Output Prediction Logic: A High Performance CMOS Design Technique

Carl Sechen

Collaborator: Larry McMurchie

Dept. of Electrical Engineering

U. of Washington

Seattle

206-619-5671

sechen@ee.washington.edu

Outline

- Background
- Why static CMOS is slow
- Output Prediction Logic (OPL)
- OPL clocking
- Single-rail results: TSMC 0.25um process
- OPL-differential logic
- Results for TSMC 0.18um process
- Robustness with PVT variations and noise
- World's fastest 64b adder
- Conclusion

Background

- Dynamic circuit families such as domino are commonly used in today’s high-performance microprocessors

- Increased performance due to:
  - reduced input capacitance
  - lower switching thresholds
  - fewer levels of logic (due to the use of wide gates)

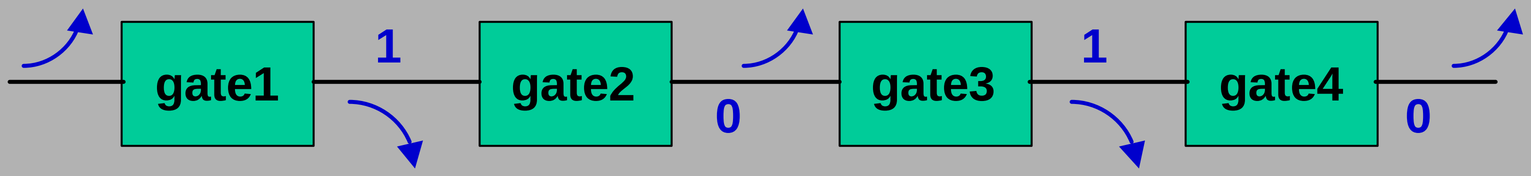
- Dynamic logic yields average speed improvement of 60% over static CMOS for random logic blocks
  - when using synthesis tools tailored specifically for dynamic logic
  - Dual rail domino, DS domino, Monotonic Static, CD domino

Background (cont’d)

- Dynamic circuits have notable disadvantages
- Domino logic must be mapped to a unate network, which usually requires duplication of logic
- Main disadvantage going forward: increased noise sensitivity (compared to static CMOS)
- Increase noise margin: sacrifice performance gain
- Elusive goal: retain the good attributes of static CMOS (high noise immunity and easy technology mapping) while obtaining greater speed

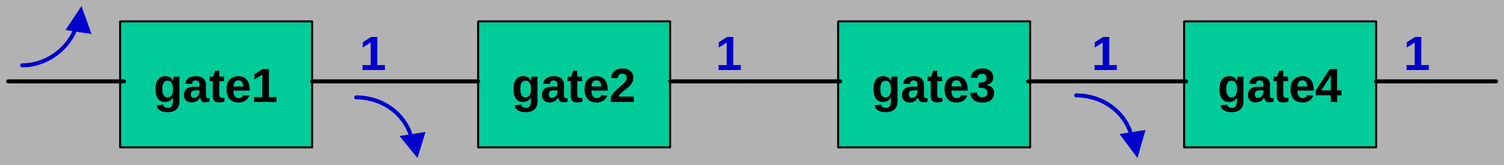
Why Static CMOS is So Slow

- All gates are inherently inverting
- On any circuit path, in the worst case:
  - Every output must fully transition from 1 to 0, or 0 to 1
- You must design for the worst case



Output Prediction Logic

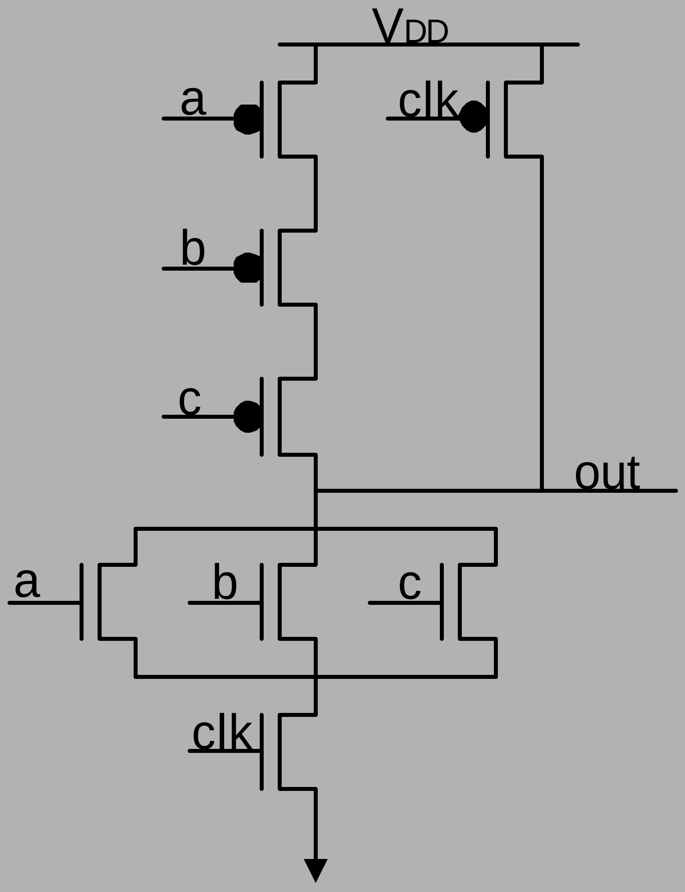
- Goal: reduce the worst case
- Assume all outputs on a critical path will be 1
- You will be correct EXACTLY half the time
  - Every other gate on the path will not have to make ANY transition
- Critical path delay will be reduced by at least 50%



Output Prediction Logic

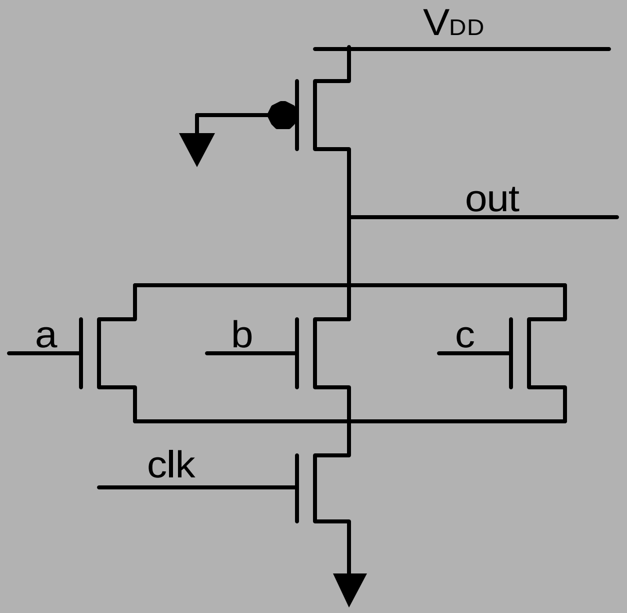
- Problem:
  - 1 at every output (and therefore input) is not a stable state for an inverting gate
  - The 1 will erode (possibly going to 0) in the latter gates of a critical path
- Solution:
  - Disable each gate (1 at inputs and a 1 output is no longer a contradiction)
  - Disable each gate until its inputs are ready for evaluation
  - Predicted output value is therefore maintained

