B
- Timing impact & gate count

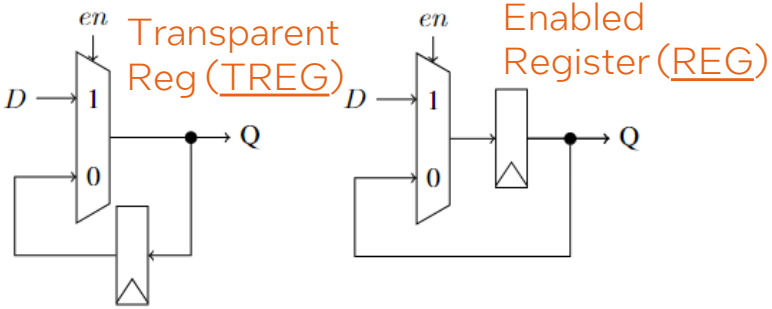
2) Transparent Latch/Register – Operand Isolation

- When enabled is transparent
- When disabled input is “frozen” – no power consumed in datapath

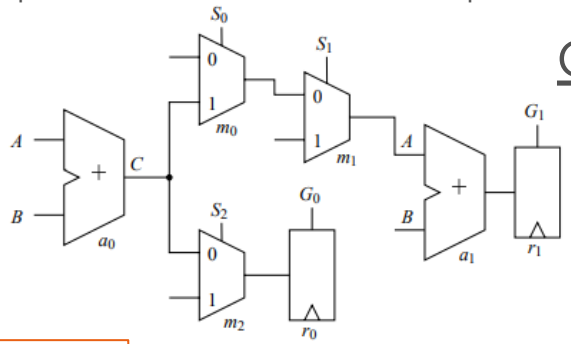


2) Transparent Latch/Register – Operand Isolation

- When enabled is transparent
- When disabled input is "frozen" – no power consumed in datapath

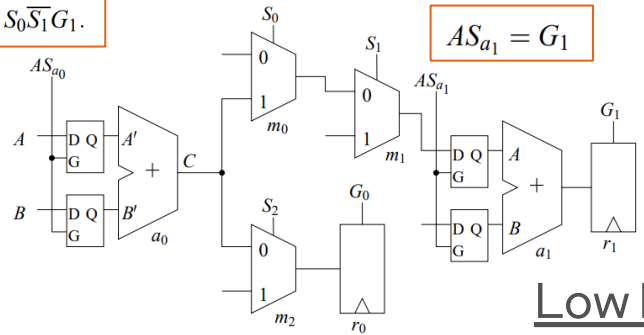


Original



$\overline{S_2}G_0 + S_0\overline{S_1}G_1.$

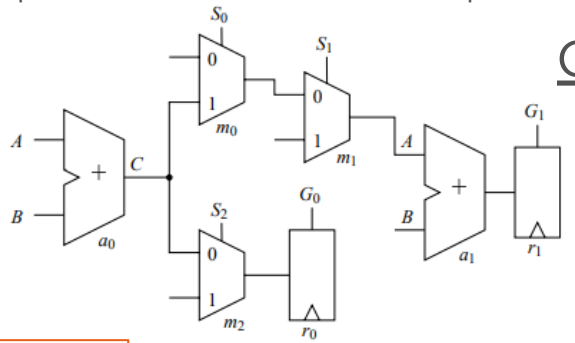
$AS_{a_1} = G_1$



Low Power

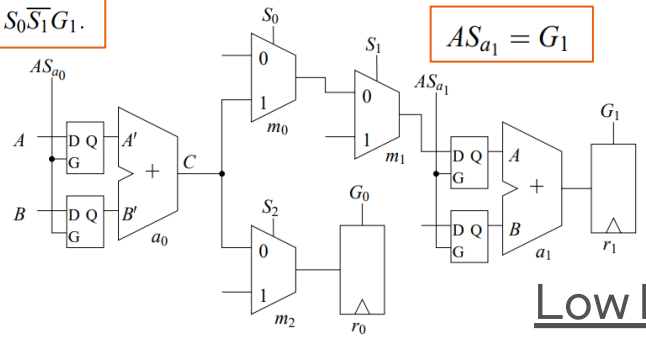
2) Transparent Latch/Register – Operand Isolation

