CAL POLY
SAN LUIS OBISPO

EE 431: COMPUTER-AIDED DESIGN OF VLSI DEVICES

Adders

Nishith N. Chakraborty

November, 2024

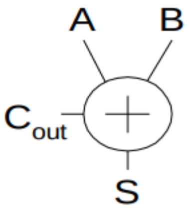
OUTLINE

- Single-bit Addition
- Carry-Ripple Adder
- Carry-Skip Adder
- Carry-Lookahead Adder
- Carry-Select Adder
- Carry-Increment Adder
- Tree Adder

SINGLE-BIT ADDITION

- Half Adder

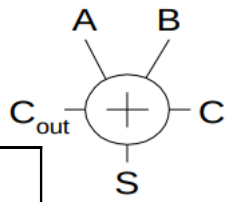
$$S = A \oplus B$$
$$C_{out} = A \cdot B$$



A	B	C_{out}	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

- Full Adder

$$S = A \oplus B \oplus C$$
$$C_{out} = MAJ(A, B, C)$$



A	B	C	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

PGK

- For a full adder, define what happens to carries
  - Generate: C_{out} = 1 independent of C
    - G =
  - Propagate: C_{out} = C
    - P =
  - Kill: C_{out} = 0 independent of C
    - K =

PGK

- For a full adder, define what happens to carries
  - Generate: C_{out} = 1 independent of C
    - G = A • B
  - Propagate: C_{out} = C
    - $P = A \oplus B$
  - Kill: C_{out} = 0 independent of C
    - K = ~A •~ B

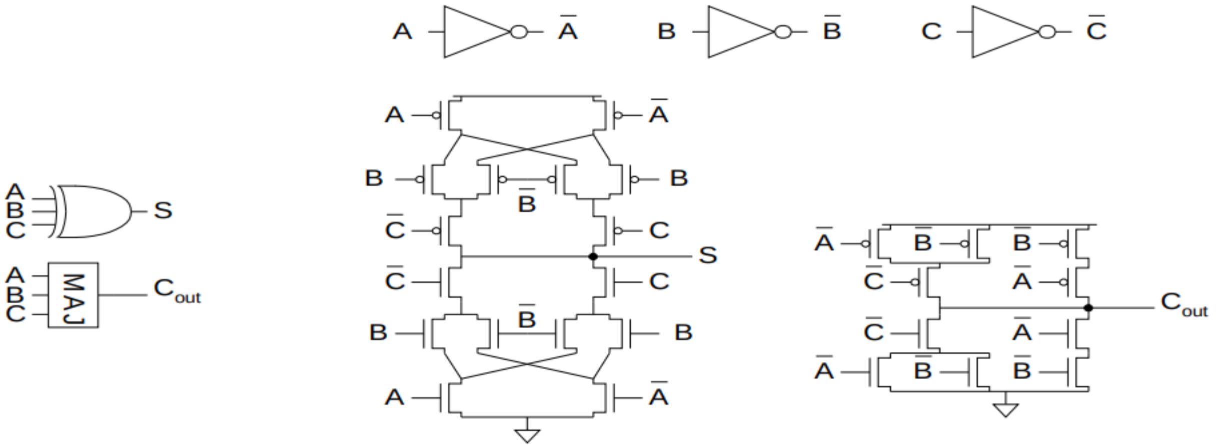
A	*B*	*C*	*G*	*P*	*K*	C_{out}	*S*
0	0	0	0	0	1	0	0
		1				0	1
0	1	0	0	1	0	0	1
		1				1	0
1	0	0	0	1	0	0	1
		1				1	0
1	1	0	1	0	0	1	0
		1				1	1