OPL-Static CMOS NOR3

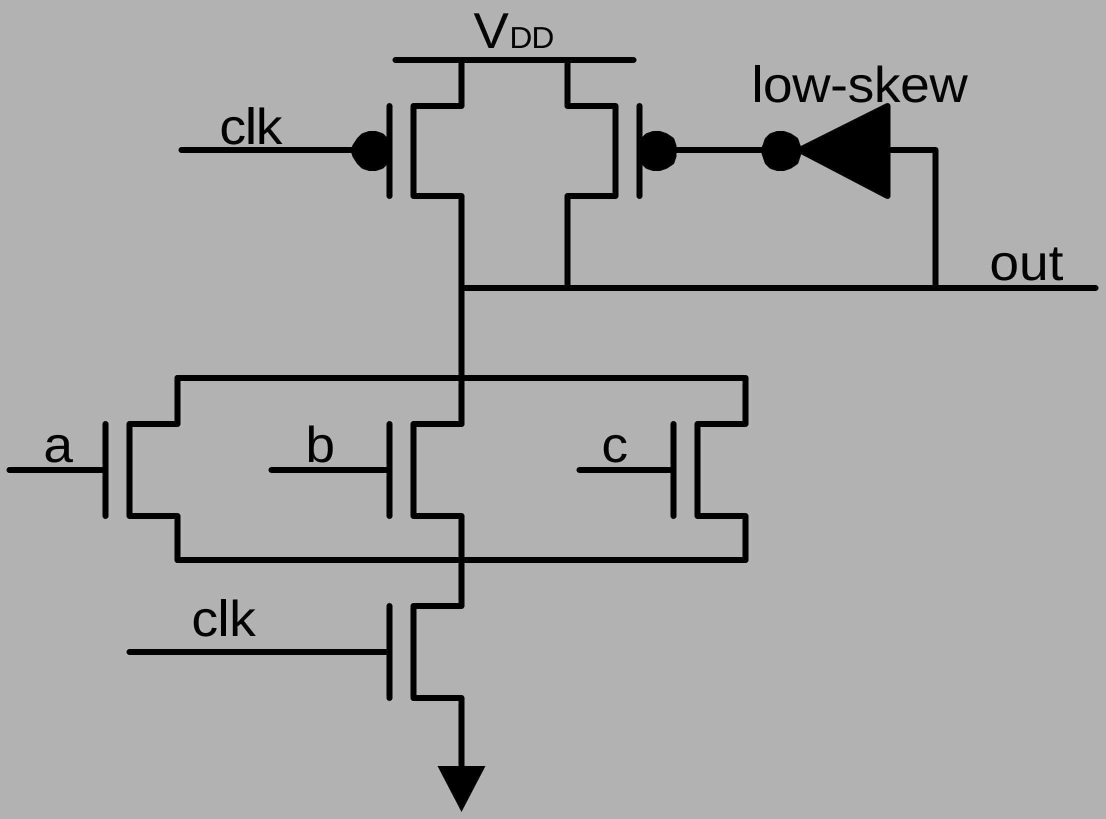


OPL Pseudo-nMOS Gate

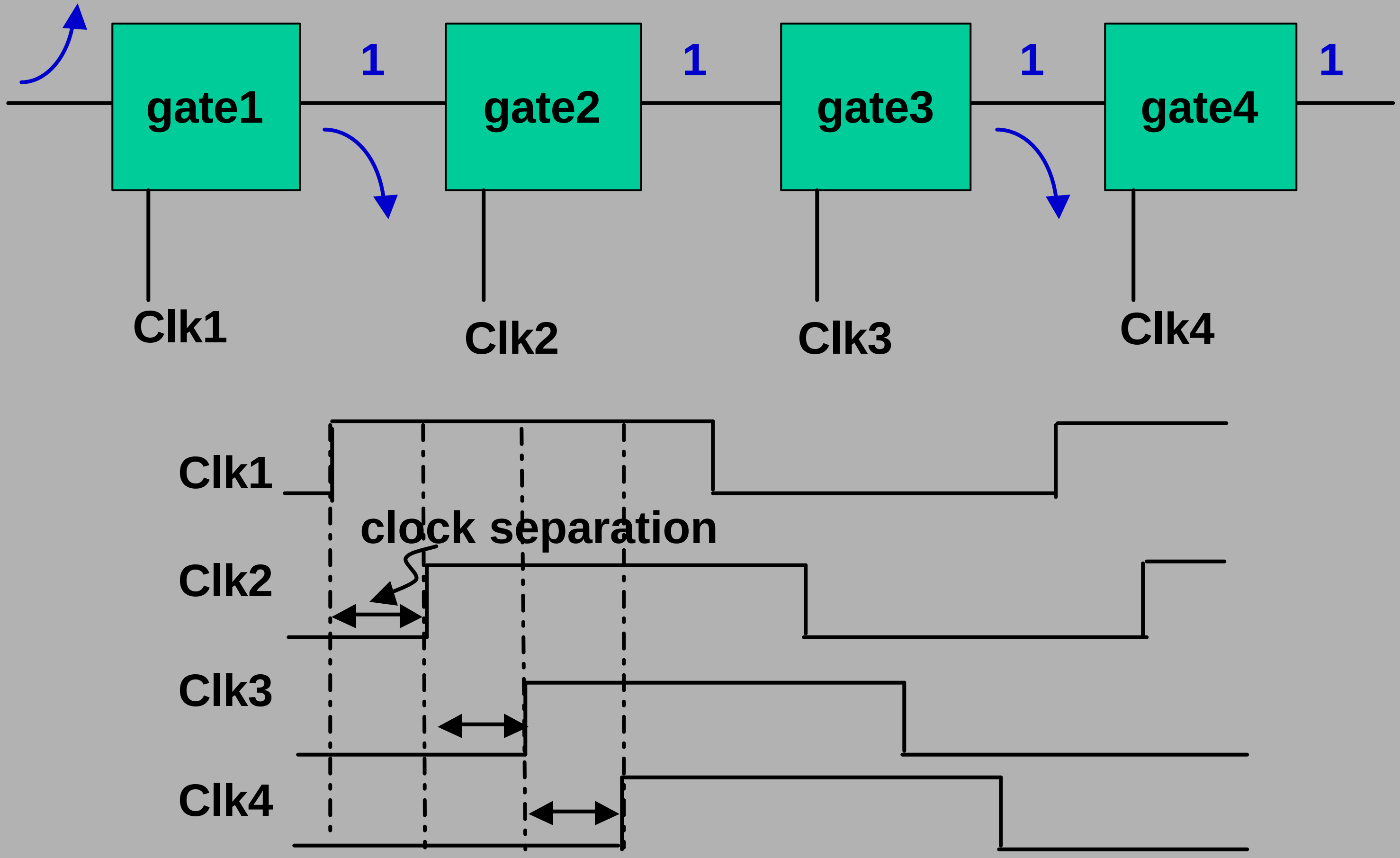
- Tri-state, pre-charge high inverting gate
- Size of pull-up device has small impact on delay
- Reasonable delays with increasing pull-down stack height