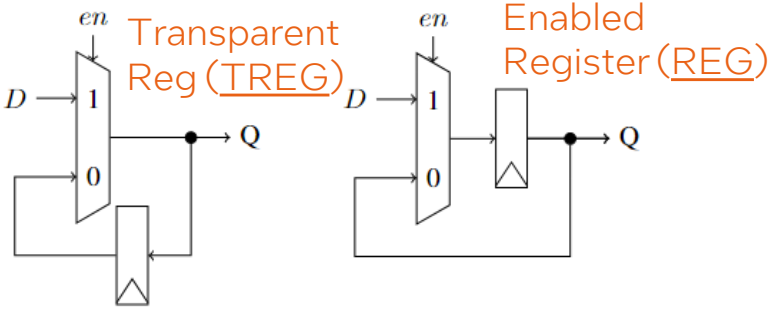
- When enabled is transparent
- When disabled input is "frozen" – no power consumed in datapath



$\overline{S_2}G_0 + S_0\overline{S_1}G_1.$

$AS_{a_1} = G_1$





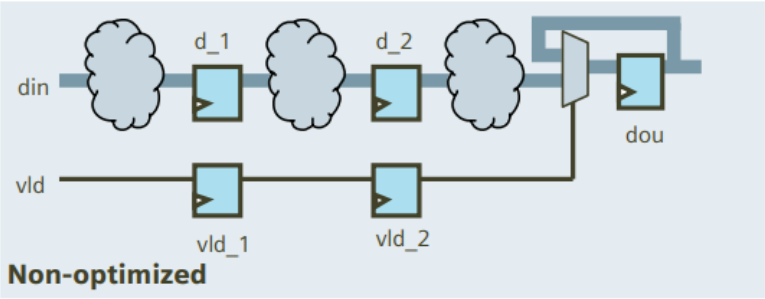
Left	Right
$s\ ?\ b : c$	$s\ ?\ TREG(b,s) : TREG(c,\bar{s})$
$TREG(a\ op\ b, s)$	$TREG(a,s)\ op\ TREG(b,s)$
$REG(a, en)$	$REG(TREG(a,en), en)$

Considerations:

- Clock gating available?
- Timing impact & gate count

3) Clock Gating – Switching off Datapath

- Modify register enable signals
- Construct *observability don't care* conditions

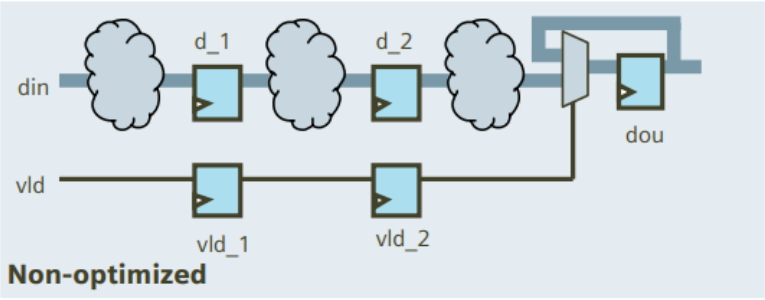


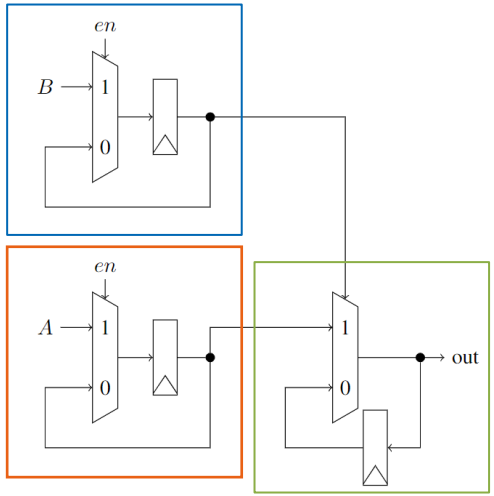
[Siemens White Paper]

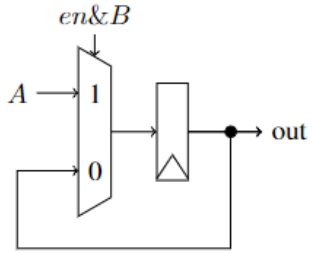
3) Clock Gating – Switching off Datapath