FULL ADDER DESIGN I

- Brute force implementation from equations

$$S = A \oplus B \oplus C$$

$$C_{\text{out}} = MAJ(A, B, C)$$



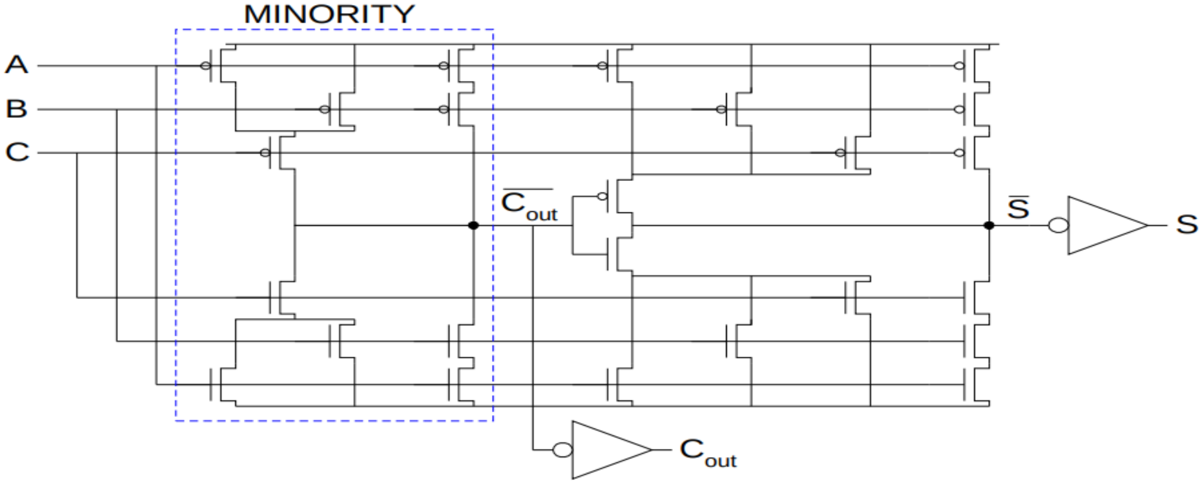
CAL POLY
SAN LUIS OBISPO

FULL ADDER DESIGN II

- Factor S in terms of C_{out}

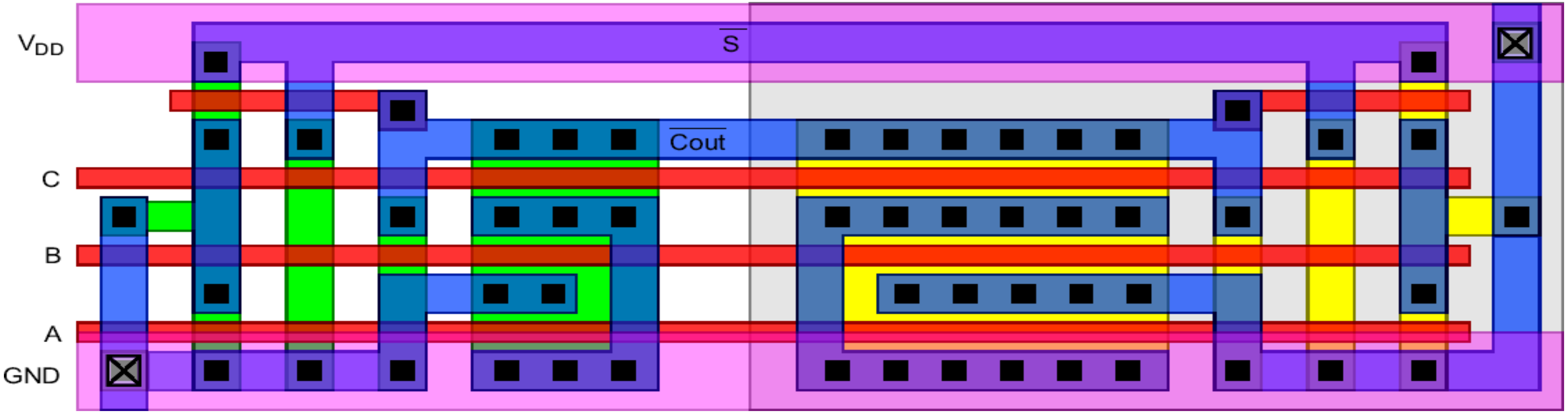
$$S = ABC + (A + B + C)(\sim C_{out})$$

- Critical path is usually C to C_{out} in ripple adder



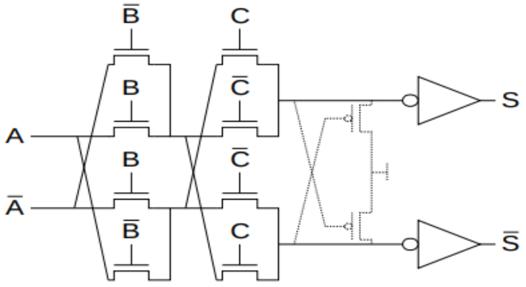
LAYOUT

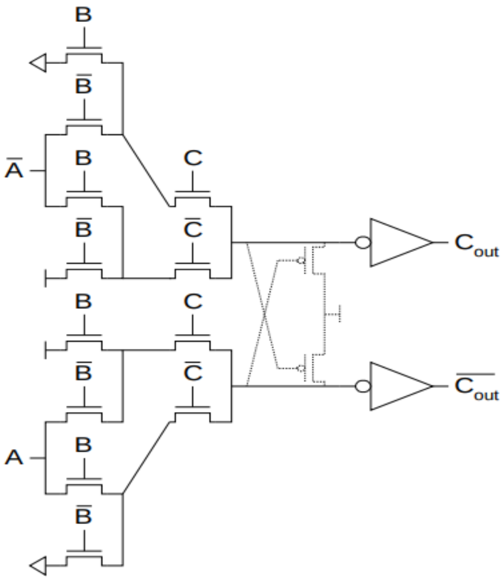
- Clever layout circumvents usual line of diffusion
  - Use wide transistors on critical path
  - Eliminate output inverters



FULL ADDER DESIGN III

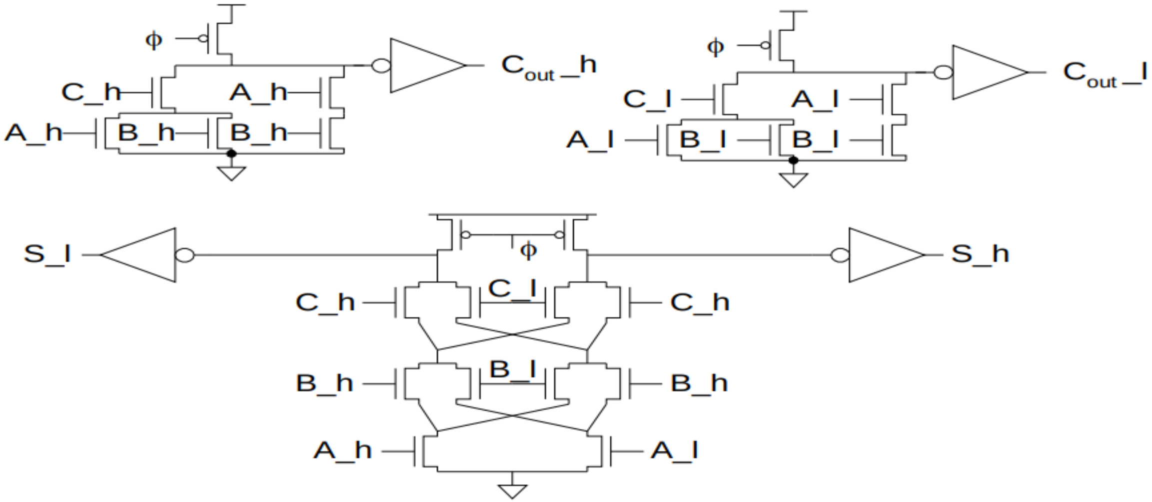
- Complementary Pass Transistor Logic (CPL)
- Slightly faster, but more area than II





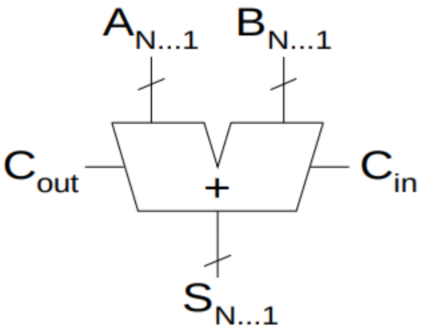
FULL ADDER DESIGN IV

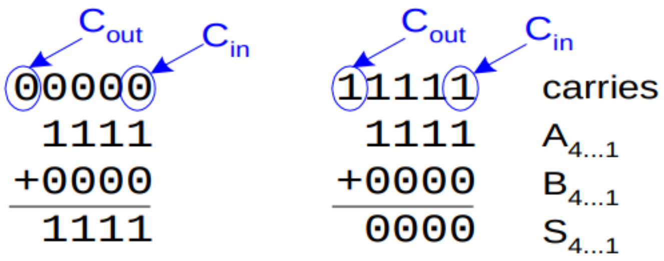
- Dual-rail domino
  - Very fast, but large and power hungry
  - Used in very fast multipliers



CARRY PROPAGATE ADDERS

- N-bit adder called CPA
  - Each sum bit depends on all previous carries
  - How do we compute all these carries quickly?

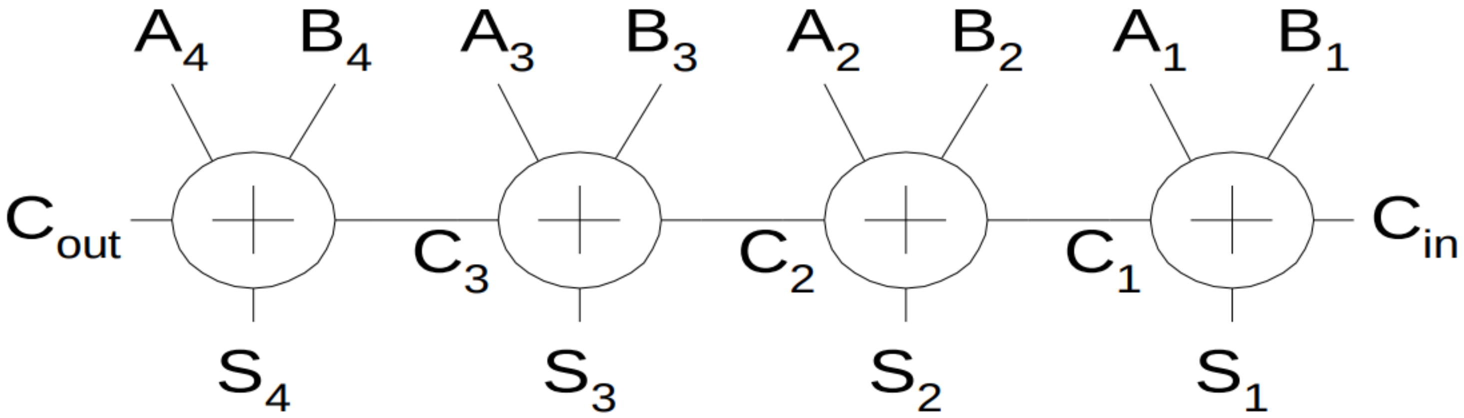




CAL POLY
SAN LUIS OBISPO

CARRY-RIPPLE ADDER

- Simplest design: cascade full adders
  - Critical path goes from C_{in} to C_{out}
  - Design full adder to have fast carry delay