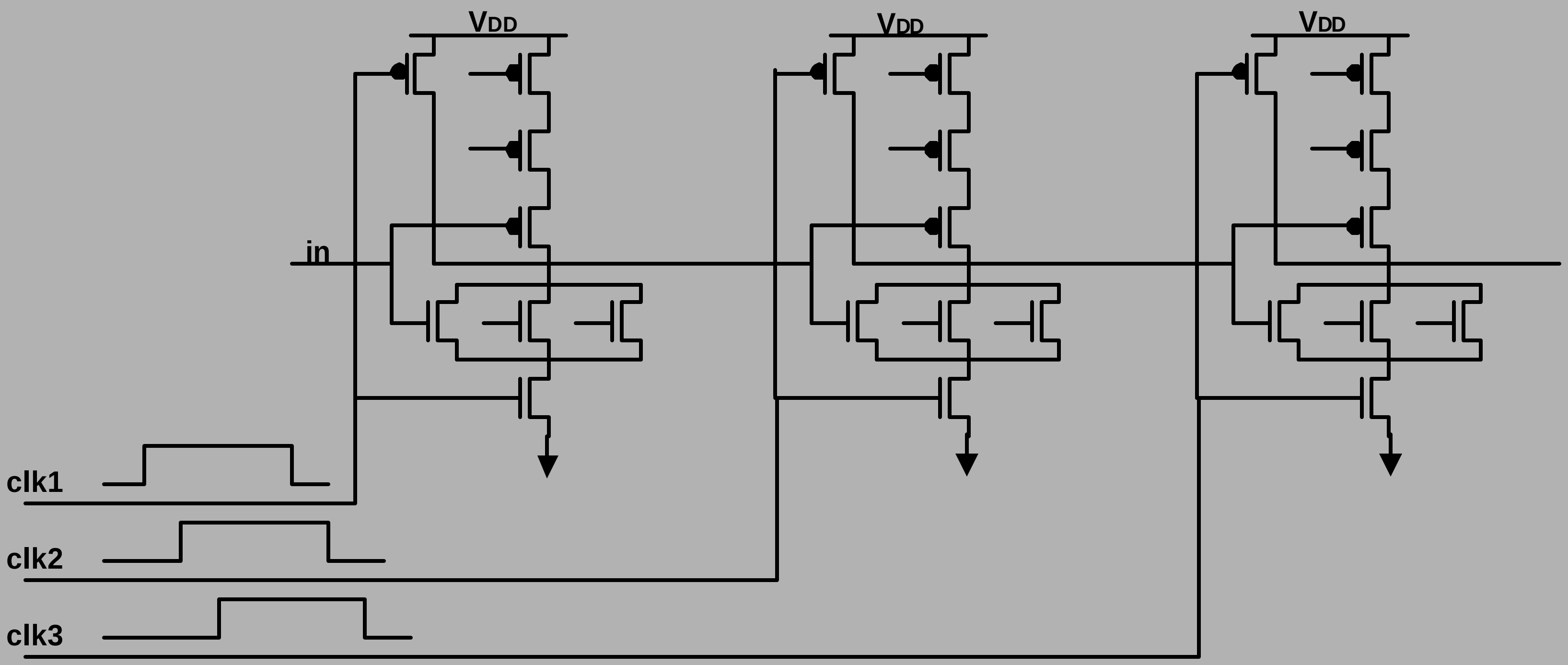
OPL-Dynamic NOR3



OPL Clocking

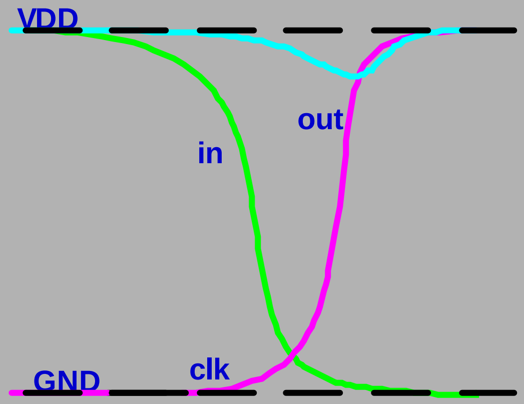


Chain of 3 OPL-Static NOR3's



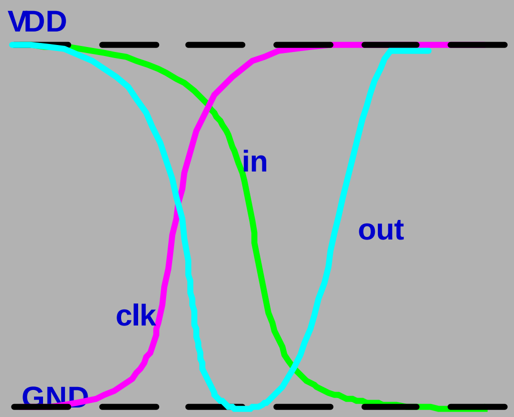
OPL Clocking

- When a clock arrives after inputs have settled:



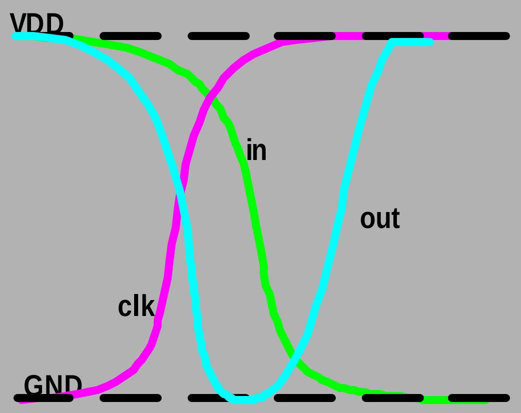
OPL Clocking (cont’d)

- **When a clock arrives BEFORE inputs have settled:**

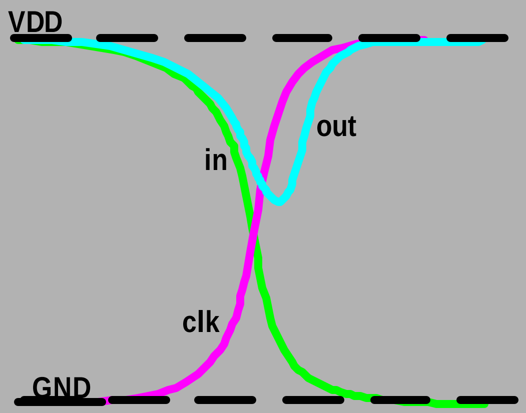


Optimal OPL Clocking

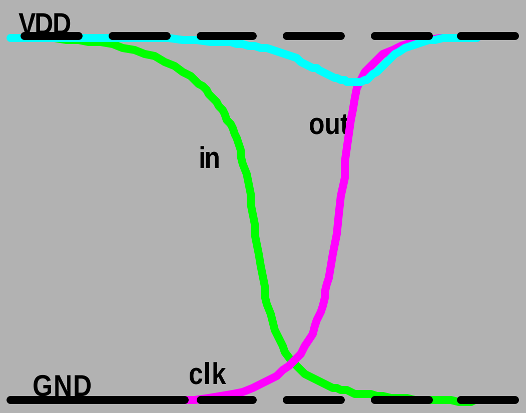
- Consider a gate whose (controlling) input goes low: output should remain 1