- Modify register enable signals
- Construct *observability don't care* conditions

$$TREG\big(REG(A, en), REG(B, en)\big) \rightarrow REG(A, en \,\&\, B)$$







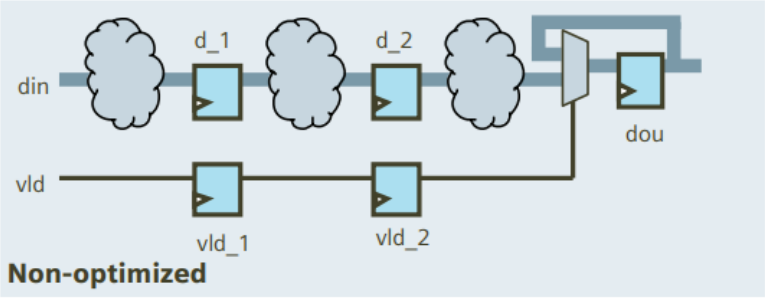
^Gated Reg^

[Siemens White Paper]

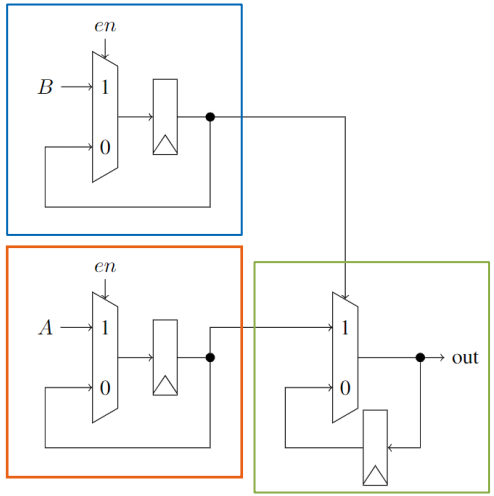
3) Clock Gating – Switching off Datapath

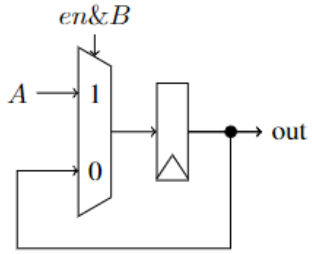
- Modify register enable signals
- Construct *observability don't care* conditions