GENERATE/PROPAGATE

- Equations often factored into G and P
- Generate and propagate for groups spanning i:j

$$G_{i:j} =$$

$$P_{i:j} =$$

- Base case

$$G_{i:i} \equiv G_i =$$

$$P_{i:i} \equiv P_i =$$

$$G_{0:0} \equiv G_0 =$$

$$P_{0:0} \equiv P_0 =$$

- Sum:

$$S_i =$$

GENERATE/PROPAGATE

- Equations often factored into G and P
- Generate and propagate for groups spanning i:j, where $i \geq k > j$

$$G_{i:j} = G_{i:k} + P_{i:k} \cdot G_{k-1:j}$$

$$P_{i:j} = P_{i:k} \cdot P_{k-1:j}$$

- Base case

$$G_{i:i} \equiv G_i = A_i \cdot B_i$$

$$P_{i:i} \equiv P_i = A_i \oplus B_i$$

$$G_{0:0} \equiv G_0 = C_{in}$$

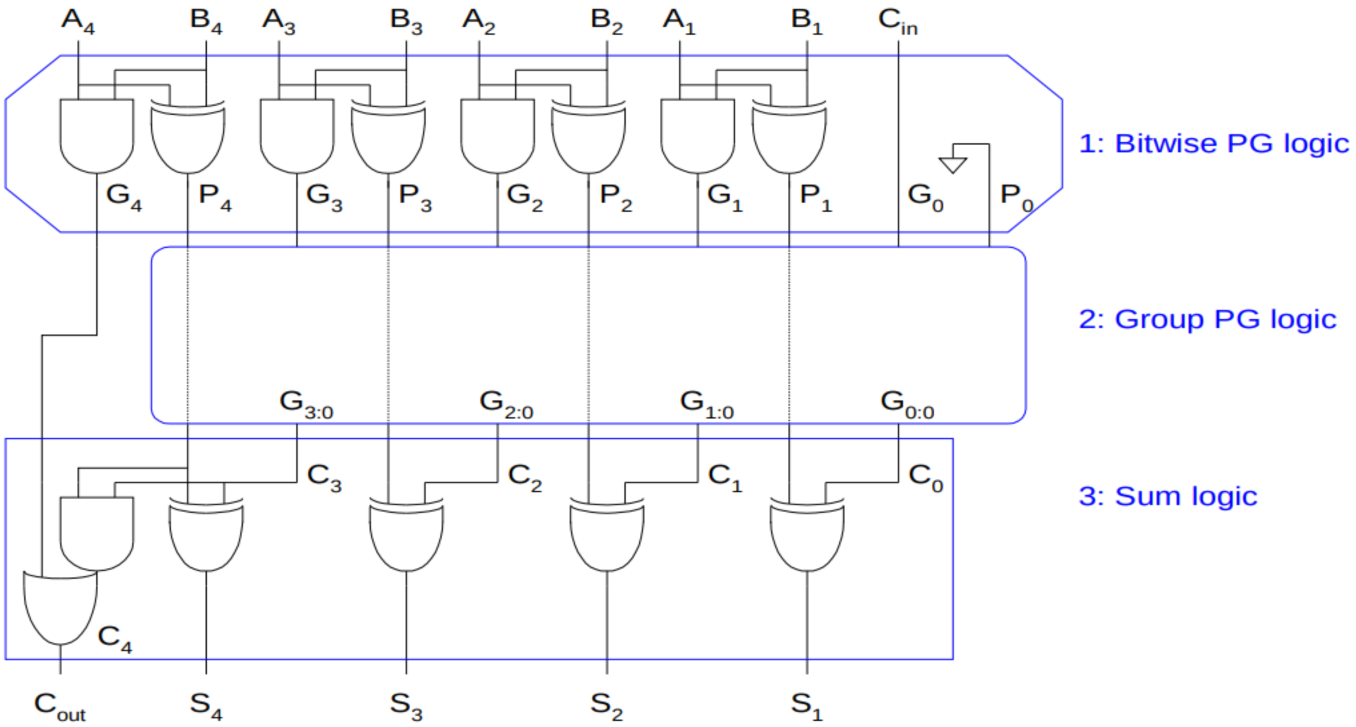
$$P_{0:0} \equiv P_0 = 0$$

- Sum:

$$S_i = P_i \oplus G_{i-1:0}$$

CAL POLY
SAN LUIS OBISPO

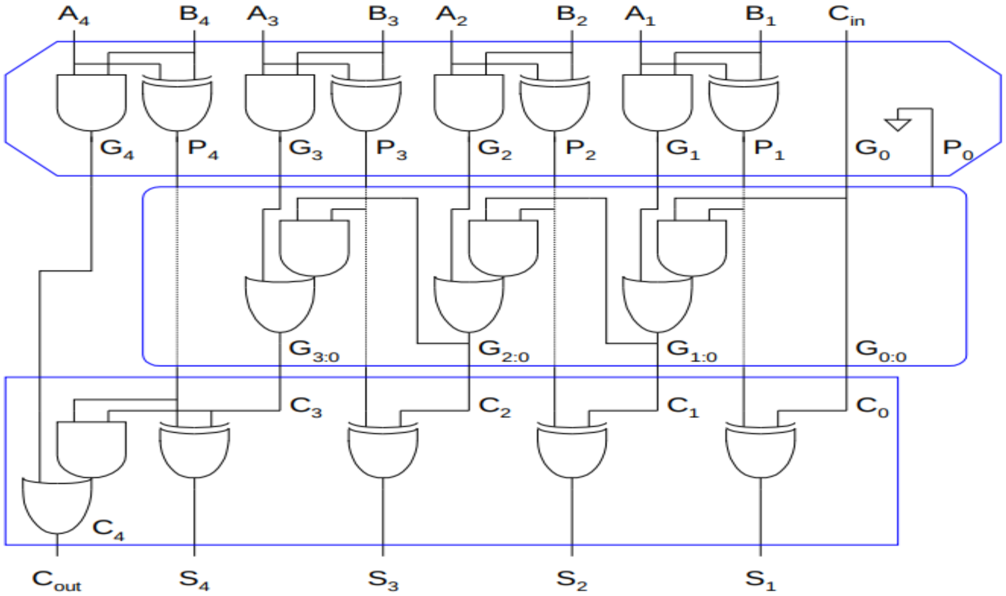
PG LOGIC



CAL POLY
SAN LUIS OBISPO

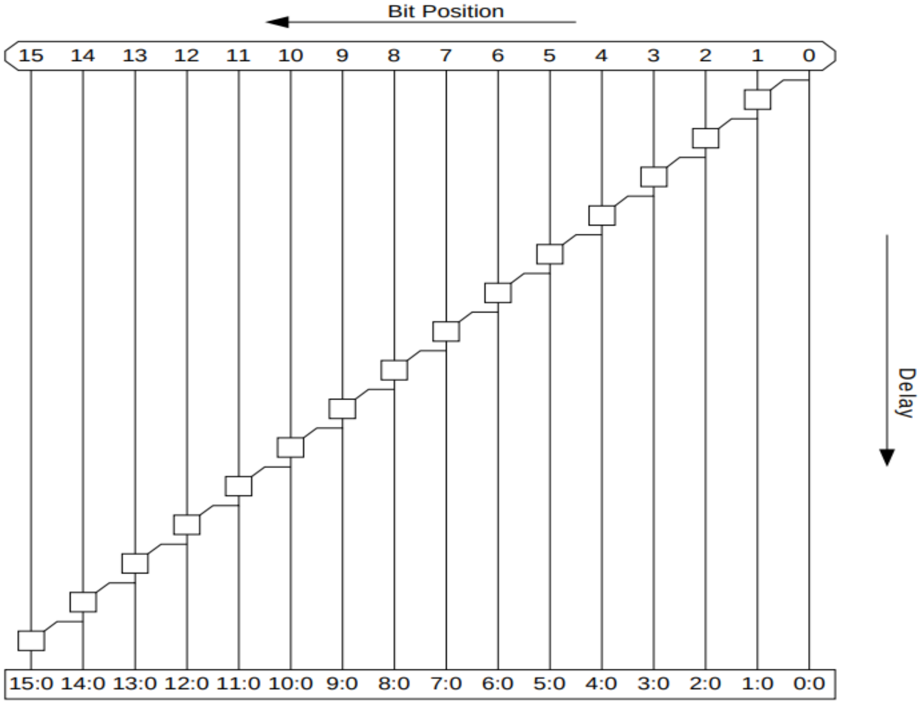
CARRY-RIPPLE REVISITED

$$G_{i:0} = G_i + P_i \cdot G_{i-1:0}$$



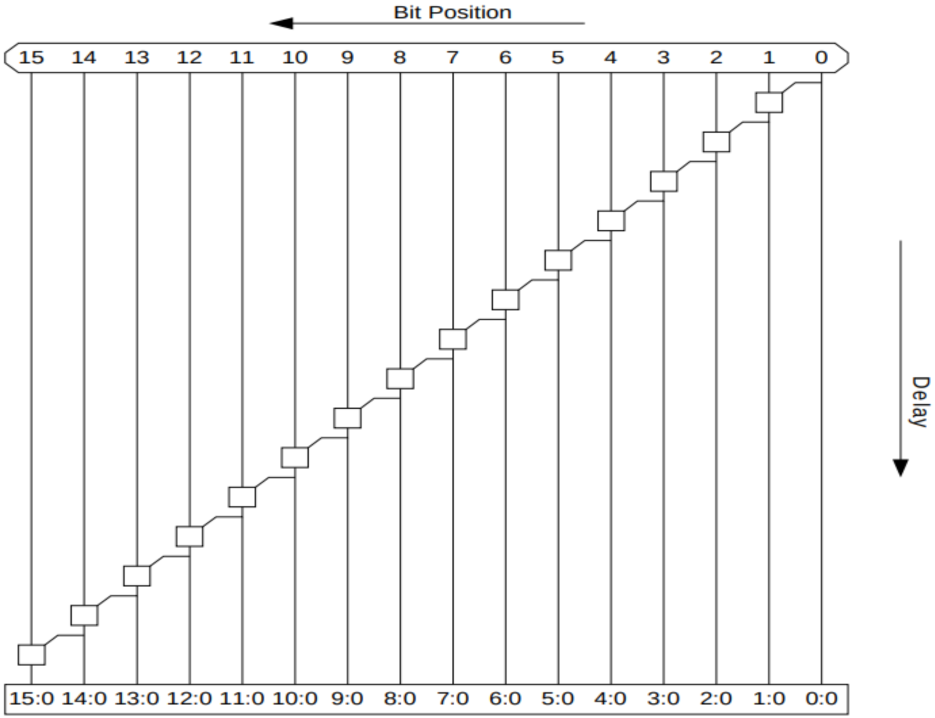
CARRY-RIPPLE PG DIAGRAM

$t_{\text{ripple}} =$



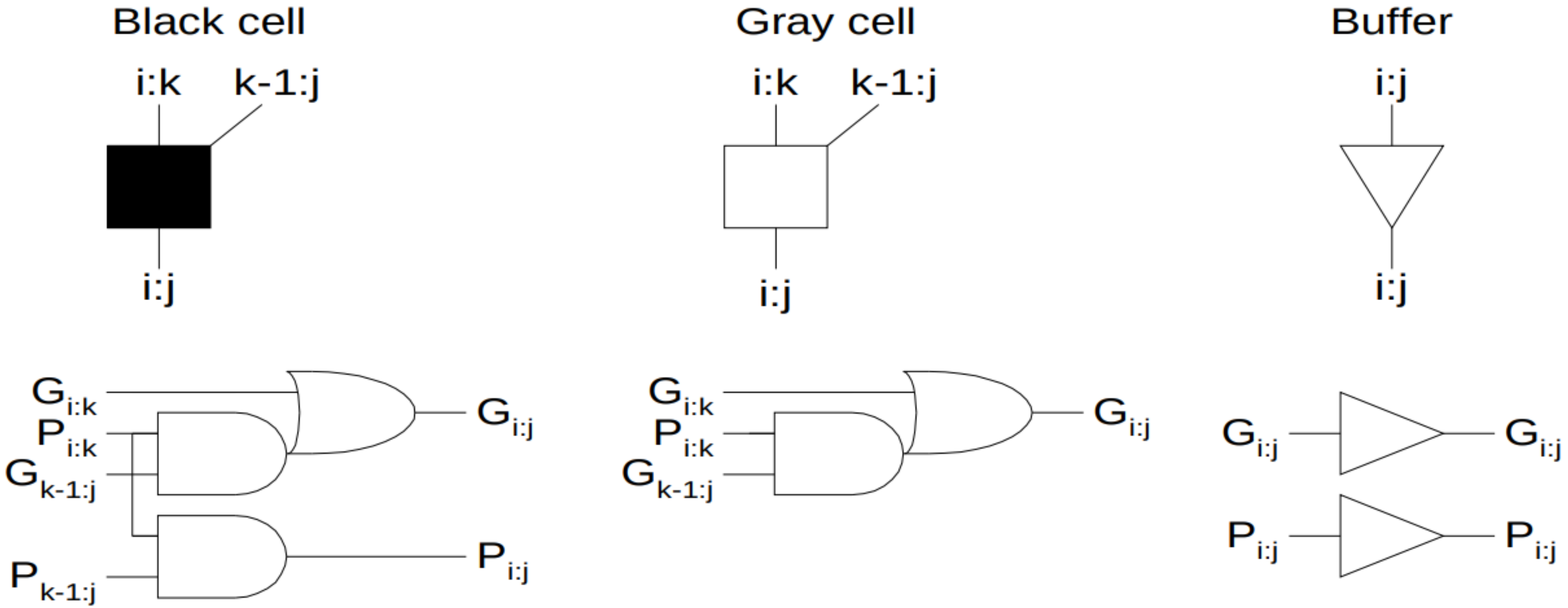
CARRY-RIPPLE PG DIAGRAM

$$t_{\text{ripple}} = t_{pg} + (N - 1)t_{AO} + t_{\text{xor}}$$



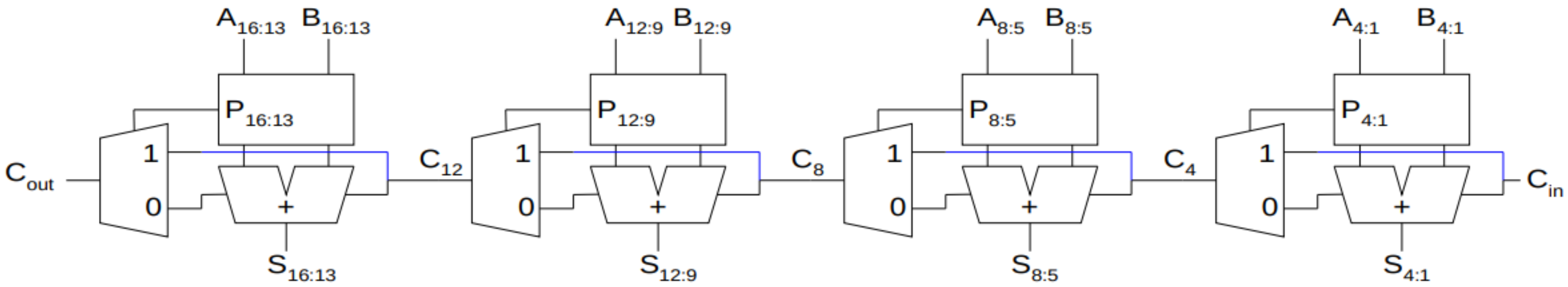
CAL POLY
SAN LUIS OBISPO