a. Early Clock

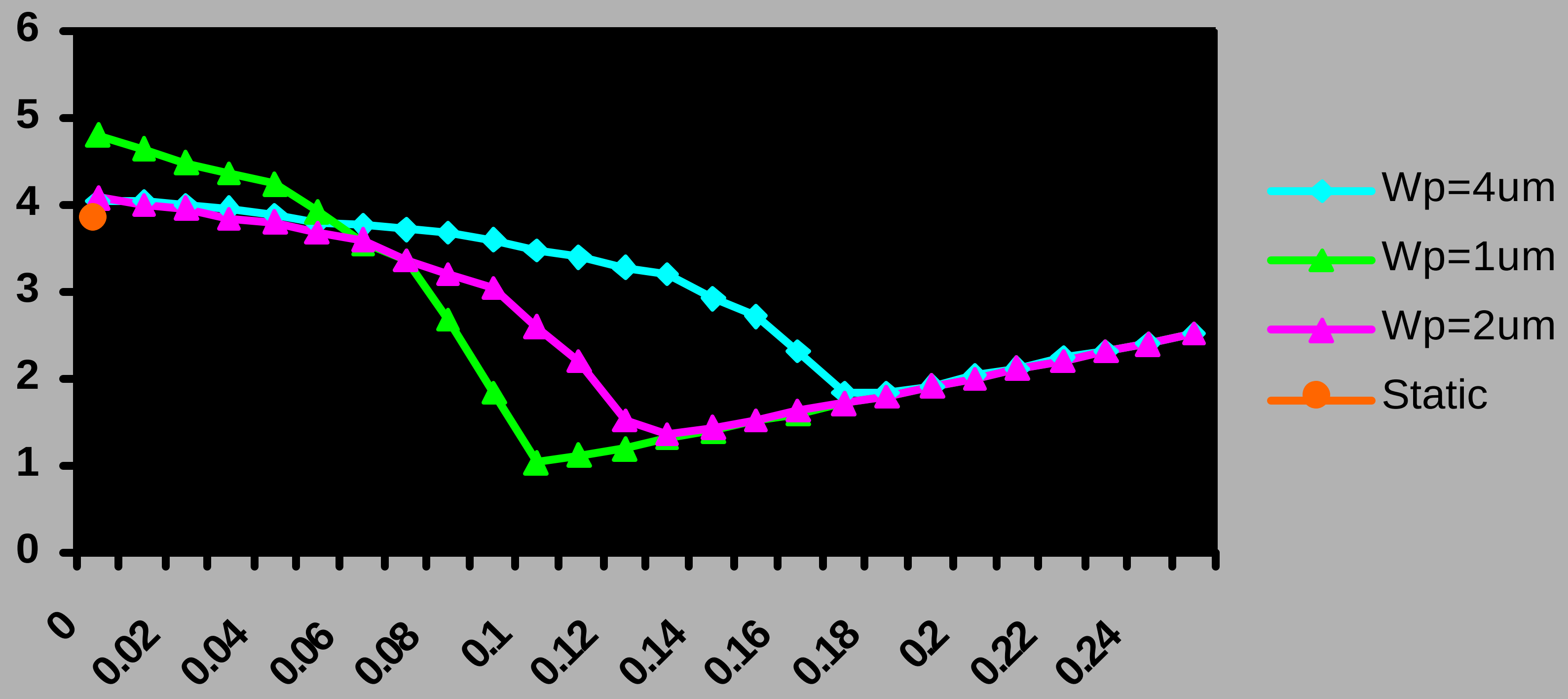


b. Optimal Clock

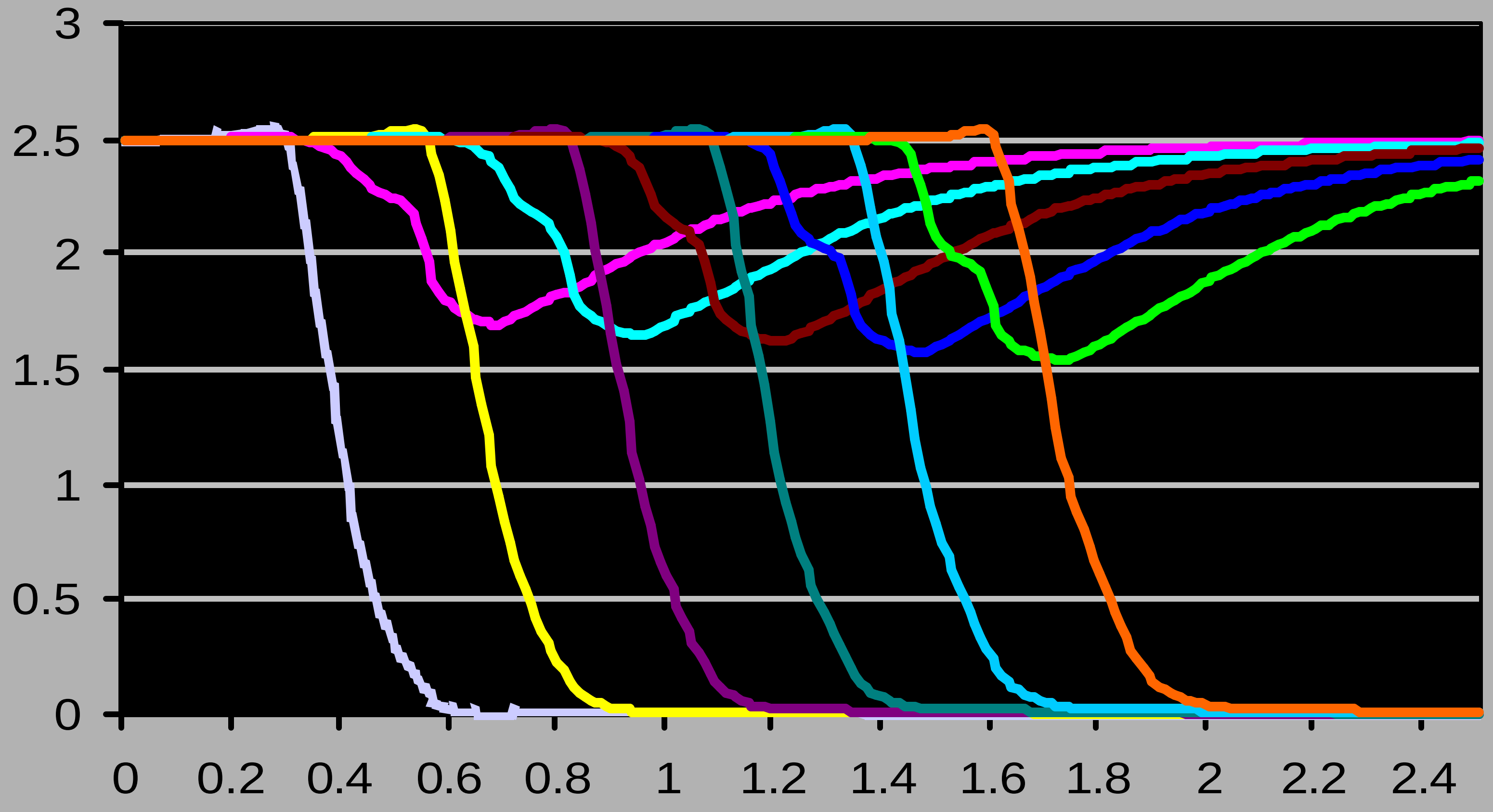


c. Late Clock

Delay vs. Clock Separation for OPL-Static NOR3 Chain



Waveforms for OPL NOR3 Chain

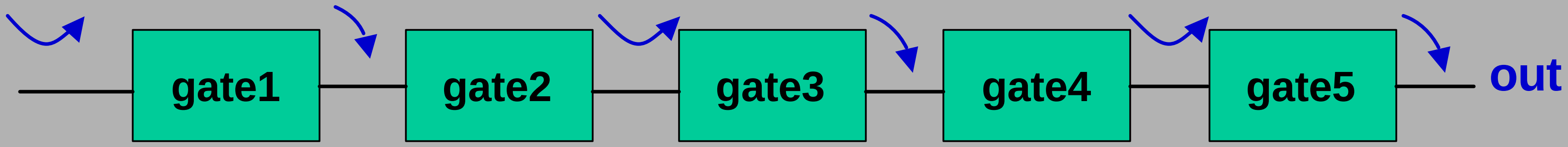


OPL Clocking for General Circuits

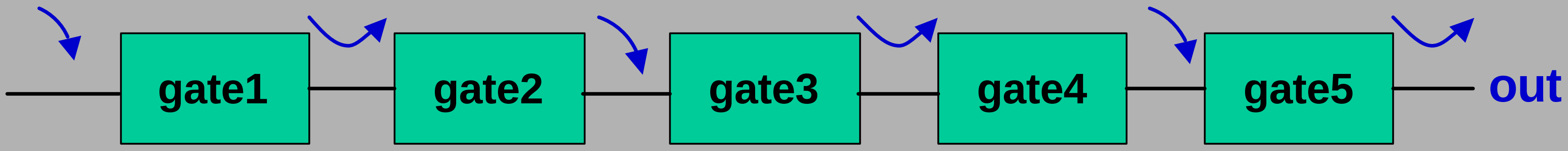
- Levelize the circuit
- Each level gets its own clock phase
- May have to add a buffer (two inverters) if a signal jumps two or more levels

