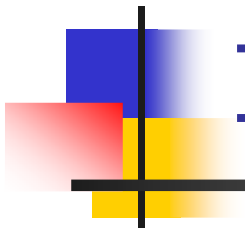
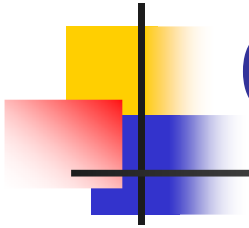


Timing-Power Optimization for Mixed-Radix Ling Adders by Integer Linear Programming

Yi Zhu, Jianhua Liu, Haikun Zhu and Chung-Kuan Cheng

Department of Computer Science & Engineering, University of California, San Diego



Outline

- Prefix Adder Problem
  - Background & Previous Work
  - Extensions: High-radix, Ling
- Our Work
  - Area/Timing/Power Models
  - ILP Formulation
- Experimental Results
- Future Work

Binary Addition

- **Input:** two *n*-bit binary numbers $a_{n-1}\ldots a_1 a_0$ and $b_{n-1}\ldots b_1 b_0$, one bit carry-in c_0
- **Output:** n-bit sum $s_{n-1}\ldots s_1 s_0$ and one bit carry out c_n
- Prefix Addition: Carry generation & propagation

$$\text{Generate}: \quad g_i = a_i b_i$$

$$\text{Propagate}: \quad p_i = a_i \oplus b_i$$

$$c_{i+1} = g_i + p_i \cdot c_i$$

$$s_i = c_i \oplus (a_i \oplus b_i)$$

Prefix Addition – Formulation

Pre-processing:

$$g_i = a_i b_i \qquad p_i = a_i \oplus b_i$$

Prefix Computation:

$$G_{[i:k]} = G_{[i:j]} + P_{[i:j]} G_{[j-1:k]}$$

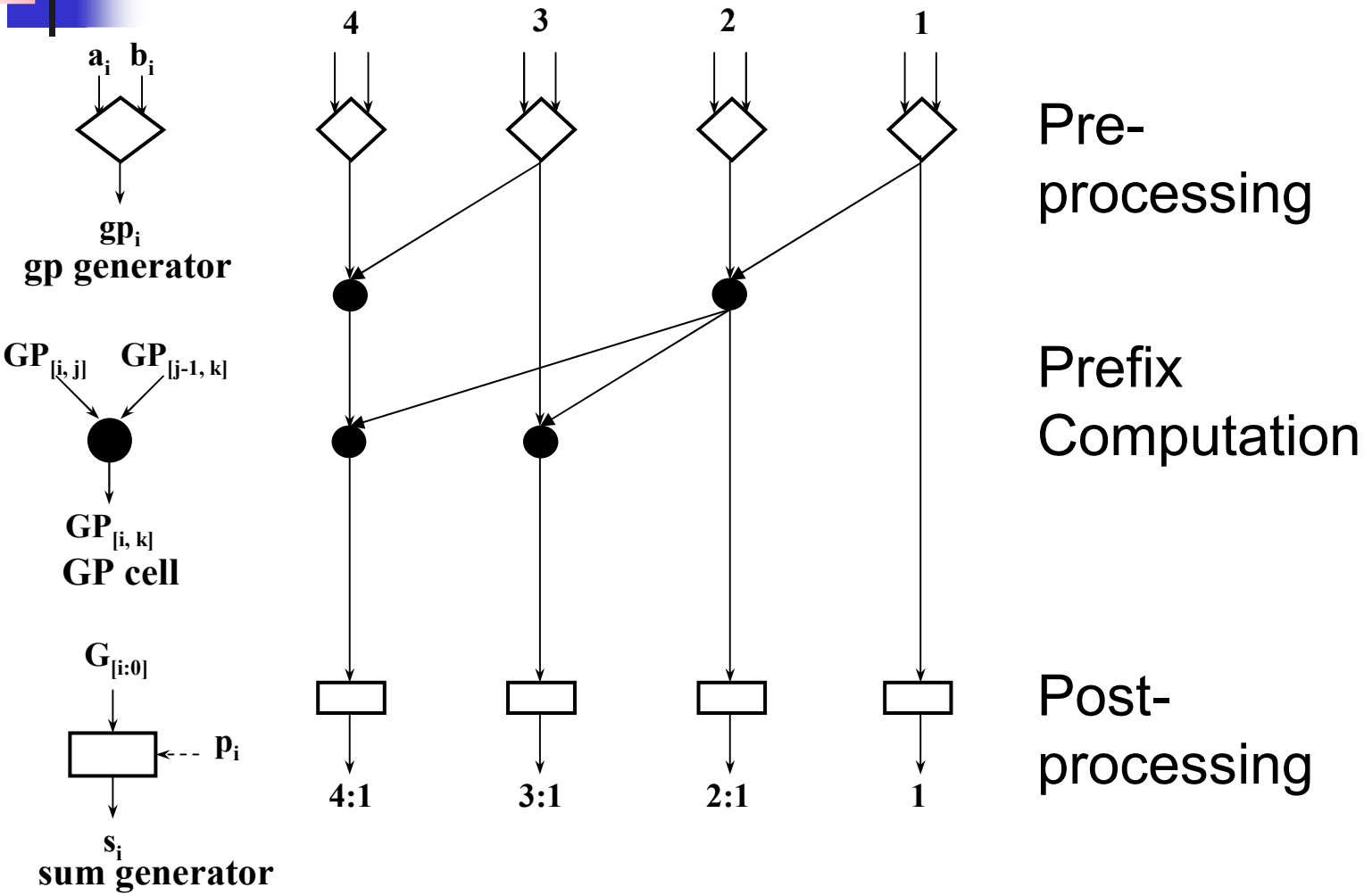
$$P_{[i:k]} = P_{[i:j]} P_{[j-1:k]}$$

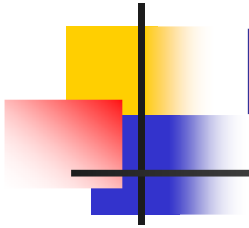
Post-processing:

$$c_{i+1} = G_{[i:0]} + P_{[i:0]} \cdot c_0$$

$$s_i = p_i \oplus c_i$$

Prefix Adder – Prefix Structure Graph

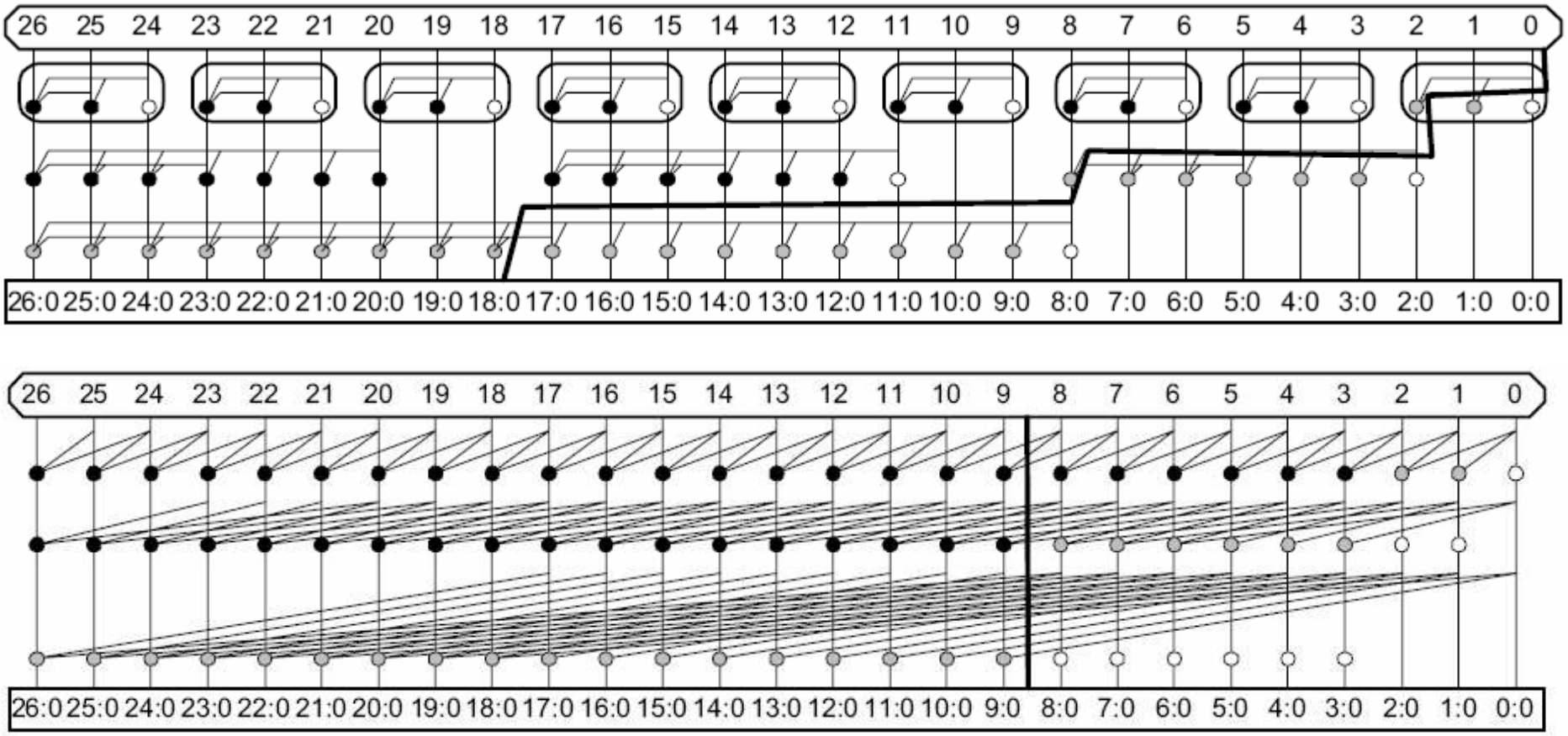




High-Radix Adders

- Each cell has more than two fan-in's
- Pros: less logic levels
  - 6 levels (radix-2) vs. 3 levels (radix-4) for 64-bit addition
- Cons: larger delay and power in each cell

Radix-3 Sklansky & Kogge-Stone Adder

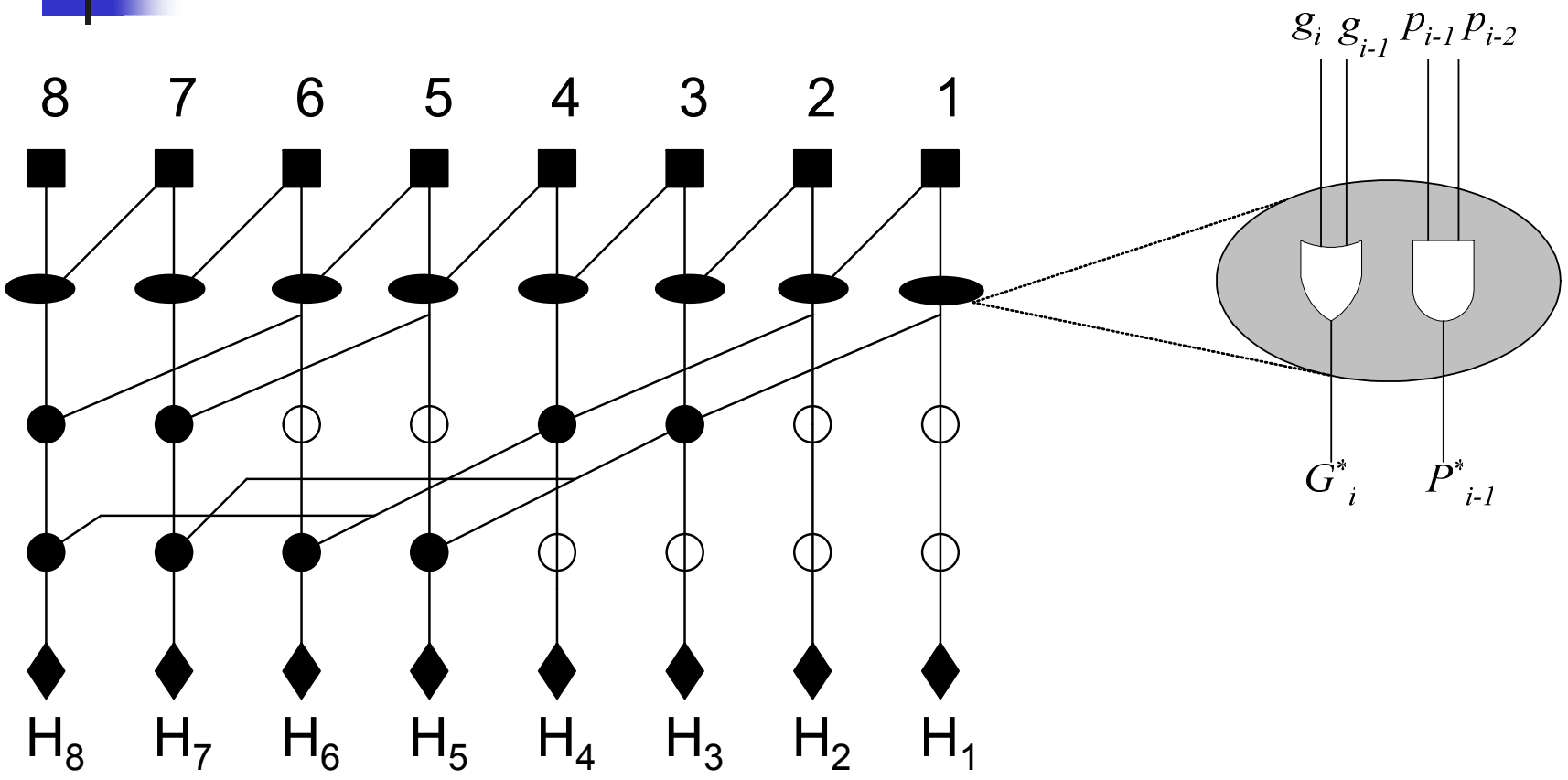


David Harris, "Logical Effort of Higher Valency Adders"