$$TREG\big(REG(A, en), REG(B, en)\big) \rightarrow REG(A, en \,\&\, B)$$



Siemens White Paper





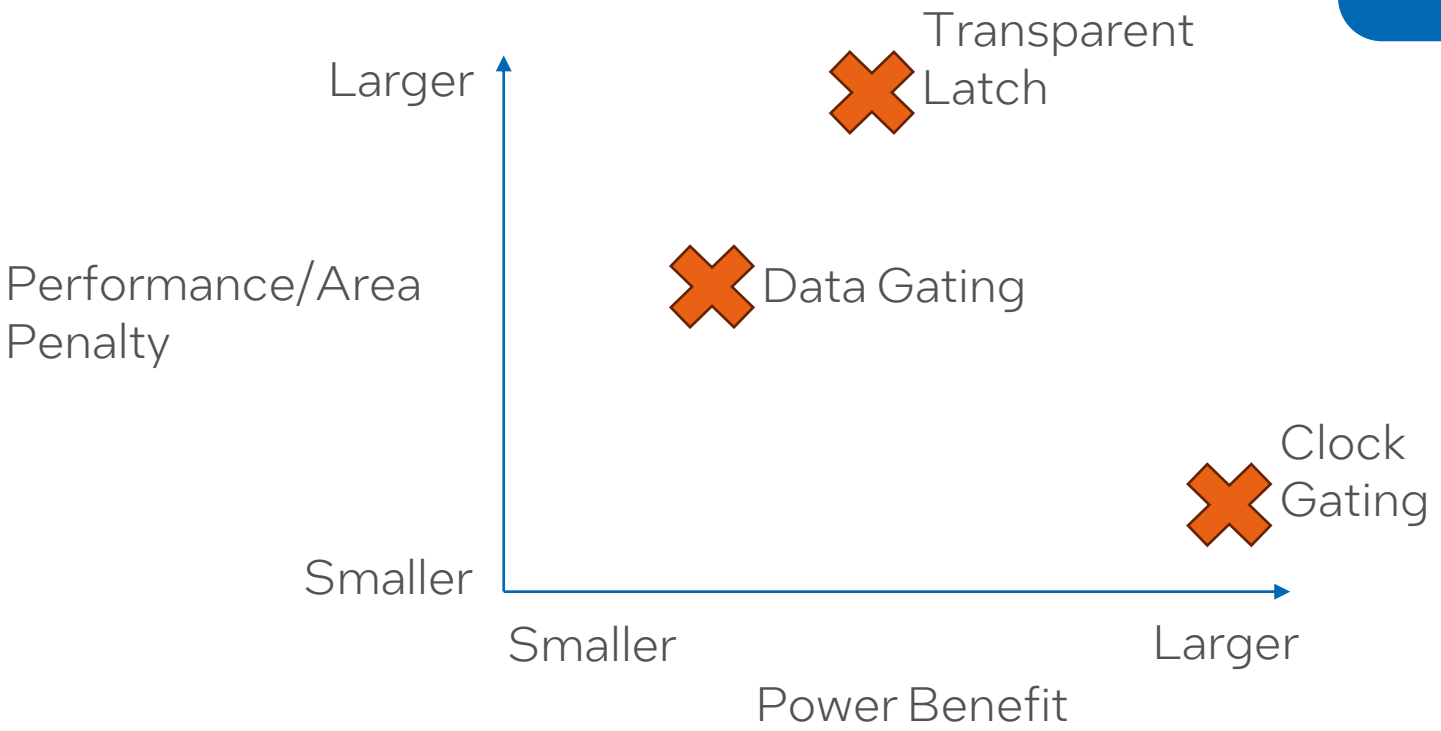
^Gated Reg^

Considerations:

- Availability of clock gating signals
- Timing impact & gate count

Evaluating the Methods

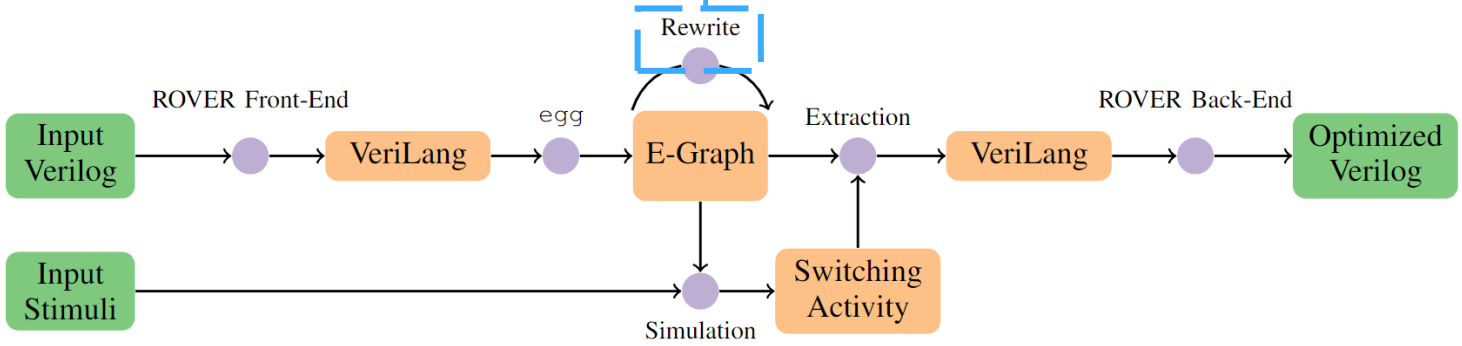
Disclaimer:
Highly dependent upon context!