PG DIAGRAM NOTATION

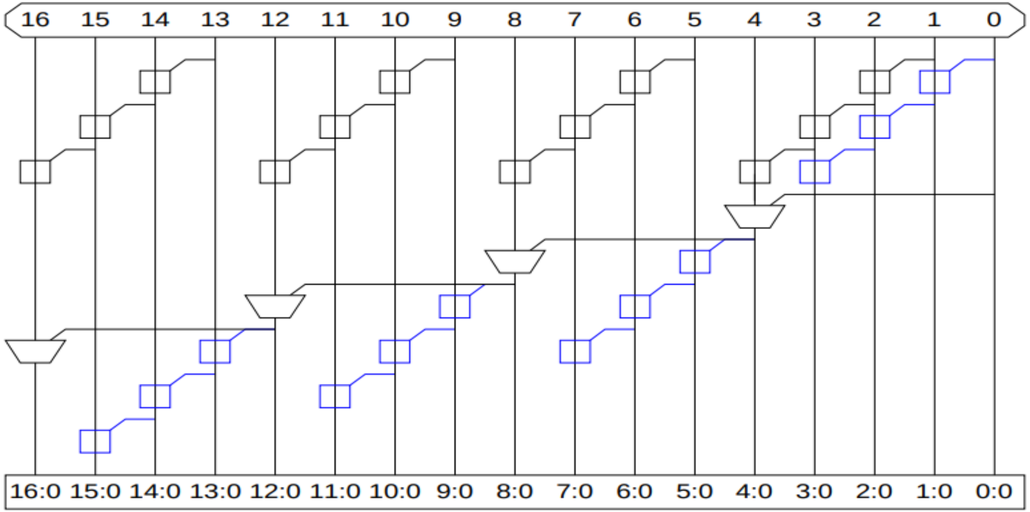


CARRY-SKIP ADDER

- Carry-ripple is slow through all N stages
- Carry-skip allows carry to skip over groups of n bits
  - Decision based on n-bit propagate signal



CARRY-SKIP PG DIAGRAM

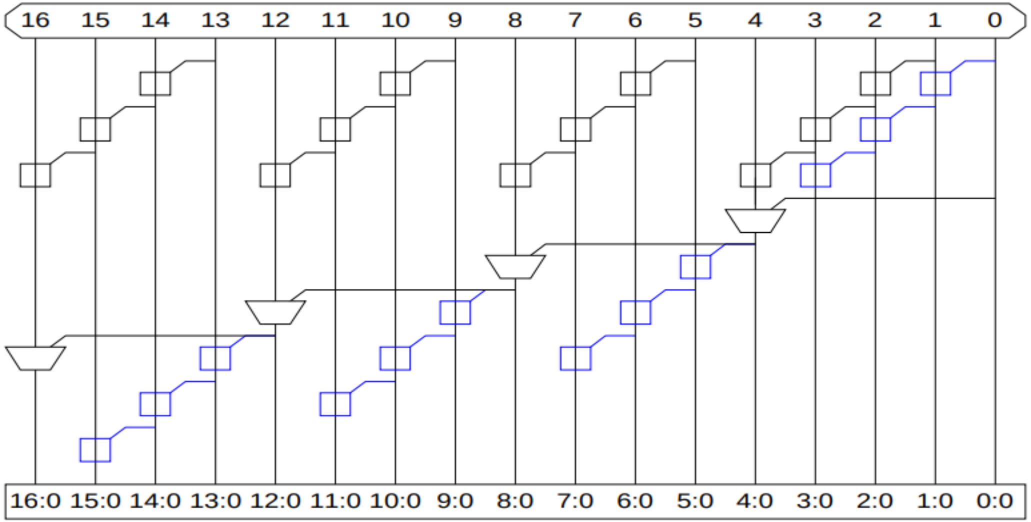


For k n-bit groups (N = nk)

$$t_{\text{skip}} =$$

CAL POLY
SAN LUIS OBISPO

CARRY-SKIP PG DIAGRAM

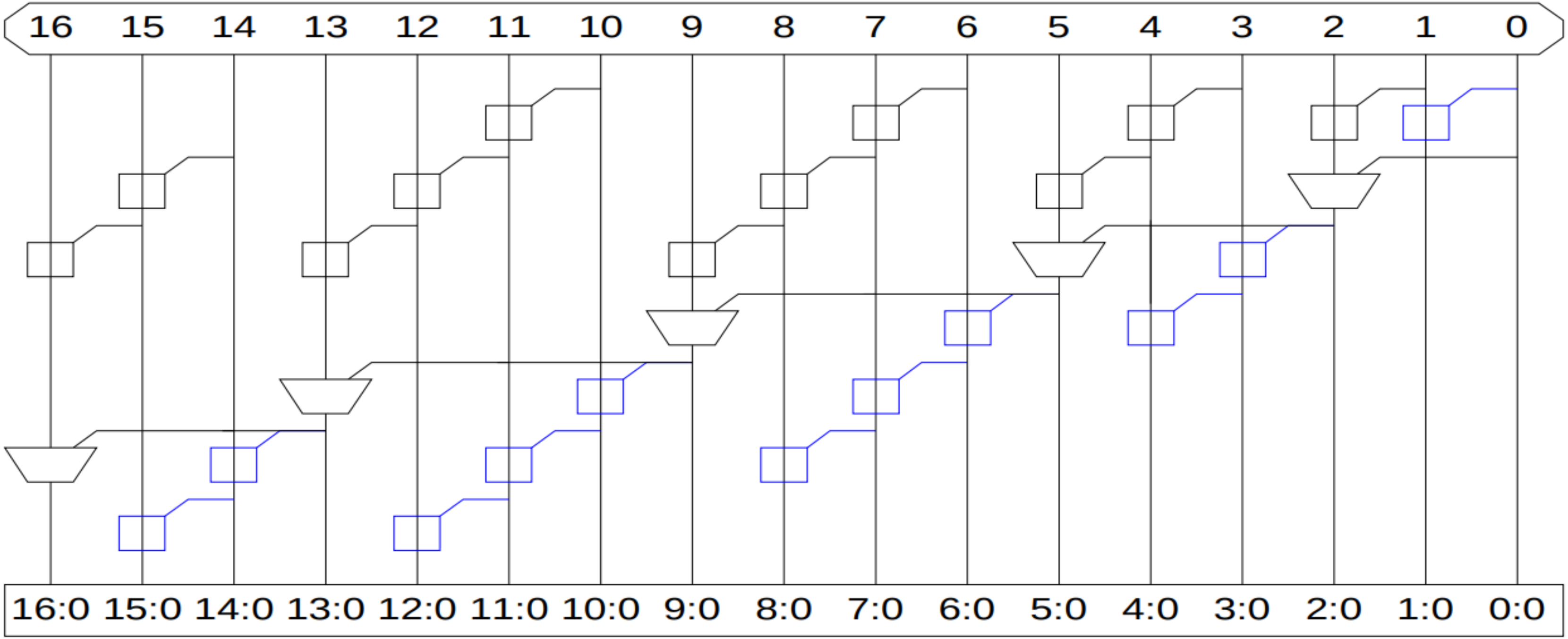


For k n-bit groups (N = nk)

$$t_{\text{skip}} = t_{pg} + 2(n-1)t_{AO} + (k-1)t_{\text{mux}} + t_{\text{xor}}$$