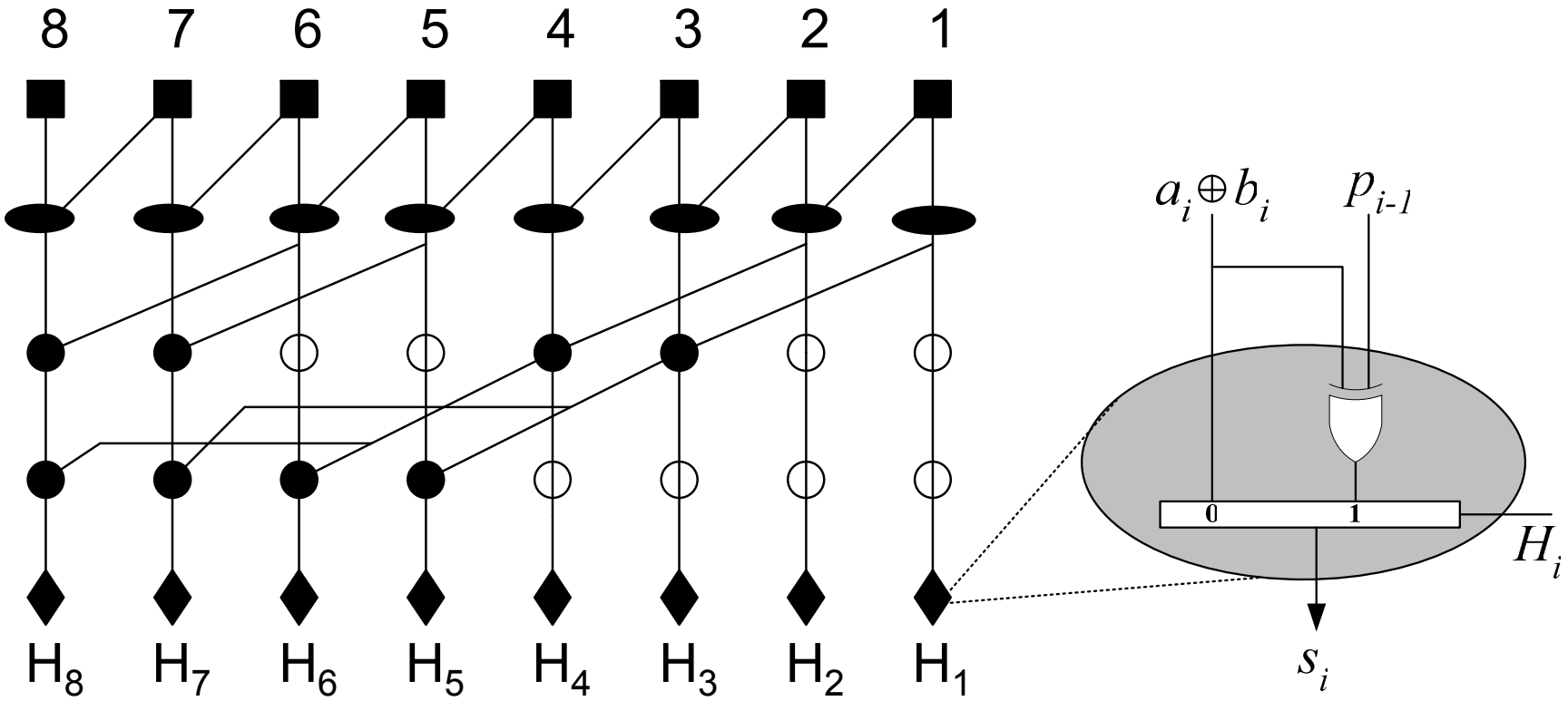
Ling Adders

	Prefix	Ling
Pre-processing:	$g_i = a_i b_i \qquad p_i = a_i \oplus b_i$	$g_i = a_i b_i \qquad p_i = a_i \oplus b_i$
Prefix Computation:	$G_{[i:k]} = G_{[i:j]} + P_{[i:j]} G_{[j-1:k]}$ $P_{[i:k]} = P_{[i:j]} P_{[j-1:k]}$	$G^*_{[i:i-1]} = g_i + g_{i-1}, P^*_{[i:i-1]} = p_i + p_{i-1}$ $G^*_{[i:k]} = G^*_{[i:j]} + P^*_{[i-1:j]} G^*_{[j-2:k]}$ $P^*_{[i:k]} = P^*_{[i:j]} P^*_{[j-2:k]}$
Post-processing:	$c_i = G_{[i-1:0]} + P_{[i-1:0]} \cdot c_0$ $s_i = p_i \oplus c_i$	$c_i = p_{i-1} \cdot G^*_{[i-1:0]}$ $s_i = \overline{G^*_{[i-1:0]}} \cdot d_i + G^*_{[i-1:0]} \cdot (d_i \cdot p_{i-1})$

An 8-bit Ling Adder

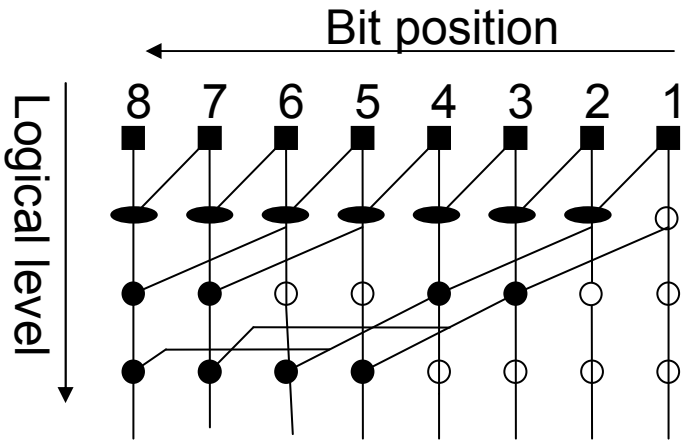


An 8-bit Ling Adder

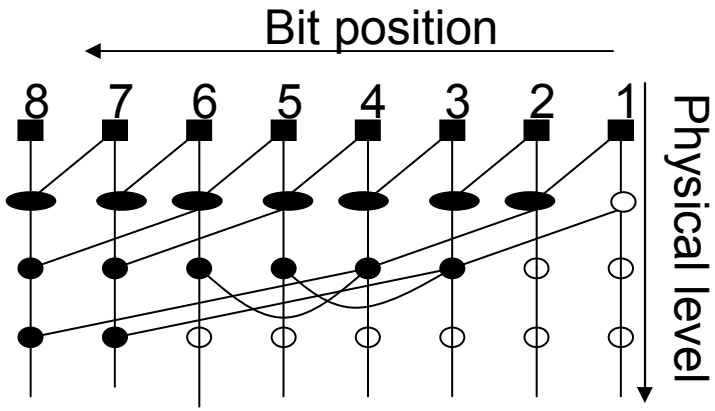


Area Model

- Distinguish physical placement from logical structure, but keep the bit-slice structure.



Logical view

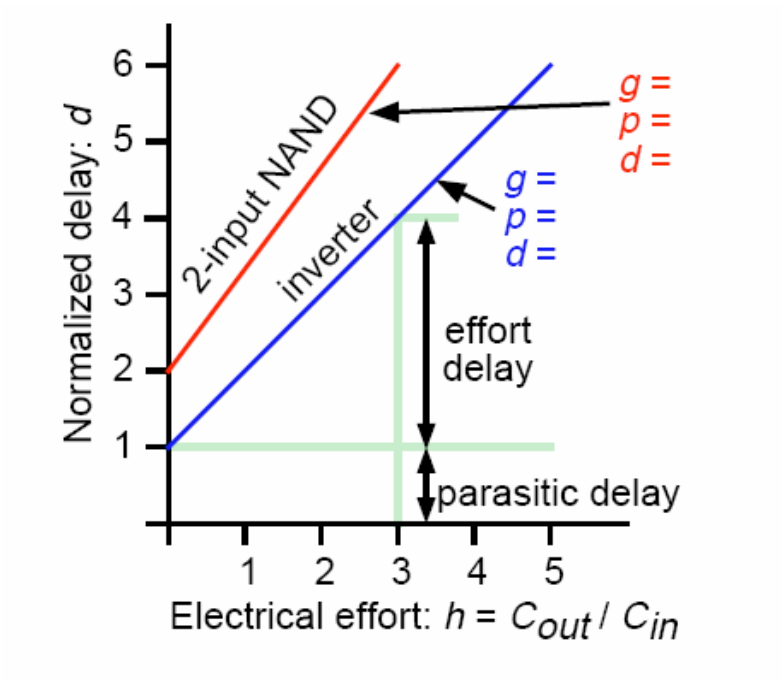


Physical view

Compact placement

Timing Model

- Cell delay calculation:



$$d = f + p$$

Effort Delay

Intrinsic Delay

$$f = g \bullet h$$

Logical Effort

Electrical Effort = Cout/Cin
= (fanouts+wirelength) / size

Intrinsic properties of the cell

Power Model

- Total power consumption:

  Dynamic power + Static Power

- Static power: leakage current of device

  $P_{sta} = \lambda * \#cells$

- Dynamic power: current switching capacitance

  $P_{dyn} = \rho \times C_{load}$

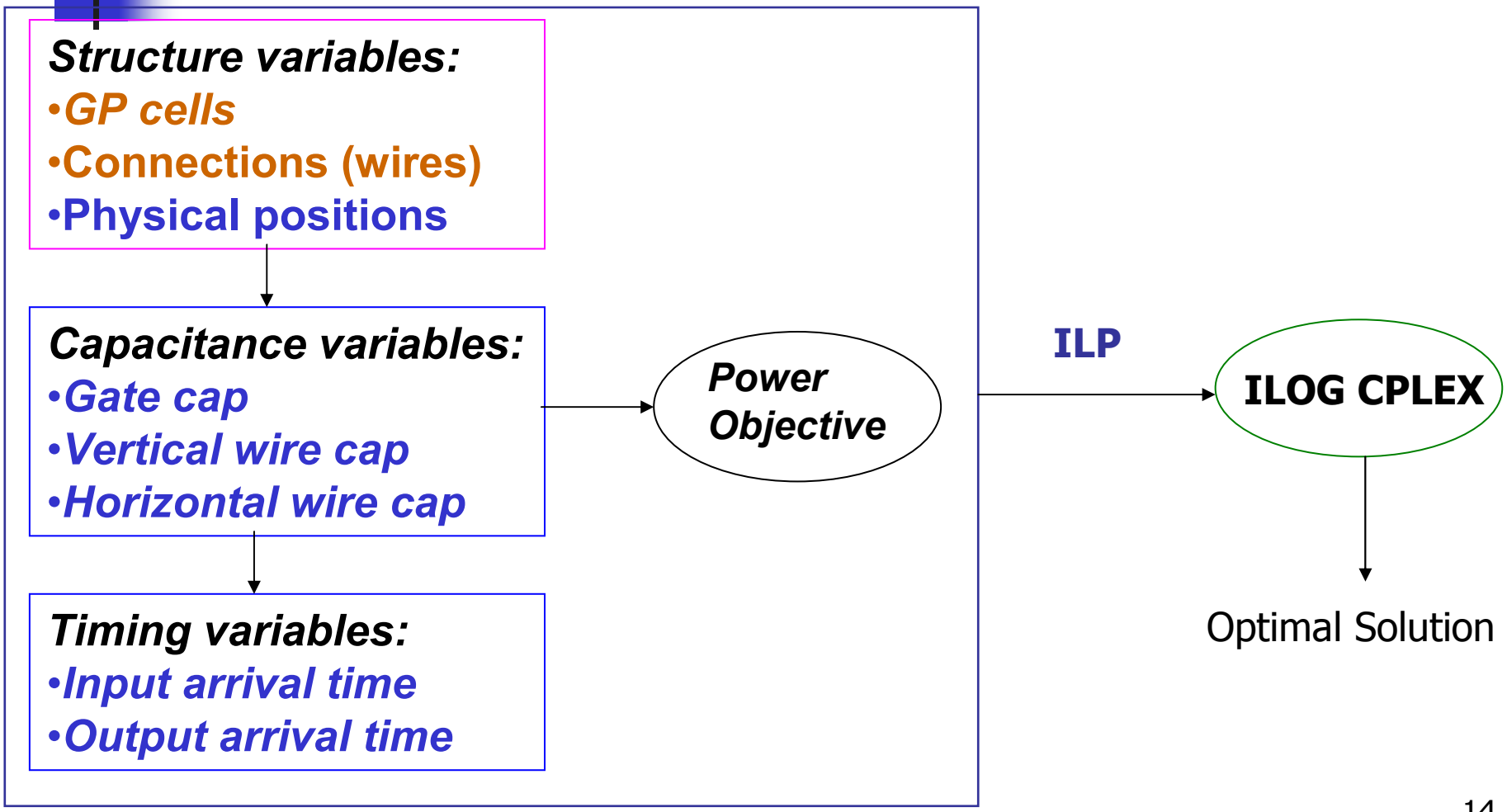
- ρ is the switching probability

  $\rho = j$ (j is the logical level*)

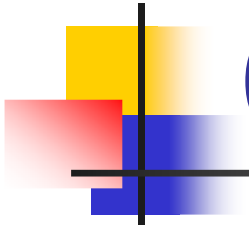
$$P_{total} = P_{dyn} + P_{sta} = j \cdot C_{load} + \lambda \cdot \#cells$$

* Vanichayobon S, etc, “Power-speed Trade-off in Parallel Prefix Circuits”

ILP Formulation Overview

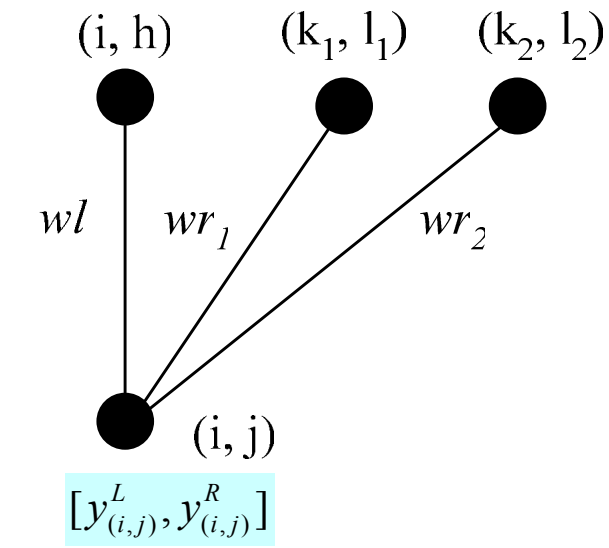


Integer Linear Programming (ILP)



- **ILP:** Linear Programming with integer variables.
- Difficulties and techniques:
  - Constraints are not linear
    - Linearize using pseudo linear constraints
  - Search Space too large
    - Reduce search space
  - Search is slow
    - Add redundant constraints to speedup

Linearization



$$y^R_{(i,h)} = y^L_{(k,l)} + 1 \quad \text{if} \quad wl(i,j,h) = wr1(i,j,k,l) + 1$$

$$\downarrow \quad y^L_{(i,j)} = i$$

$$y^R_{(i,h)} = \sum_{(k,l)} k \cdot wr1(i,j,k,l) + 1 \quad \text{if} \quad wl(i,j,h) = 1$$

$$\downarrow \quad \textit{Linearize}$$

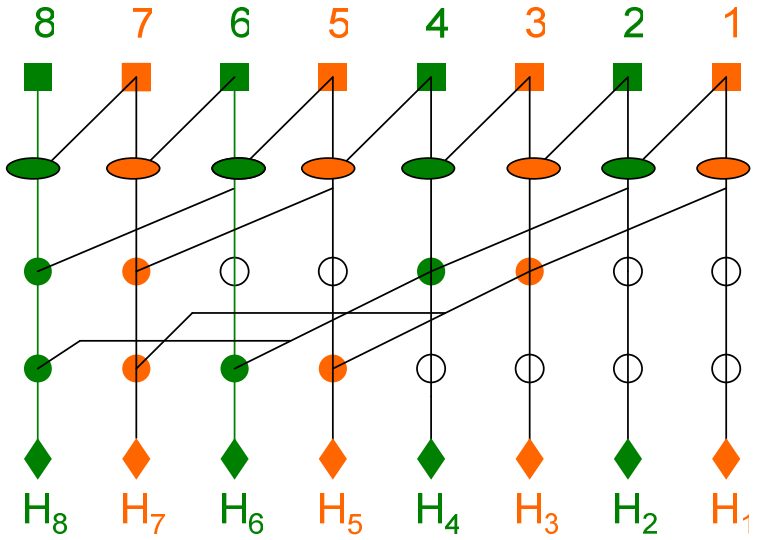
$$y^R_{(i,h)} \geq \sum_{(k,l)} k \cdot wr1(i,j,k,l) - n \cdot (1 - wl(i,j,h)) + 1$$

$$y^R_{(i,h)} \leq \sum_{(k,l)} k \cdot wr1(i,j,k,l) + n \cdot (1 - wl(i,j,h)) + 1$$

Pseudo Linear

Search Space Reduction

- Ling's adder: separate odd and even bits
- Double the bit-width we are able to search