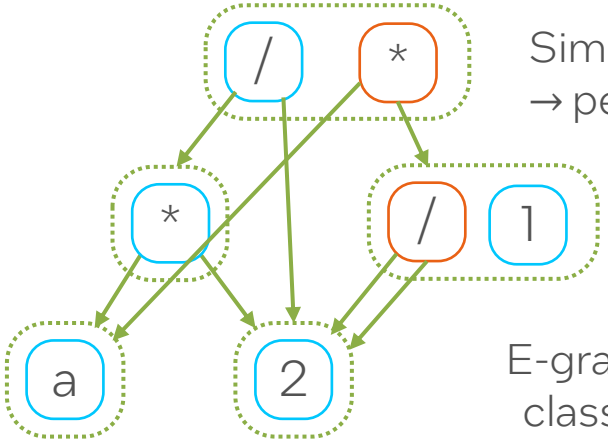
Combining Power and Arithmetic Optimization

- Arithmetic rewrites – associativity, carry-save
- Logic rewrites – mux tree rearrangement
- Logic arithmetic exchange

All rewrites applied and combined during exploration

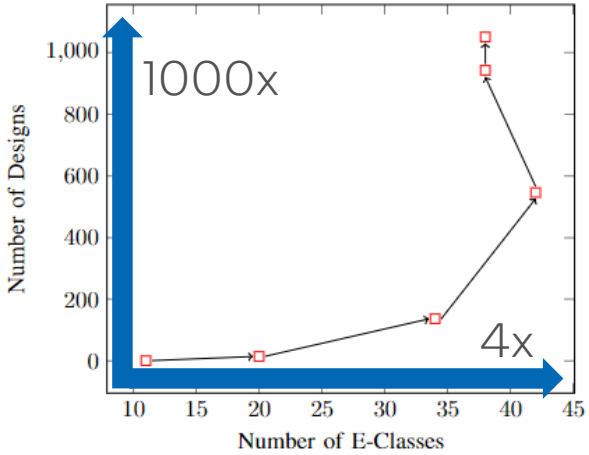


Computationally Efficient E-Graph Simulation



Simulate one node per e-class
→ per class switching activity

E-graph simulation scales with classes not implementations

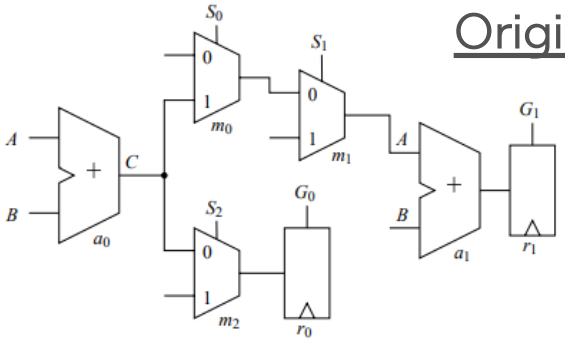


Extraction:
- Nodes in an e-class use more/less energy
- Estimate node cost based on switching activities and operator area
- Extract best using ILP

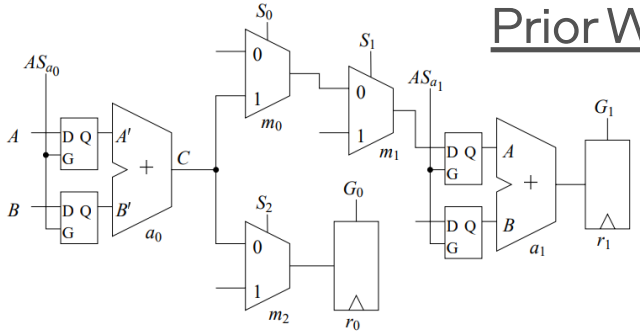
Untapped simulation/ testing opportunities?