Measuring Delays for OPL

- For each primary output, you must check two cases to get the worst-case delay:
  - output low



  - output high



10-Gate Critical Path Delays (FO of 4)

- To determine the performance possible with OPL, we simulated critical paths consisting of 10 gates, each gate in the path driving a load of four identical gates
- We used nominal simulation parameters for the 0.25 micron TSMC process, having a drawn channel length of 0.30 microns

10-Gate Critical Path Delays (FO of 4)

- **Pull-down nMOS devices for all gates were sized to have an effective width of 2 microns**
  - **pull-down stack of *k* transistors implies transistor sizes were 2*k* um**
- **Static CMOS pMOS transistors were uniformly sized by sweeping their size versus overall delay for the chain of 10 gates**
  - **select the size that minimized the worst case delay for the chain**

10-Gate Critical Path Delays (FO of 4)

Chain Type	Static CMOS	OPL-static	OPL-pseudo	OPL-dynamic
INV	1.62ns (1.0)	430ps (3.77)	420ps (3.86)	430ps (3.77)
NOR3	3.83ns (1.0)	1.34ns (2.86)	710ps (5.39)	760ps (5.04)
NAND2	2.45ns (1.0)	940ps (2.61)	930ps (2.63)	1.02ns (2.40)
NAND3	3.32ns (1.0)	1.44ns (2.31)	1.54ns (2.16)	1.54ns (2.16)
NAND4	4.24ns (1.0)	1.97ns (2.15)	2.16ns (1.96)	2.15ns (1.97)
AOI22	4.75ns (1.0)	2.13ns (2.23)	1.81ns (2.62)	1.80ns (2.64)
AOI222	6.75ns (1.0)	3.04ns (2.22)	2.63ns (2.57)	2.49ns (2.71)
Average Speedup	(1.0)	(2.59)	(3.03)	(2.96)

Energy Consumption

Chain Type	Static CMOS	OPL-static	OPL-pseudo	OPL-dynamic
INV	2.00 pJ (1.0)	3.80 pJ (1.90)	4.97 pJ (2.49)	4.41pJ (2.21)
NOR3	3.19 pJ (1.0)	4.45 pJ (1.39)	6.07 pJ (1.90)	4.47pJ (1.40)
NAND2	3.83 pJ (1.0)	5.00 pJ (1.31)	8.39 pJ (2.19)	5.60pJ (1.46)
NAND3	6.23 pJ (1.0)	6.66 pJ (1.07)	12.7pJ (2.04)	7.51pJ (1.21)
NAND4	8.65 pJ (1.0)	12.7 pJ (1.47)	19.3 pJ (2.23)	10.0pJ (1.16)
AOI22	6.13 pJ (1.0)	6.31 pJ (1.03)	12.8 pJ (2.09)	7.01pJ (1.14)
AOI222	7.08 pJ (1.0)	7.70 pJ (1.09)	16.7 pJ (2.36)	8.09pJ (1.14)
Average	(1.0)	(1.32)	(2.19)	(1.39)

Delays for an 8-Gate (FO of 4) Heterogeneous Critical Path

- NOR3, NAND3, AOI22, INV, INV, NOR3, NAND3, and AOI22
- Having the gates so ordered means that each gate type will have to pull down once and stay high once
- Each gate drives a load of four identical gates
- The device sizes used were exactly those selected for the uniform chains

Logic Family	Delay	Speedup
Static CMOS	2.13ns	1.0
OPL-static	910ps	2.34
OPL-pseudo	650ps	3.28
OPL-dynamic	688ps	3.10

Delays for Two Implementations a 32-bit Carry Look-Ahead Adder

Logic Family	Delay	Speedup	CLA type
Static CMOS	3.0ns	1.0	Three levels
OPL-static	1.5ns	2.0	Three levels
OPL-pseudo	1.8ns	1.65	Three levels
OPL-pseudo	552ps	5.43	Two levels

- First three designs used all NAND gates; last one is all NOR gates

OPL Applied to Random Logic

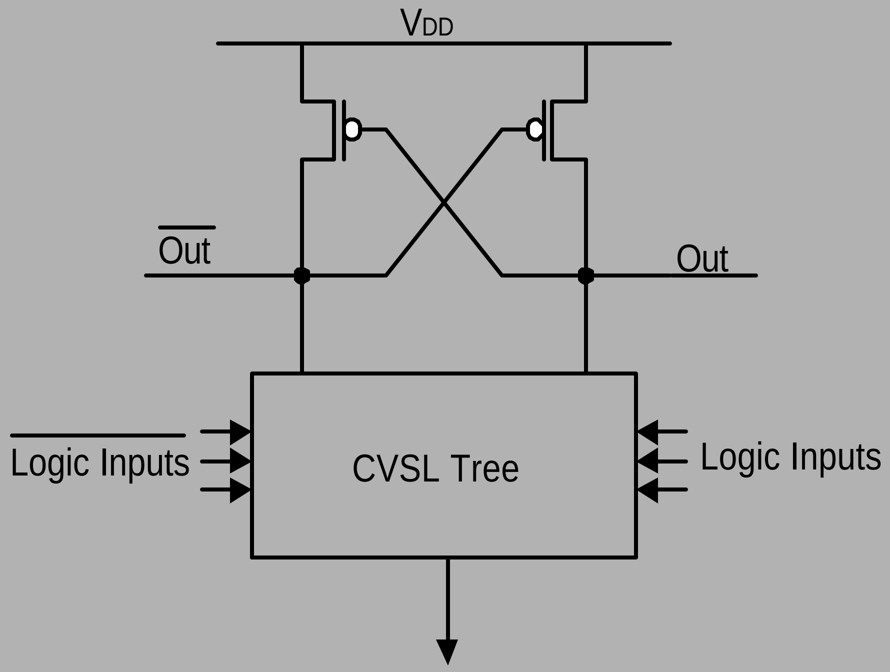
- Early experiments assigned a single clock to all gates in the same level
- At minimum total delay, some gates showed large glitches
- Two methods were used to reduce glitching in selected gates and improve total delay:
  - a) Increase pull-up sizes to allow better recovery
  - b) Allow more time for (late arriving) inputs to settle. This is done by moving glitching gate back in time by one clock
- Optimized OPL algorithm employs both methods