CAL POLY
SAN LUIS OBISPO

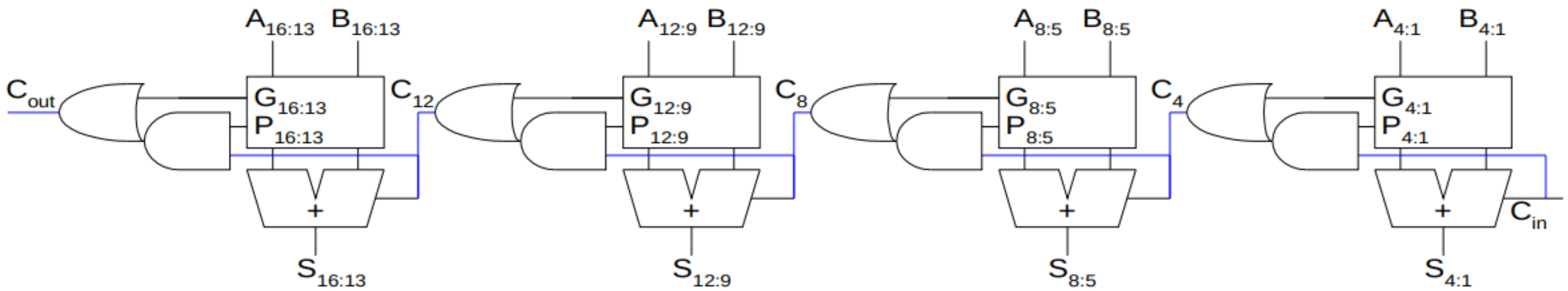
VARIABLE GROUP SIZE



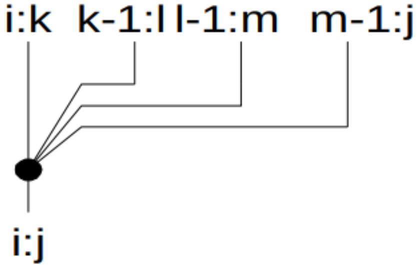
Delay grows as O(sqrt(N))

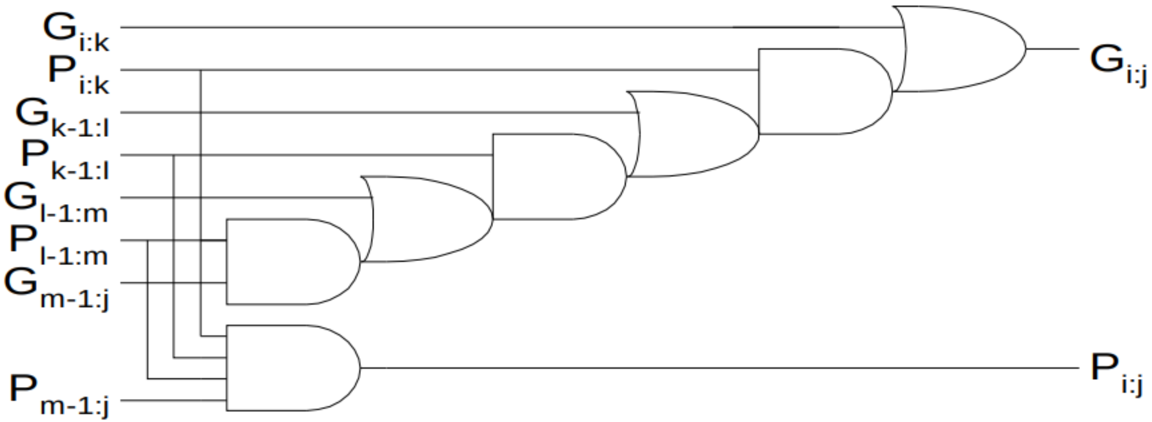
CARRY-LOOKAHEAD ADDER

- Carry-lookahead adder computes $G_{i:0}$ for many bits in parallel.
- Uses higher-valency cells with more than two inputs.



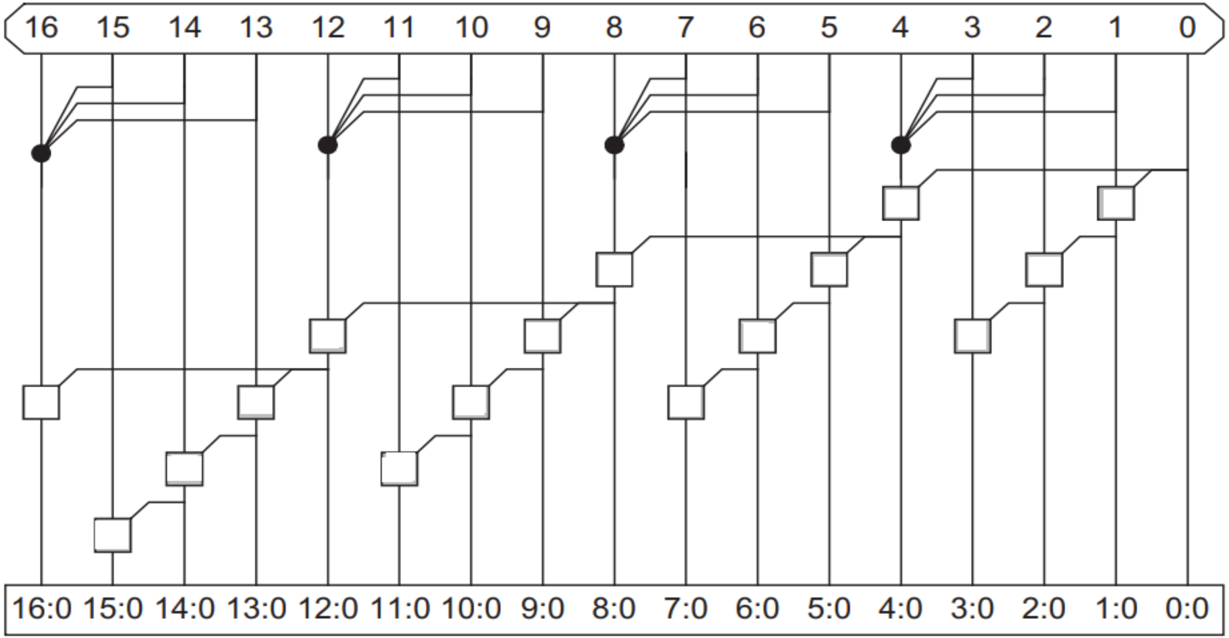
HIGHER-VALENCY CELLS





CAL POLY
SAN LUIS OBISPO

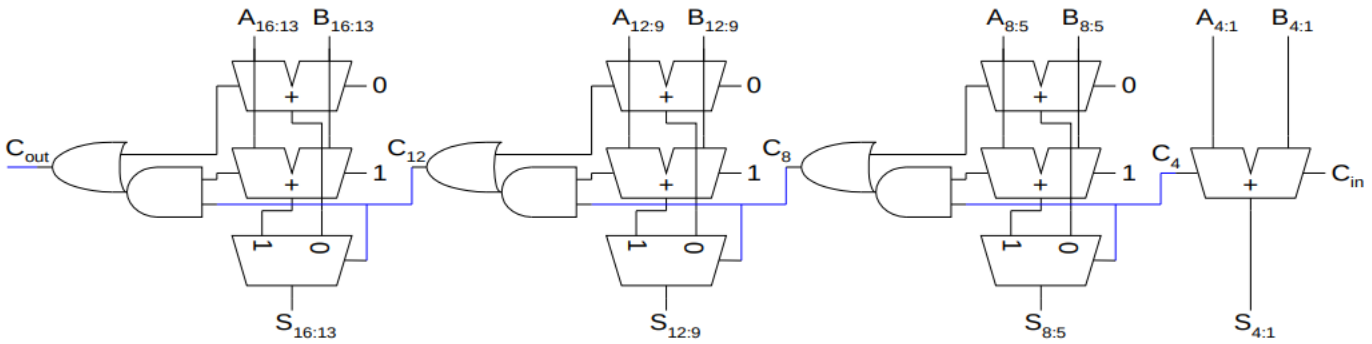
CLA PG DIAGRAM



$$t_{\mathrm{cla}} = t_{pg} + t_{pg(n)} + \left[\left(n-1\right)+\left(k-1\right)\right] t_{AO} + t_{\mathrm{xor}}$$