Datapath Extraction Aware Case Study

Original



Prior Work

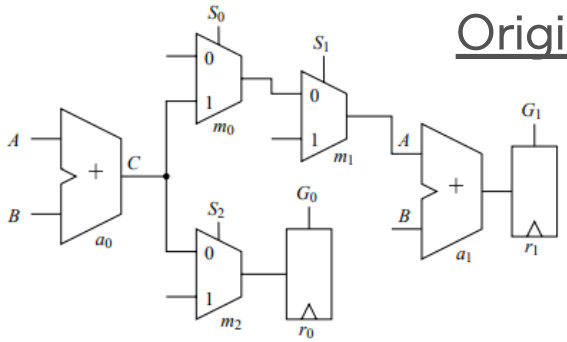


Power: -18%

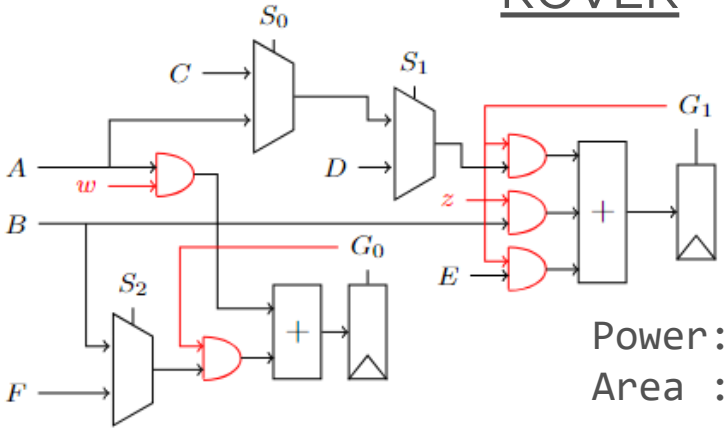
Area : +37%

Datapath Extraction Aware Case Study

Original



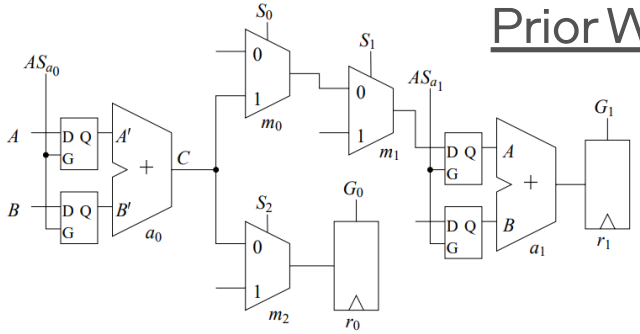
ROVER



Power: -27%
Area : +15%

$$w = \overline{S_2} G_0, z = G_1 \overline{S_1} S_0$$

Prior Work