Delays for ISCAS Random Logic Benchmarks

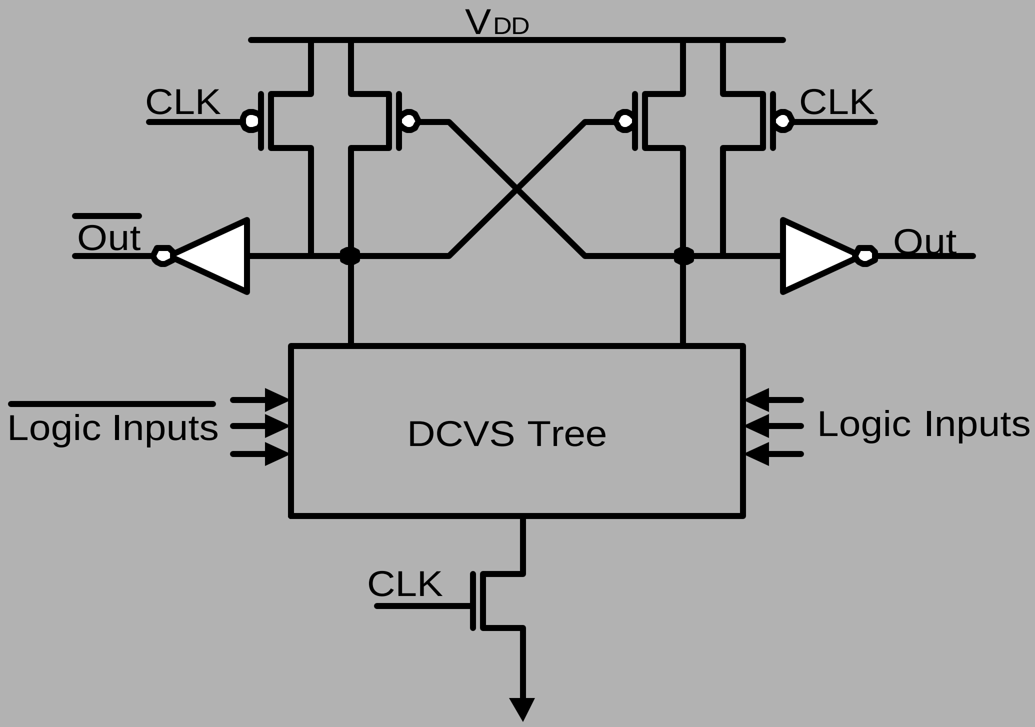
Benchmark (levels)	Static	OPL-Static	OPL-Pseudo
t481(7)	910ps (1.0)	0.46ns (1.98)	0.430ns (2.12)
term1(10)	1.38ns (1.0)	0.70ns (1.97)	0.565ns (2.44)
x3(10)	2.58ns (1.0)	0.67ns (3.85)	0.537ns (4.80)
Rot(16)	2.19ns (1.0)	1.05ns (2.09)	1.07ns (2.05)
Dalu(14)	2.35ns (1.0)	960ps (2.45)	0.857ns (2.73)
Average speedup	(1.0)	(2.47)	(2.82)

- Much higher speed-ups will be obtained when we use a technology mapper specifically for OPL

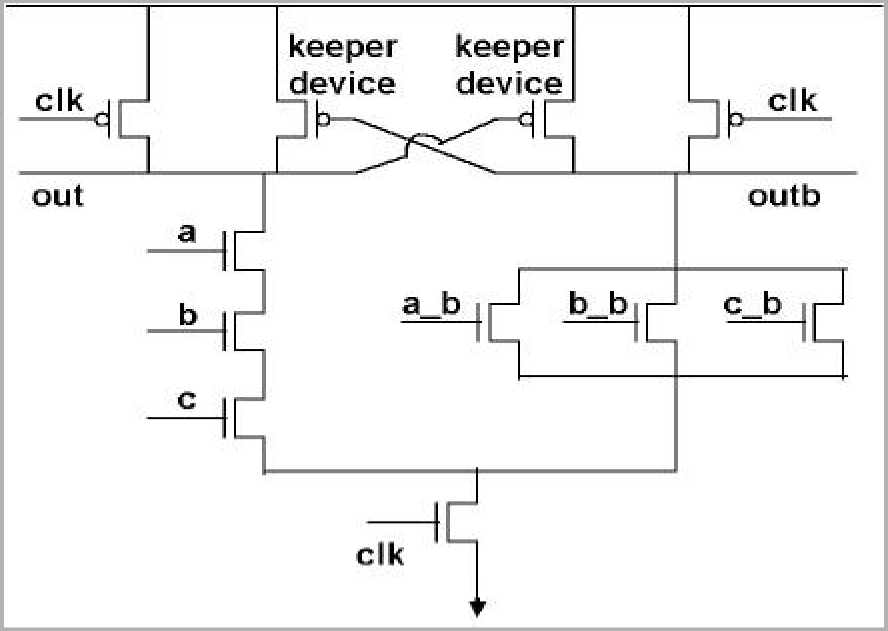
Conventional CVSL Gate



Domino CVSL Gate



OPL-differential NAND3 Gate



Delays (ns) for Chains of 10 Gates

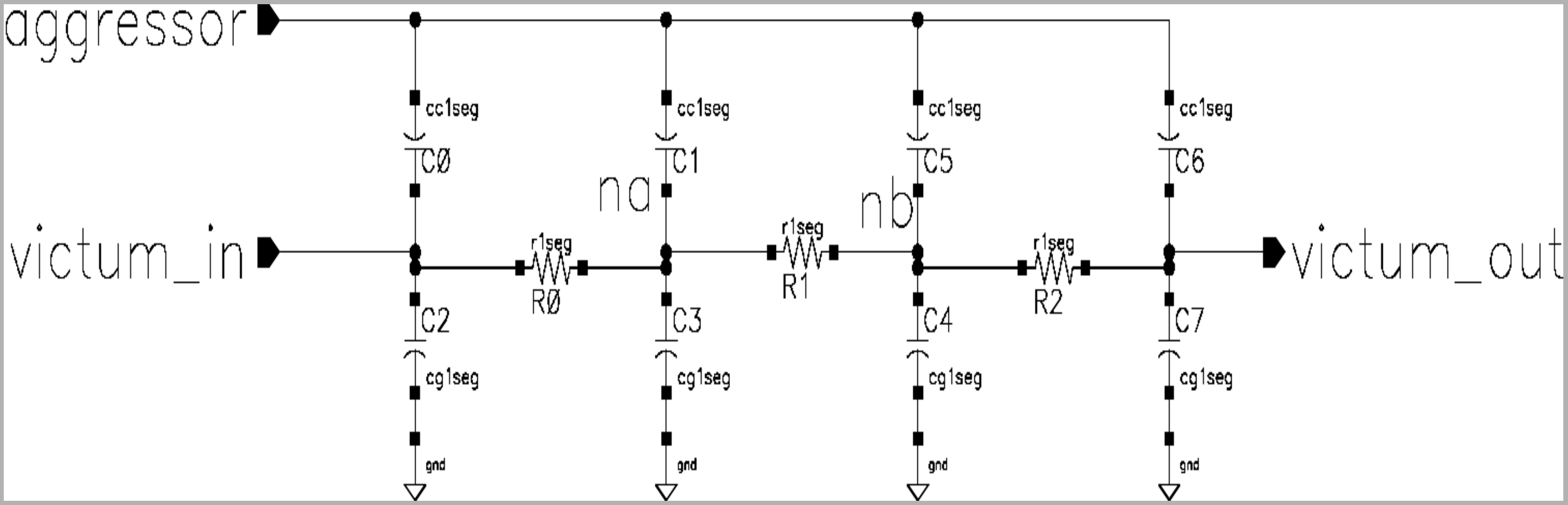
ChainType	Static CMOS	Diff. Domino	OPL-Dynamic	OPL-Diff.
INV	0.84 (1.0)	0.62 (0.74)	0.22 (0.26)	0.16 (0.19)
NOR2	1.26 (1.0)	0.66 (0.52)	0.30 (0.24)	0.25 (0.20)
NOR3	1.59 (1.0)	0.74 (0.47)	0.33 (0.21)	0.30 (0.19)
NOR4	2.34 (1.0)	0.89 (0.38)	0.41 (0.18)	0.34 (0.15)
NAND2	1.02 (1.0)	0.66 (0.65)	0.46 (0.45)	0.30 (0.29)
NAND3	1.38 (1.0)	0.80 (0.58)	0.72 (0.52)	0.45 (0.33)
NAND4	1.48 (1.0)	0.89 (0.60)	0.81 (0.55)	0.52 (0.35)
AOI21	1.30 (1.0)	0.72 (0.55)	0.41 (0.32)	0.35 (0.27)
AOI22	1.74 (1.0)	0.82 (0.47)	0.54 (0.31)	0.33 (0.19)
AOI222	2.95 (1.0)	1.01 (0.34)	0.72 (0.24)	0.54 (0.18)
AOI31	1.76 (1.0)	0.83 (0.47)	0.55 (0.31)	0.52 (0.30)
AOI33	2.60 (1.0)	1.00 (0.38)	0.82 (0.32)	0.50 (0.19)
AOI333	4.00 (1.0)	1.19 (0.30)	0.97 (0.24)	0.59 (0.14)
AOI321	2.43 (1.0)	0.91 (0.37)	0.55 (0.23)	0.54 (0.22)
average	1.91 (1.0)	0.84 (0.44)	0.56 (0.29)	0.41 (0.21)