Redundant Constraints

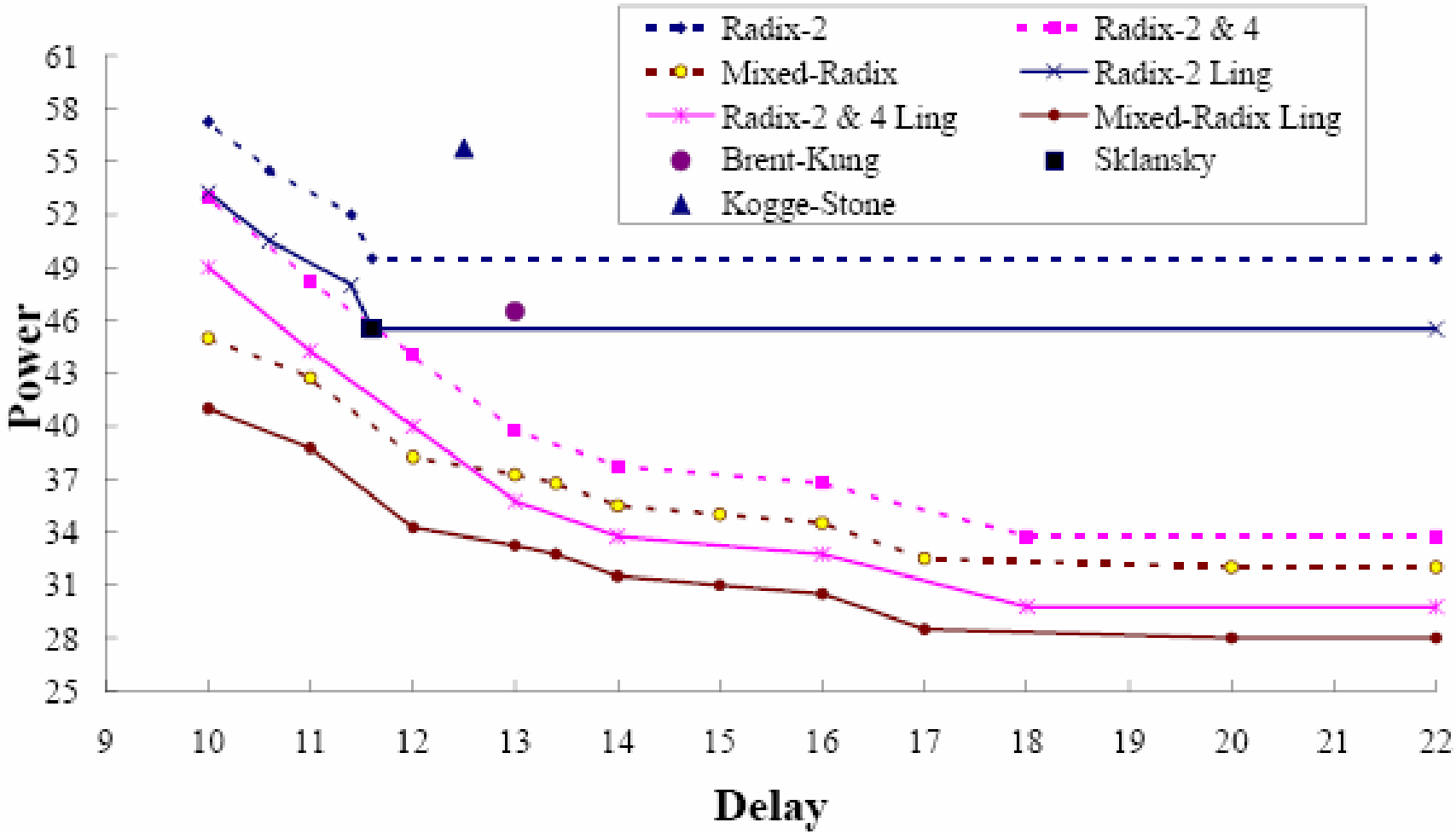
- Cell *(i,j)* is known to have logic level *j* before wire connection
- Assume load is *MinLoad* (fanout=1 with minimum wire length):

$$P_{(i,j)} \geq j \cdot MinLoad + \lambda$$

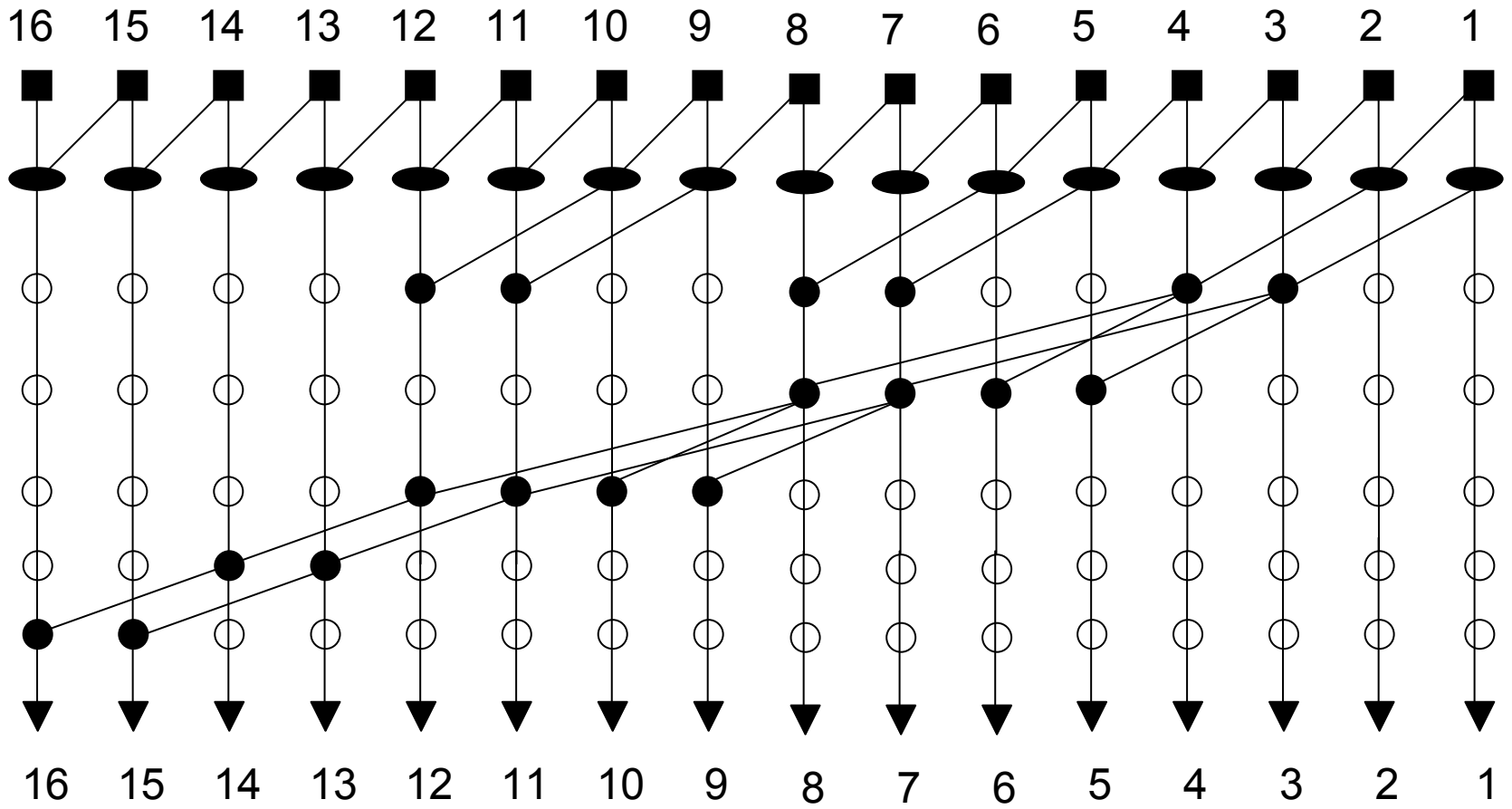
- Cell (i,j) has a path of length *j-1*
- Assume each cell along the path has *MinLoad*