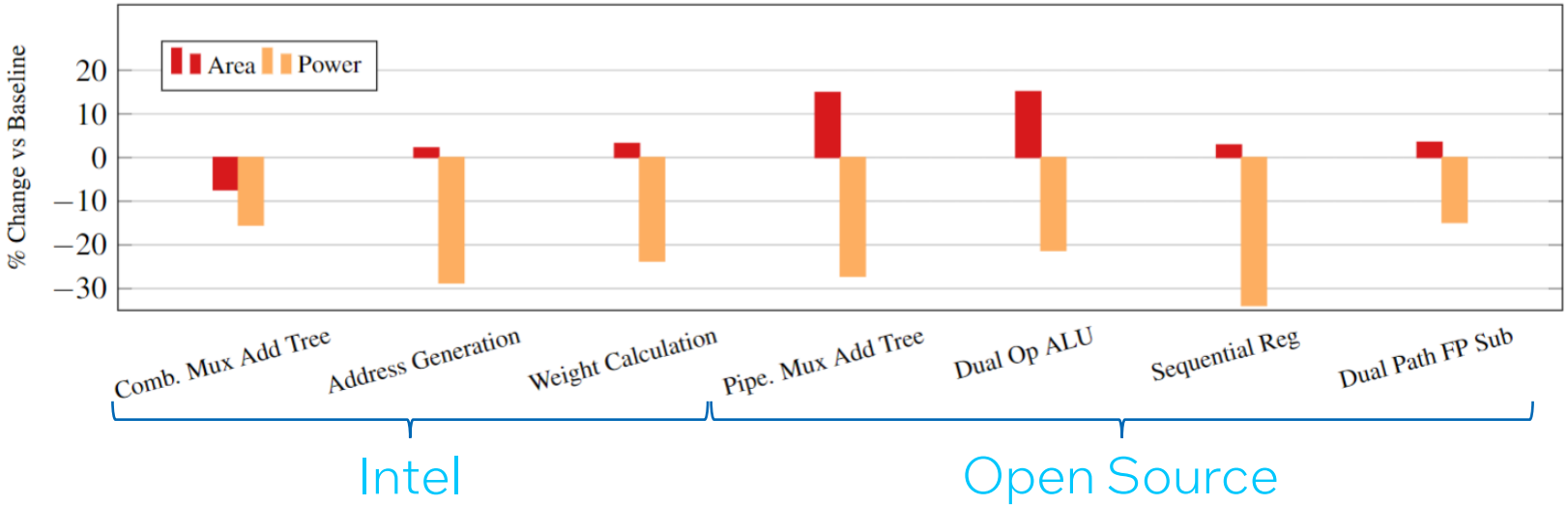


Power: -18%
Area : +37%

Push adders through muxes
Use 3 input carry-save adder

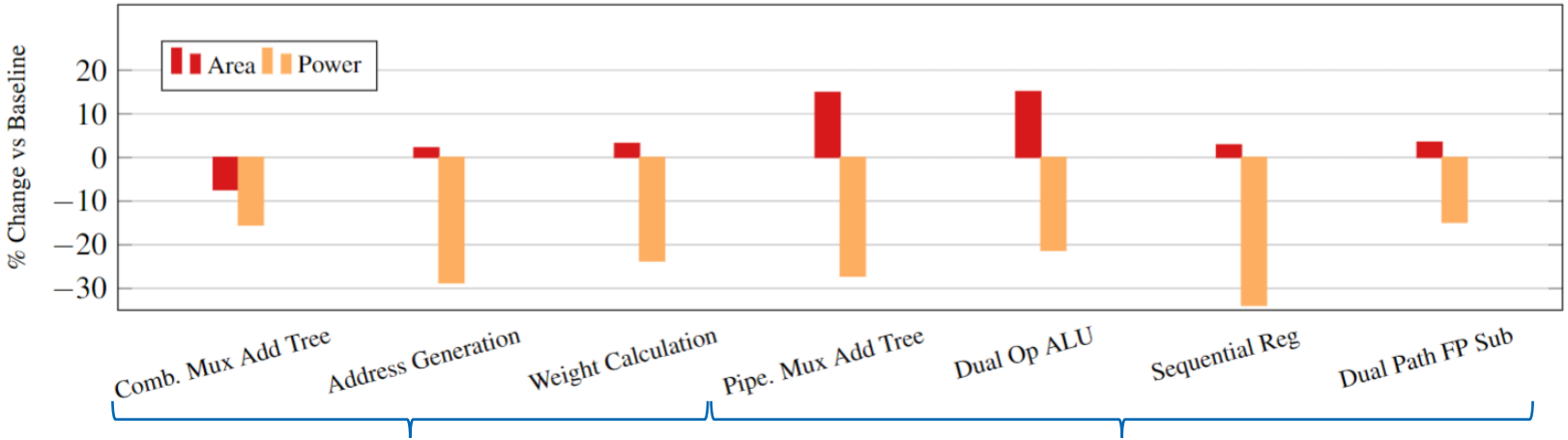
Results

Up to 34% less power, 24% on average, 5% average area overhead

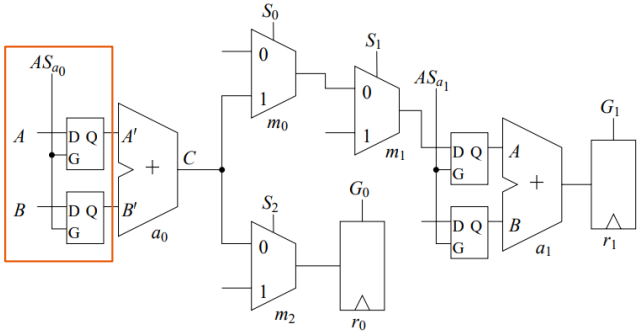


Results

Up to 34% less power, 24% on average, 5% average area overhead



Future Work



UNREACHABLE!!!