Delays (ns) for Chains of 10 Gates with PVT Variations and Clock Skew

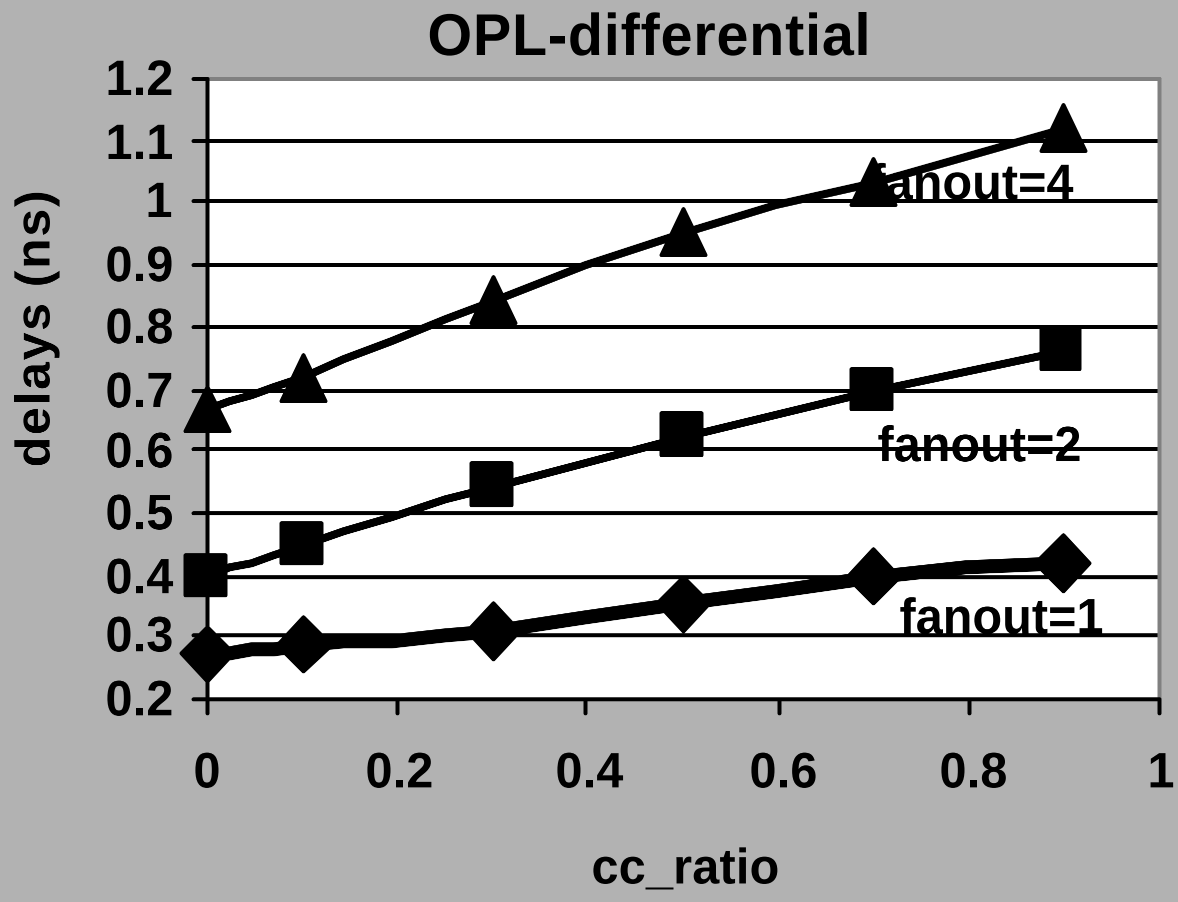
- Gaussian distribution of clock separation with 2.5? = 30ps at .25 micron
- Gaussian distribution of clock separation with 2.5? = 15ps at .18 micron
- Gaussian distribution of channel length with 2.5? = 20% of nominal

Chain Type	Static CMOS	Diff. Domino	OPL-Dynamic	OPL-Diff.
NOR3	1.59 / 1.65	0.62 / 0.80	0.33 / 0.39	0.30 / 0.38
NAND3	1.38 / 1.52	0.80 / 0.87	0.72 / 0.78	0.45 / 0.48
AOI22	1.74 / 1.81	0.82 / 0.85	0.54 / 0.59	0.33 / 0.40
AOI222	2.95 / 3.02	1.01 / 1.06	0.72 / 0.82	0.54 / 0.63
AOI333	4.00 / 4.25	1.19 / 1.24	0.97 / 1.07	0.59 / 0.73
average	2.33 / 2.45	0.89 / 0.964	0.66 / 0.73	0.44 / 0.52