$$T_{(i,j)} \geq j \cdot (PD + LE \cdot MinLoad)$$

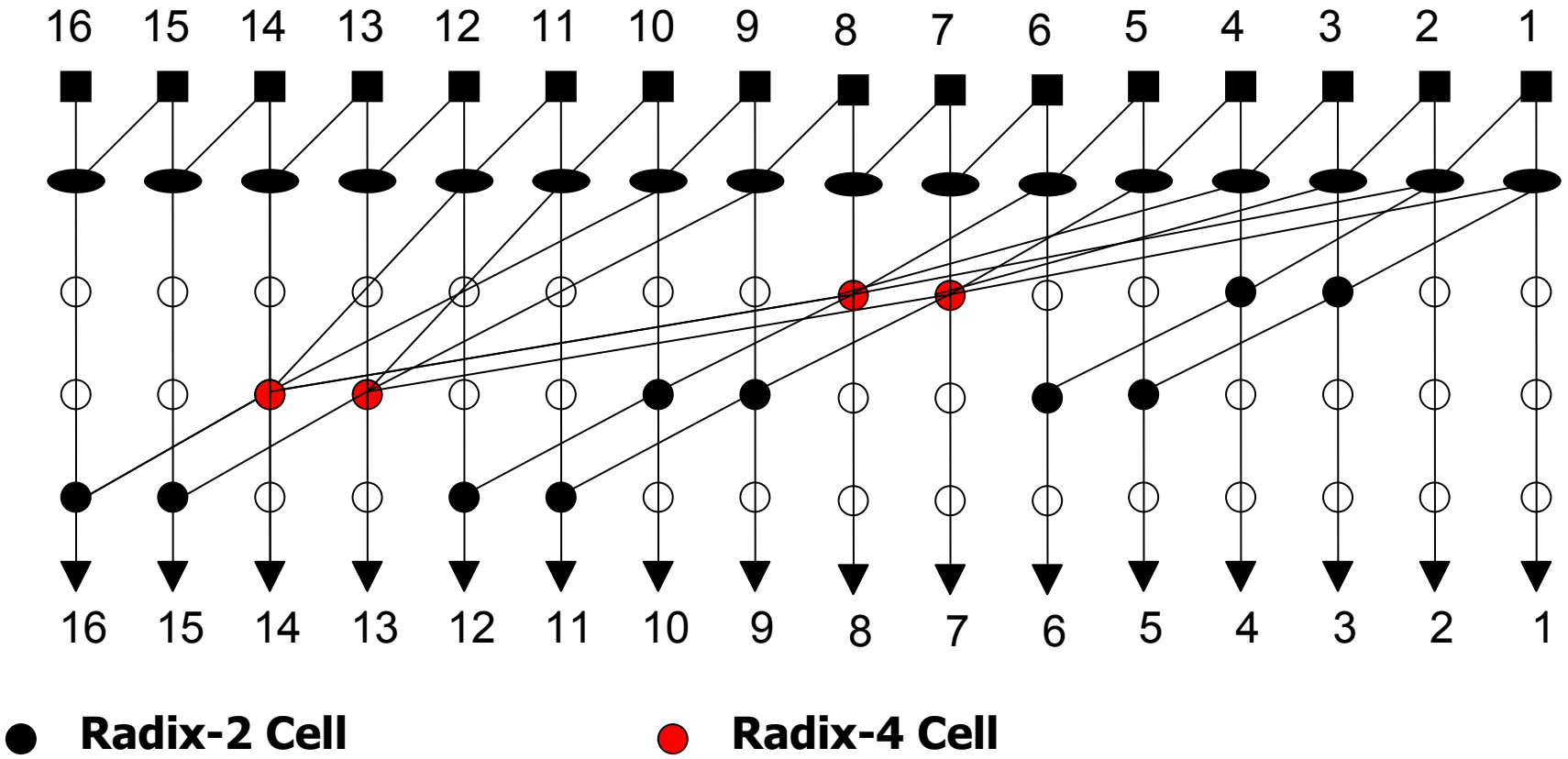
Experiments – 16-bit Uniform Timing



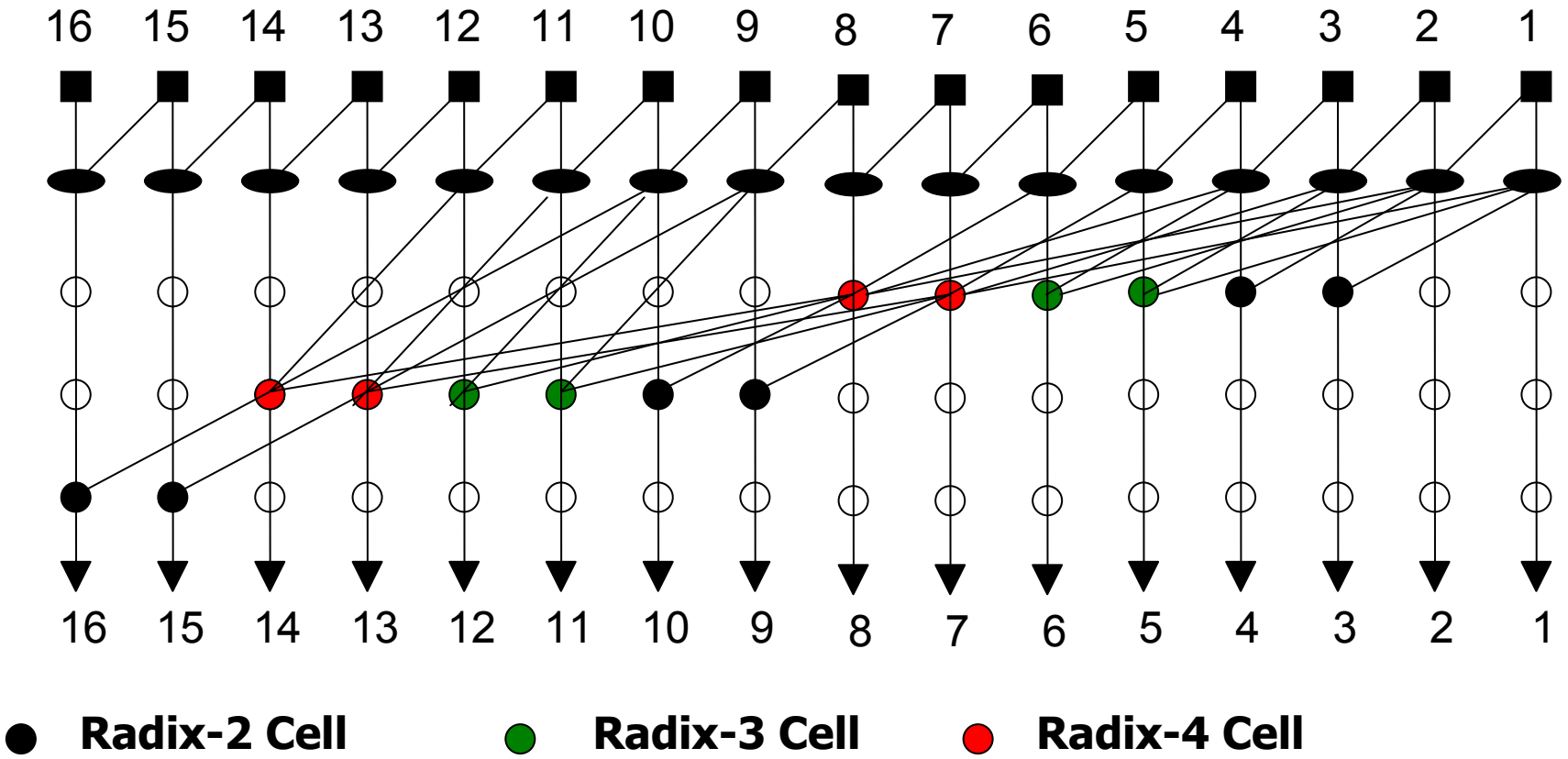
Min-Power Radix-2 Adder



Min-Power Radix-2&4 Adder



Min-Power Mixed-Radix Adder



Experiments – 16-bit Non-uniform Time

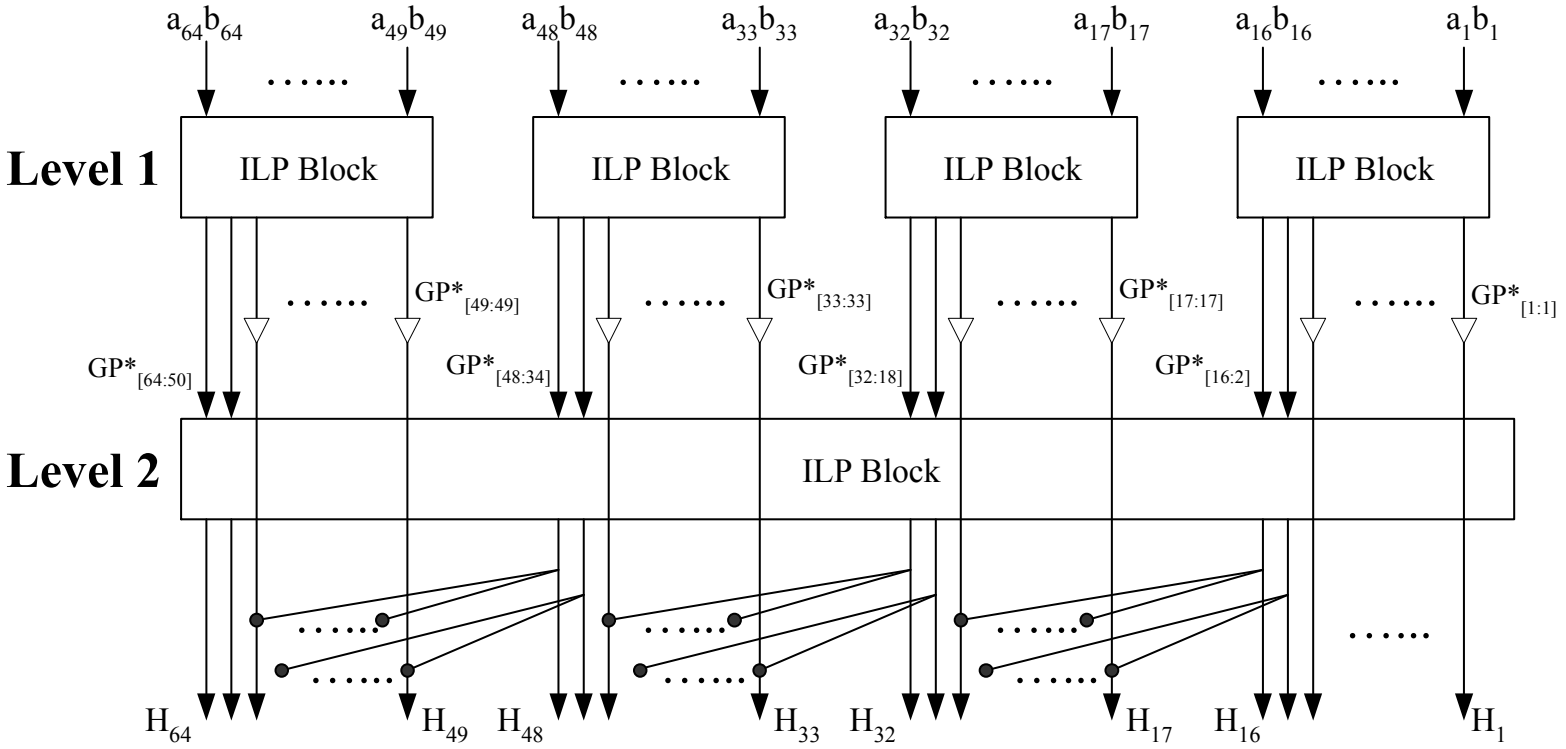
Case	Power (Prefix) (P_{FO4})	Power (Ling) (P_{FO4})
Increasing Arrival Time	35.5	27.0
Decreasing Arrival Time	34.5	30.5
Convex Arrival Time	35.9	32.4
Increasing Required Time	34.5	30.5
Decreasing Required Time	36.5	32.5
Convex Required Time	36.5	32.5

ILP is able to handle non-uniform timings

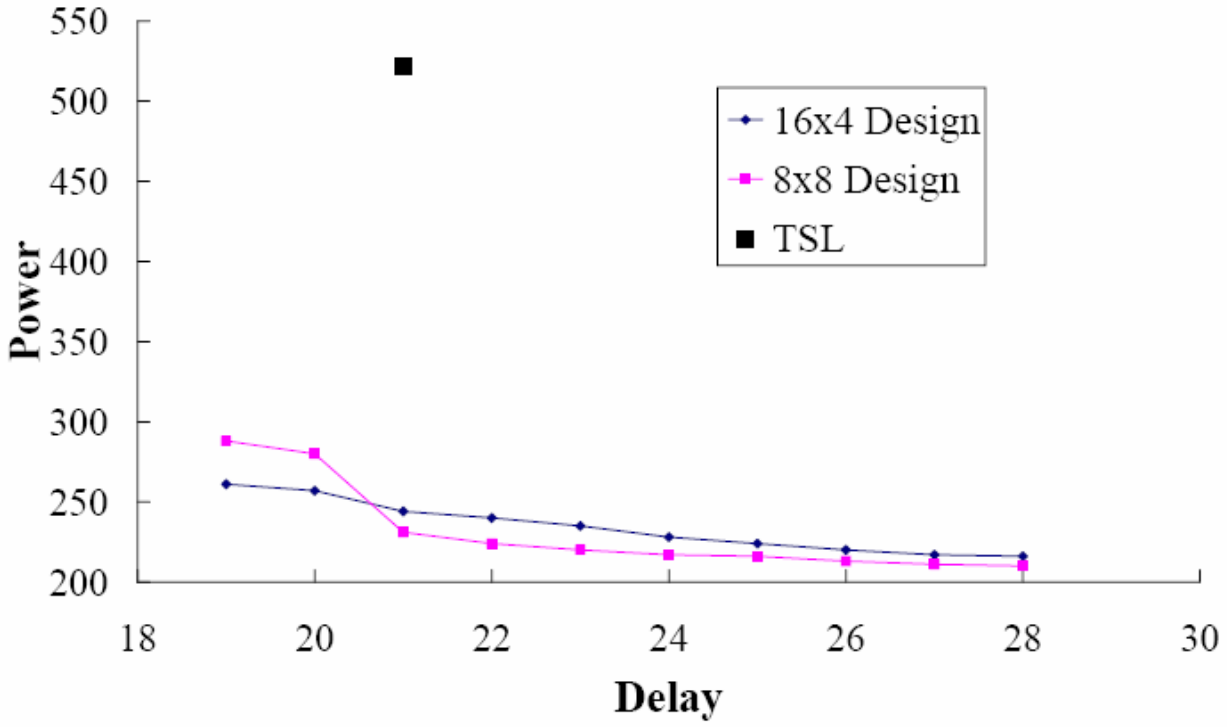
Ling adders are most superior in increasing arrival time – faster carries

Experiments – 64-bit Hierarchical Structure

- Handle high bit-width applications
- 16x4 and 8x8



Experiments – 64-bit Hierarchical Structure



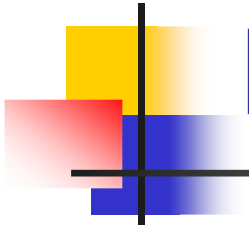
TSL: a 64-bit high-radix three-stage Ling adder

V. Oklobdzija and B. Zeydel, "Energy-Delay Characteristics of CMOS Adders", in *High-Performance Energy-Efficient Microprocessor Design,* pp. 147-170, 2006

ASIC Implementation - Results

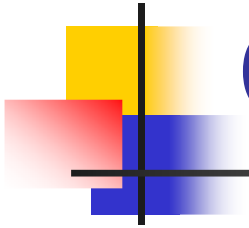
- 64-bit hierarchical design by ILP vs. fast carry look-ahead adder by Synopsys Module Compiler
- TSMC 90nm standard cell library was used