- Resource sharing –not expressible as local equivalence preserving rewrites…

Backup

Results

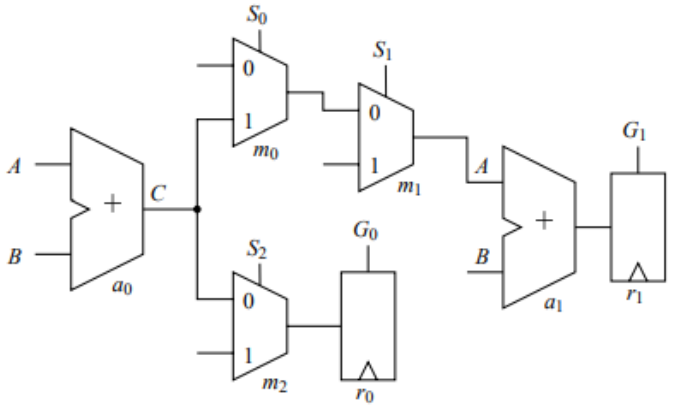
Benchmark	Nodes	Baseline Area (μm^2)	Baseline Power (μW)	Area Optimized Area (μm^2)	Area Optimized Power (μW)	Power Optimized Area (μm^2)	Power Optimized Power (μW)
Comb. Mux Add Tree	20	32.9	98.2	32.8 (- 0.4%)	98.8 (+ 0.5%)	**31.0**(- 7.4%)	**83.2** (-15.5%)
Address Generation	22	58.5	421.9	**57.1** (- 0.2%)	419.2 (-0.6%)	57.2 (+ 2.2%)	**301.2** (-28.7%)
Weight Calculation	81	51.6	1141.4	**46.4** (-10.2%)	1072.3 (-6.1%)	53.3 (+ 3.2%)	**871.5** (-23.7%)
Pipe. Mux Add Tree [9]	23	**38.6**	852.3	**38.6** (0.0%)	852.3 (0.0%)	44.1 (+14.9%)	**615.3** (-27.2%)
Dual Op ALU [10]	17	**6.5**	186.9	**6.5** (0.0%)	186.9 (0.0%)	7.5 (+15.1%)	**146.8** (-21.3%)
Sequential Reg [13]	13	**12.4**	579.6	**12.4** (0.0%)	579.6 (0.0%)	12.8 (+ 2.9%)	**383.0** (-33.9%)
Dual Path FP Sub [4]	62	**27.8**	1097.1	**27.8** (0.0%)	1097.1 (0.0%)	29.0 (+ 3.5%)	**929.4** (-14.9%)

	Muxes			Registers		Total Power (mW)	
Cfg.	S_0	S_1	S_2	G_0	G_1	Baseline	ROVER
1	0.1	0.1	0.1	0.1	0.1	1.09	0.76 (-30%)
2	0.1	0.1	0.1	0.8	0.8	1.09	0.95 (-14%)
3	0.8	0.8	0.8	0.8	0.8	1.30	1.15 (-11%)
4	0.8	0.8	0.8	0.1	0.1	1.29	1.03 (-20%)

Data Gate

Nothing

Transparent Register



Clock Gate Proof

$$L_i = \texttt{TREG}(\texttt{REG}(a_i, en_i), \texttt{REG}(b_i, en_i)) \text{ and}$$
$$R_i = \texttt{REG}(a_i, en_i \& b_i)$$

for all clock cycles i via induction. First, let

$$p_i = \texttt{REG}(a_i, en_i) \text{ and } q_i = \texttt{REG}(b_i, en_i).$$

Suppose $\forall i \leq k\ L_i = R_i$, then if $en_k = 1$:

$$\begin{aligned} q_{k+1} &= b_k & p_{k+1} &= a_k \\ L_{k+1} &= q_{k+1}\,?\,p_{k+1} &:& L_k \\ &= b_k\ ?\,a_k &:& L_k \end{aligned}$$

Then, since $en_k = 1$ and $R_k = L_k$,

$$\begin{aligned} R_{k+1} &= en_k \& b_k\,?\,a_k &:& R_k \\ &= b_k\,?\,a_k &:& R_k = L_{k+1} \end{aligned}$$

Now if $en_k = 0$, then $R_{k+1} = R_k = L_k$ and

$$\begin{aligned} q_{k+1} &= q_k & p_{k+1} &= p_k \\ L_{k+1} &= q_k\,?\,p_k &:& L_k \\ q_k = 1 \Rightarrow L_k &= q_k\,?\,p_k &:& L_{k-1} = p_k \end{aligned}$$

Therefore $L_{k+1} = L_k = R_{k+1}$ and hence $R_{k+1} = L_{k+1}$ for all values of en_k. Under the zero register initialization assumption it is trivial to prove $L_0 = R_0$.