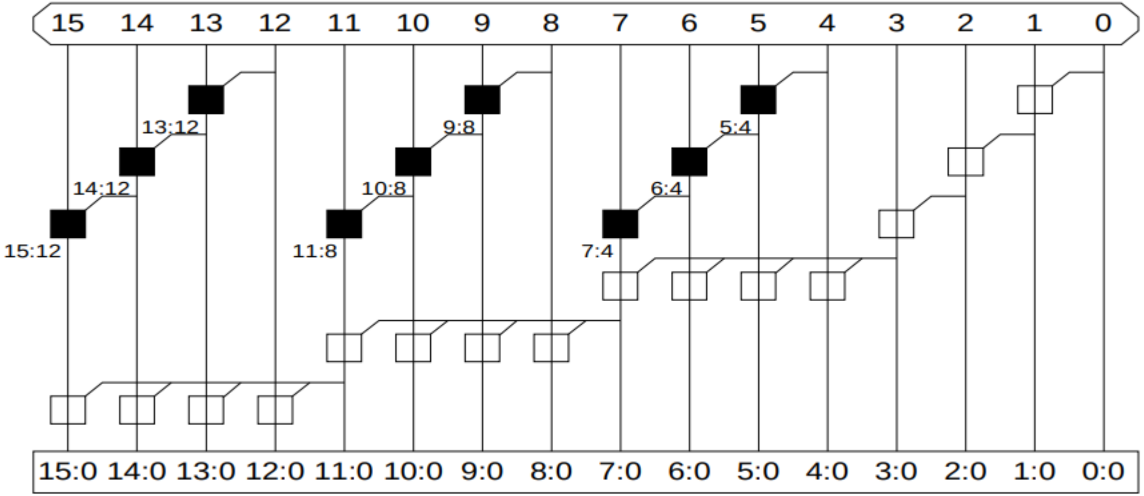
CARRY-SELECT ADDER

- Trick for critical paths dependent on late input X
  - Precompute two possible outputs for X = 0, 1
  - Select proper output when X arrives
- Carry-select adder precomputes n-bit sums
  - For both possible carries into n-bit group



CARRY-INCREMENT ADDER

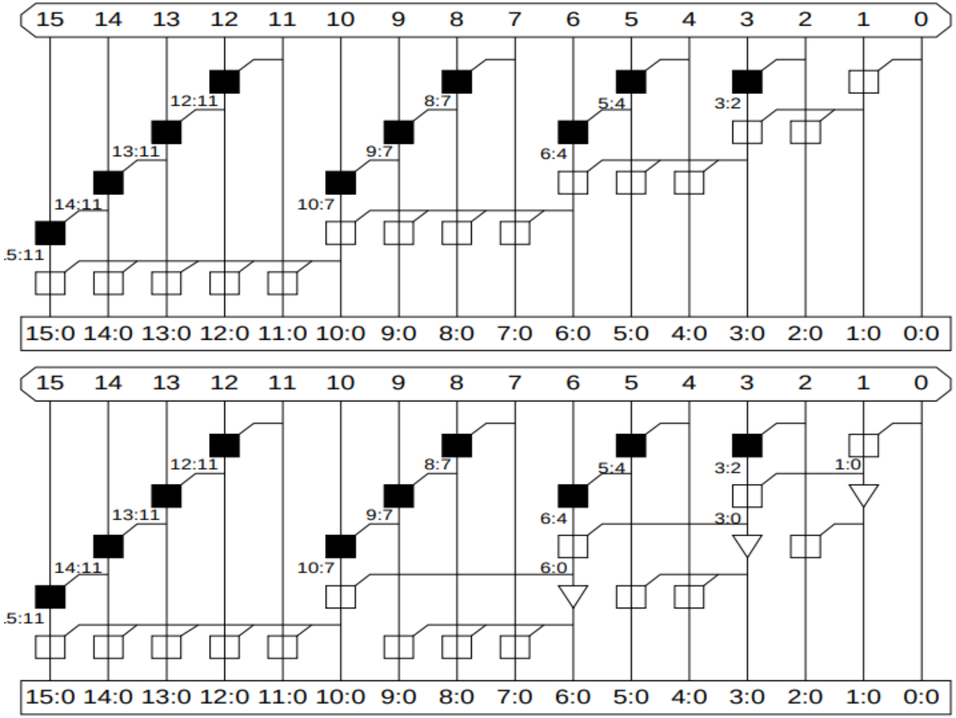
- Factor initial PG and final XOR out of carry-select



$$t_{\text{increment}} = t_{pg} + \left[(n-1) + (k-1) \right] t_{AO} + t_{\text{xor}}$$