Method	Area (nm^2)	Delay (ns)	Power (mW)
MC	3512	1.0644	5.471
ILP	3833	0.9425	2.541
ILP	3636	0.9607	2.353
ILP	3114	1.1278	1.973



Future Work

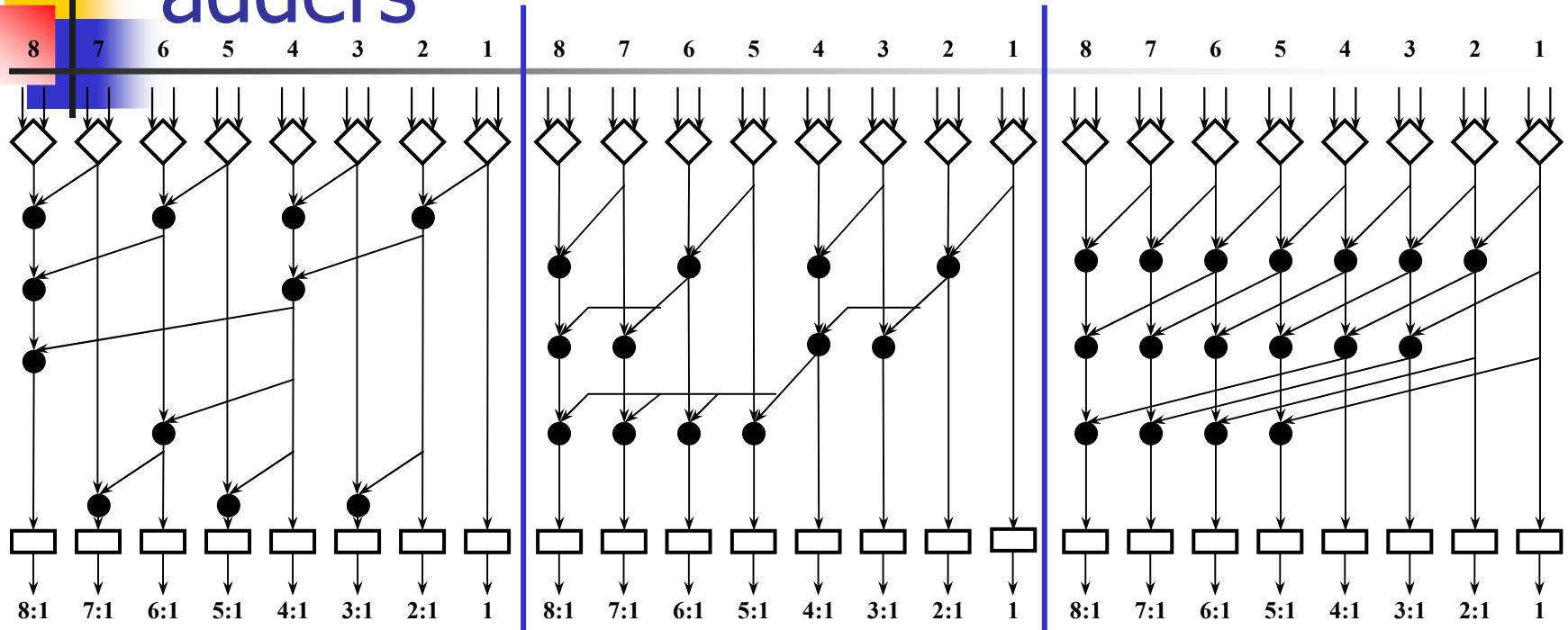
- ILP formulation improvement
  - Expected to handle 32 or 64 bit applications without hierarchical scheme
- Optimizing other computer arithmetic modules
  - Comparator, Multiplier



Q & A

Thank You!

Previous Works – Classical prefix adders

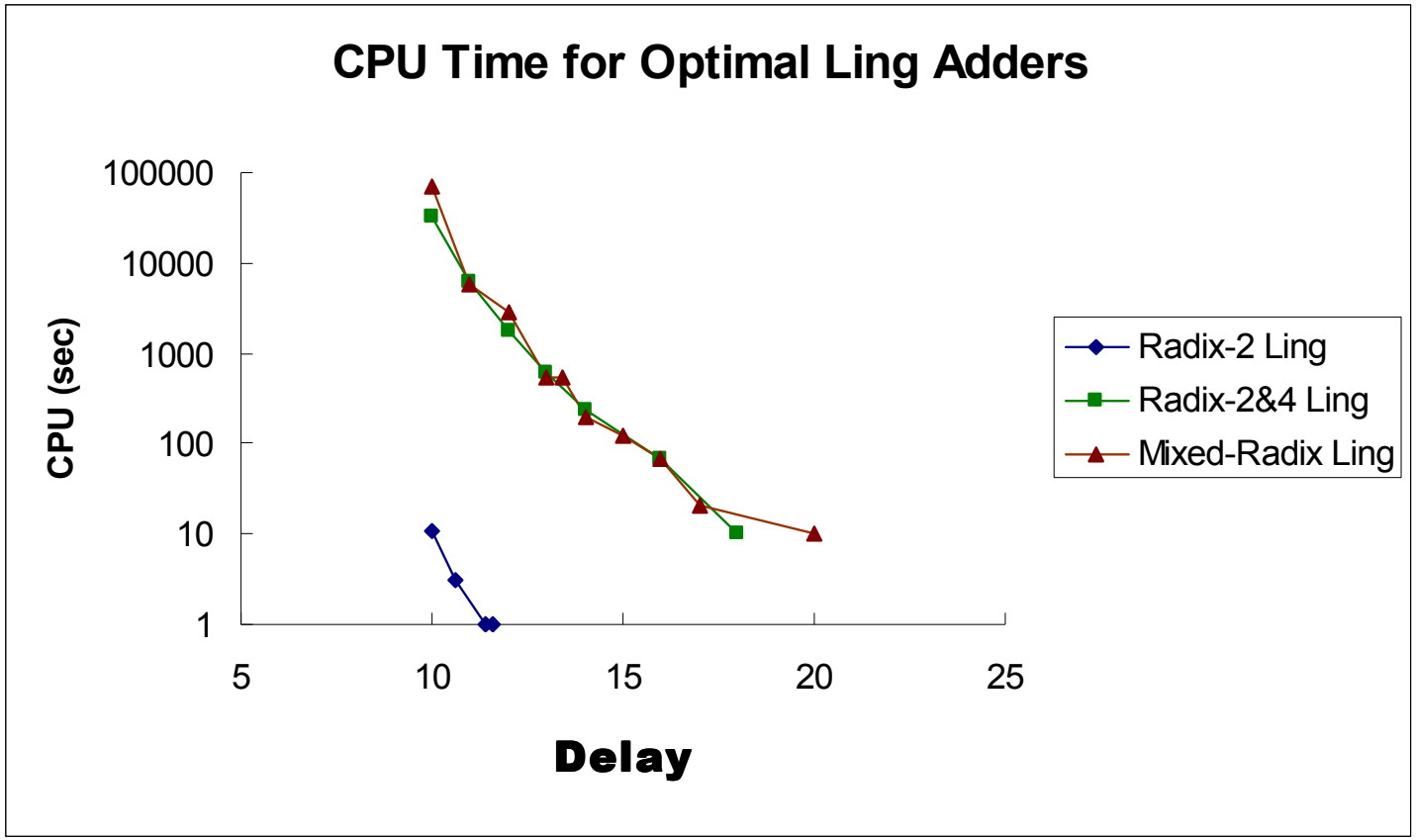


Brent-Kung:
Logical levels: $2\log_2 n - 1$
Max fanouts: 2
Wire tracks: 1

Sklansky:
Logical levels: $\log_2 n$
Max fanouts: n/2
Wire tracks: 1

Kogge-Stone:
Logical levels: $\log_2 n$
Max fanouts: 2
Wire tracks: n/2

Experiments – 16-bit Uniform Timing (CPU Time)



Smaller Delay Const -> Less accurate of LP

ASIC Implementation