VARIABLE GROUP SIZE

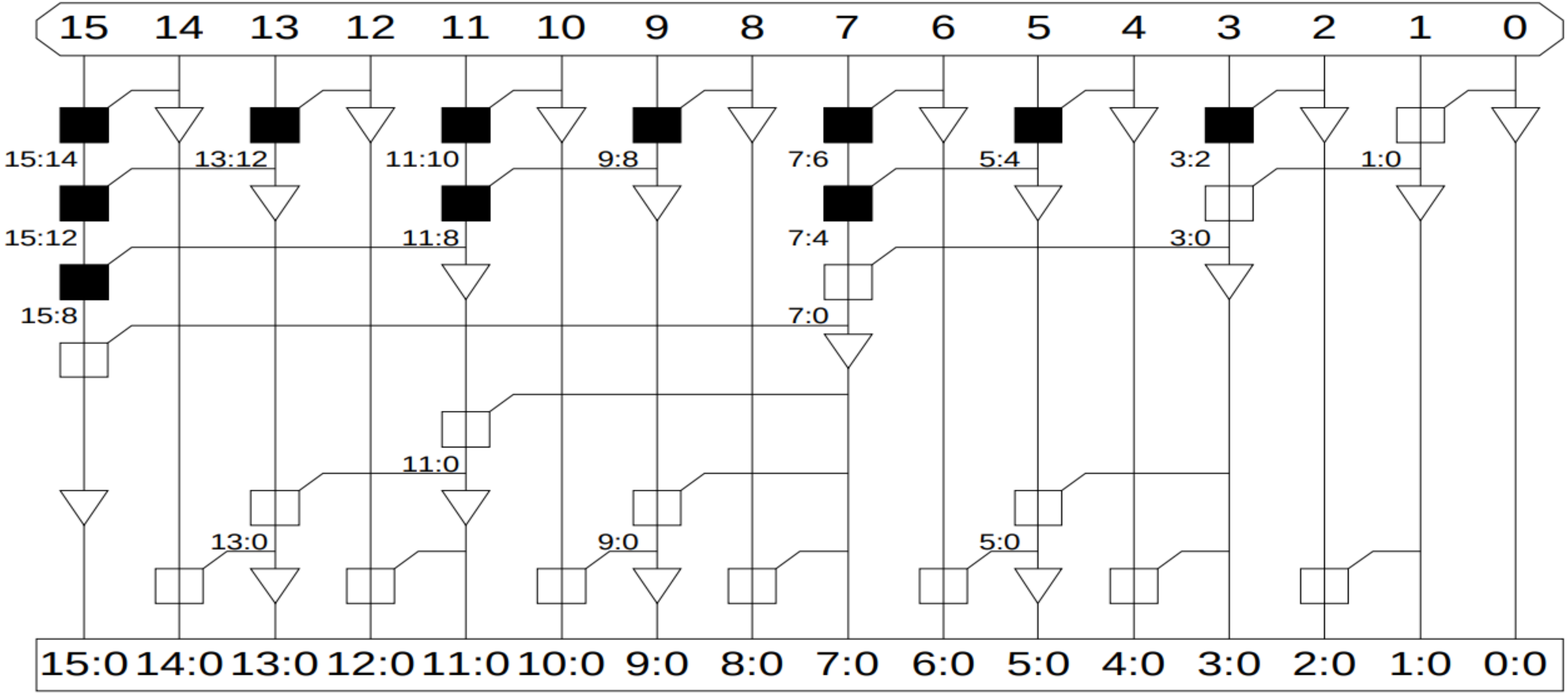
- Also buffer noncritical signals



LOGARITHMIC OR TREE ADDER

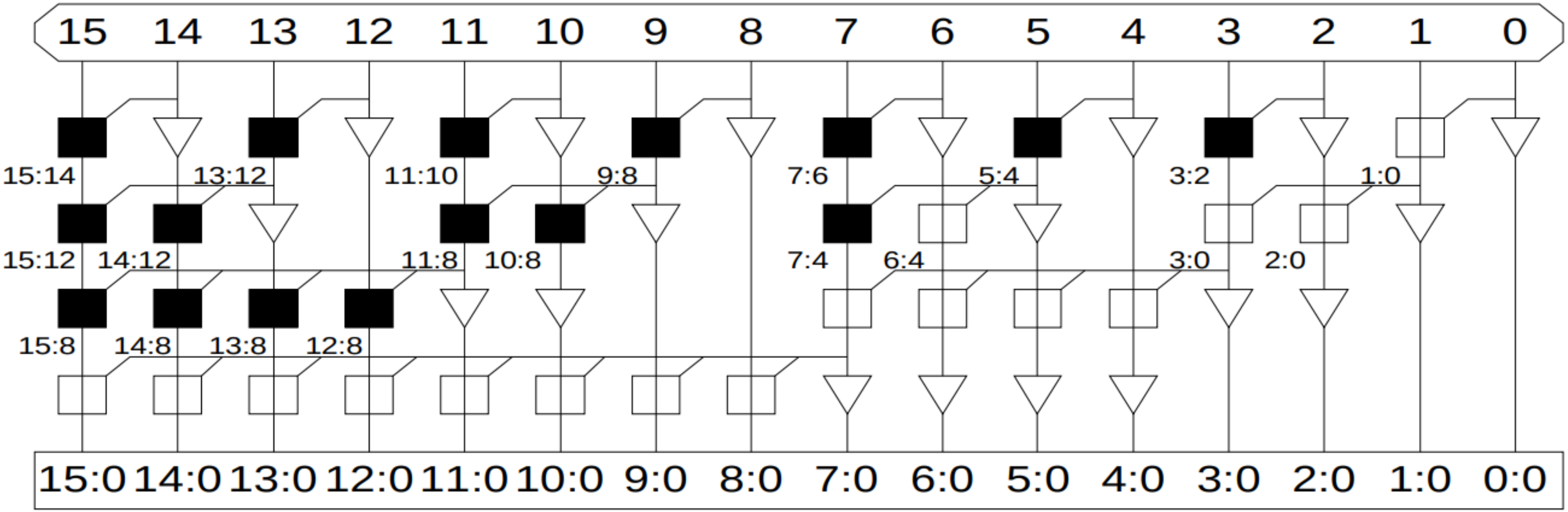
- If lookahead is good, lookahead across lookahead!
  - Recursive lookahead gives O(log N) delay
- Many variations on logarithmic adders

BRENT-KUNG

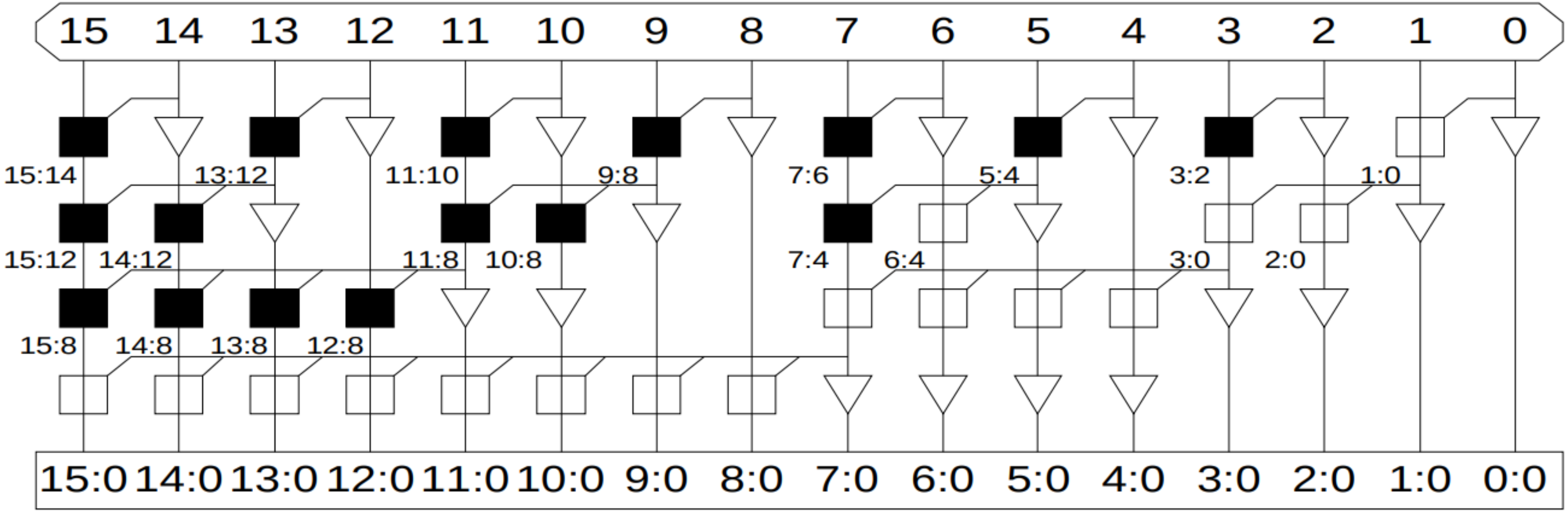


15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
15:14
13:12
11:10
9:8
7:6
5:4
3:2
1:0
15:12
11:8
7:4
3:0
15:8
7:0
11:0
13:0
9:0
5:0
15:0 14:0 13:0 12:0 11:0 10:0 9:0 8:0 7:0 6:0 5:0 4:0 3:0 2:0 1:0 0:0

SKLANSKY


15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
15:14
13:12
11:10
9:8
7:6
5:4
3:2
1:0
15:12
14:12
11:8
10:8
7:4
6:4
3:0
2:0
15:8
14:8
13:8
12:8
15:0 14:0 13:0 12:0 11:0 10:0 9:0 8:0 7:0 6:0 5:0 4:0 3:0 2:0 1:0 0:0

KOGGE-STONE



15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
15:14
13:12
11:10
9:8
7:6
5:4
3:2
1:0
15:12
14:12
11:8
10:8
7:4
6:4
3:0
2:0
15:8
14:8
13:8
12:8
15:0 14:0 13:0 12:0 11:0 10:0 9:0 8:0 7:0 6:0 5:0 4:0 3:0 2:0 1:0 0:0

Cal Poly
San Luis Obispo

Thank you!