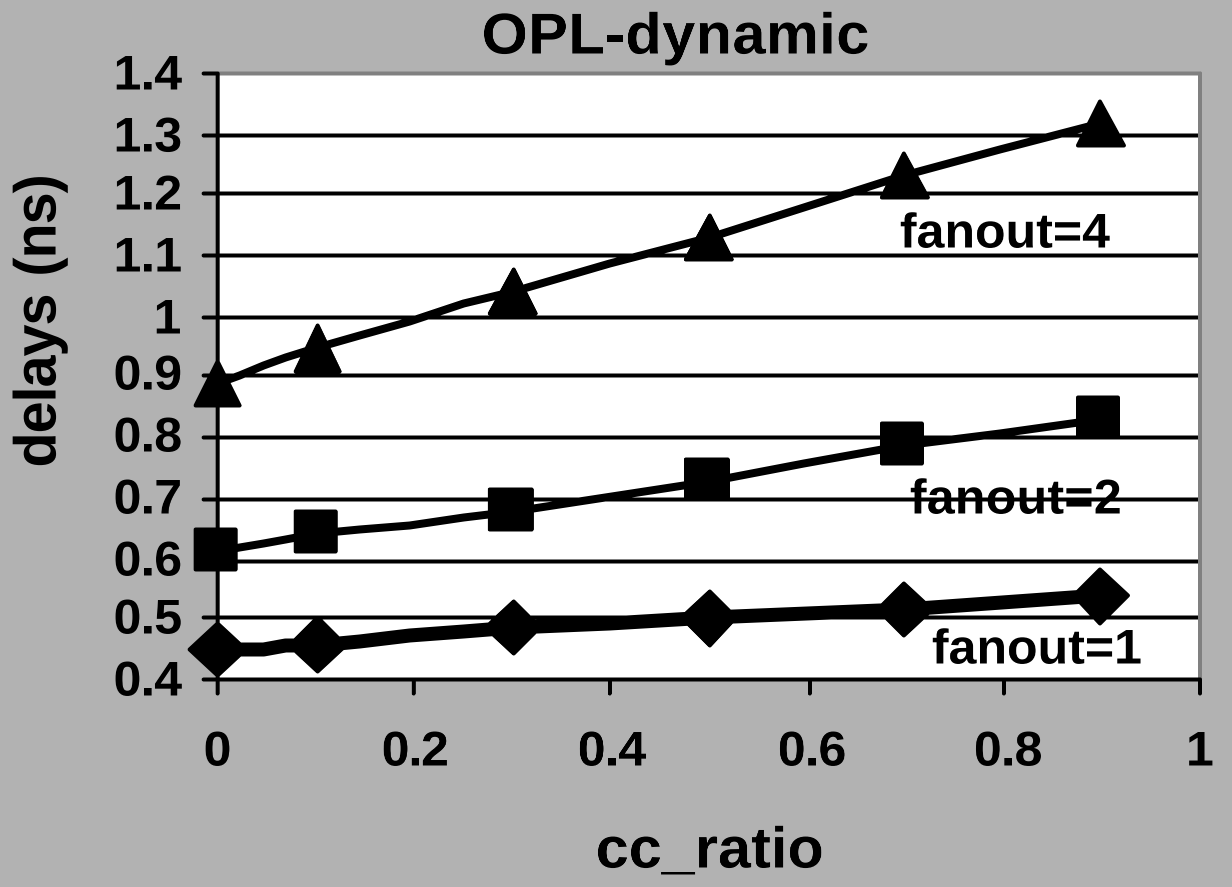
Long Wire With Noise Injection



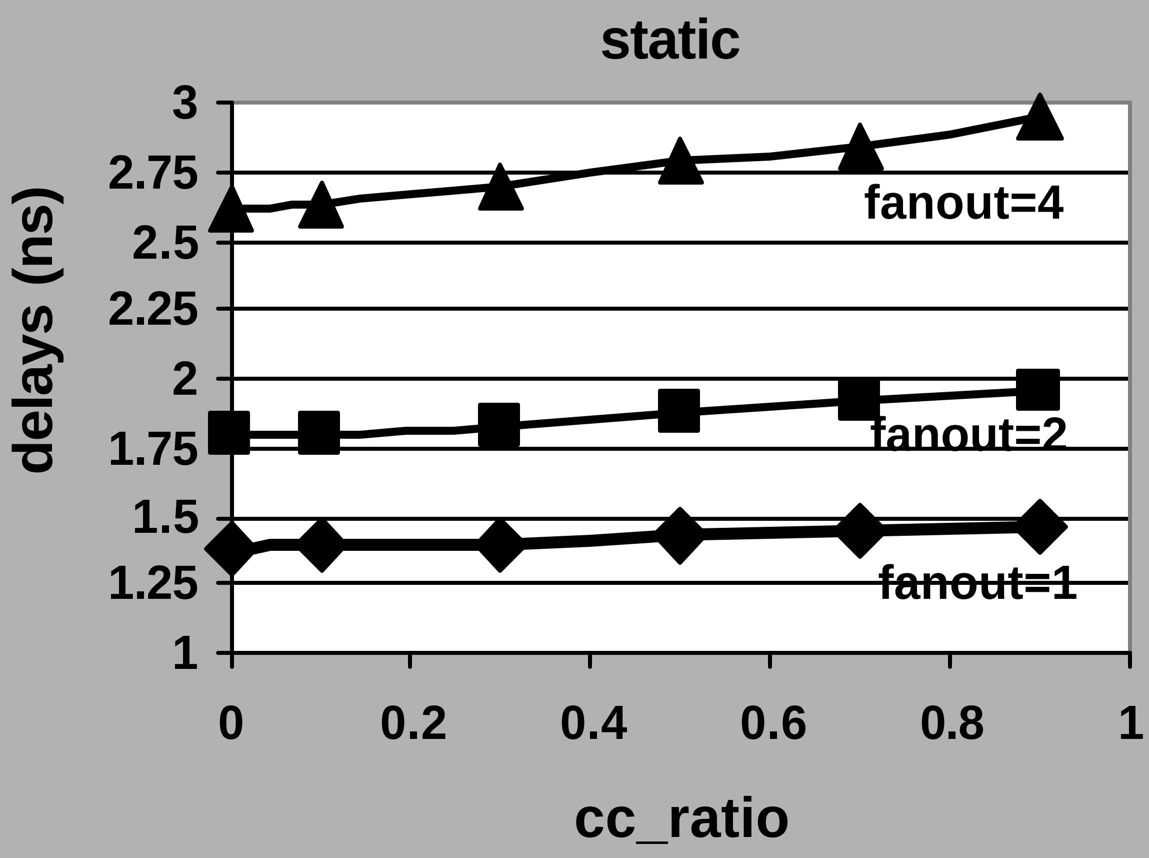
Delays of OPL-diff. AOI22 Chains including Coupling Noise.



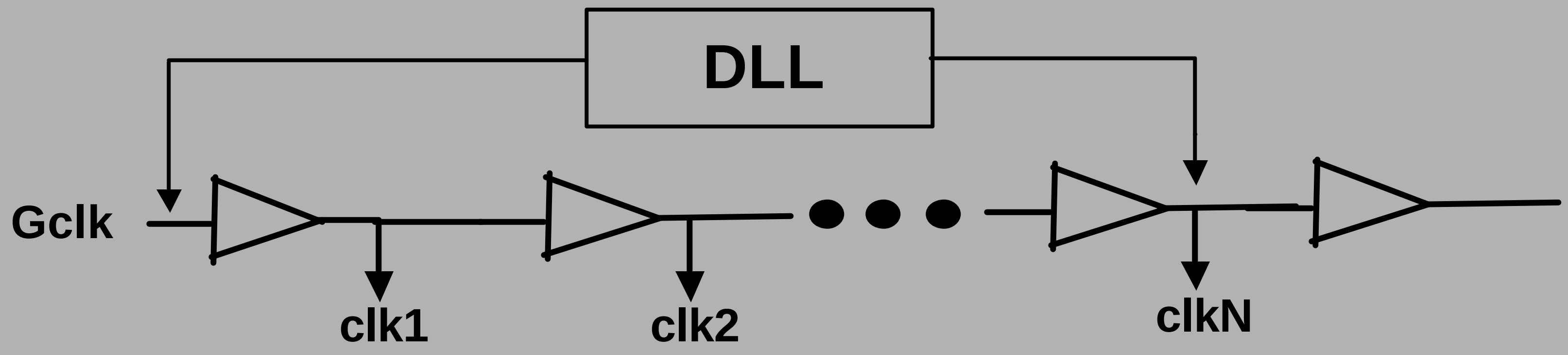
Delays of OPL-dyn. AOI22 Chains including Coupling Noise



Delays of Static CMOS AOI22 Chains including Coupling Noise

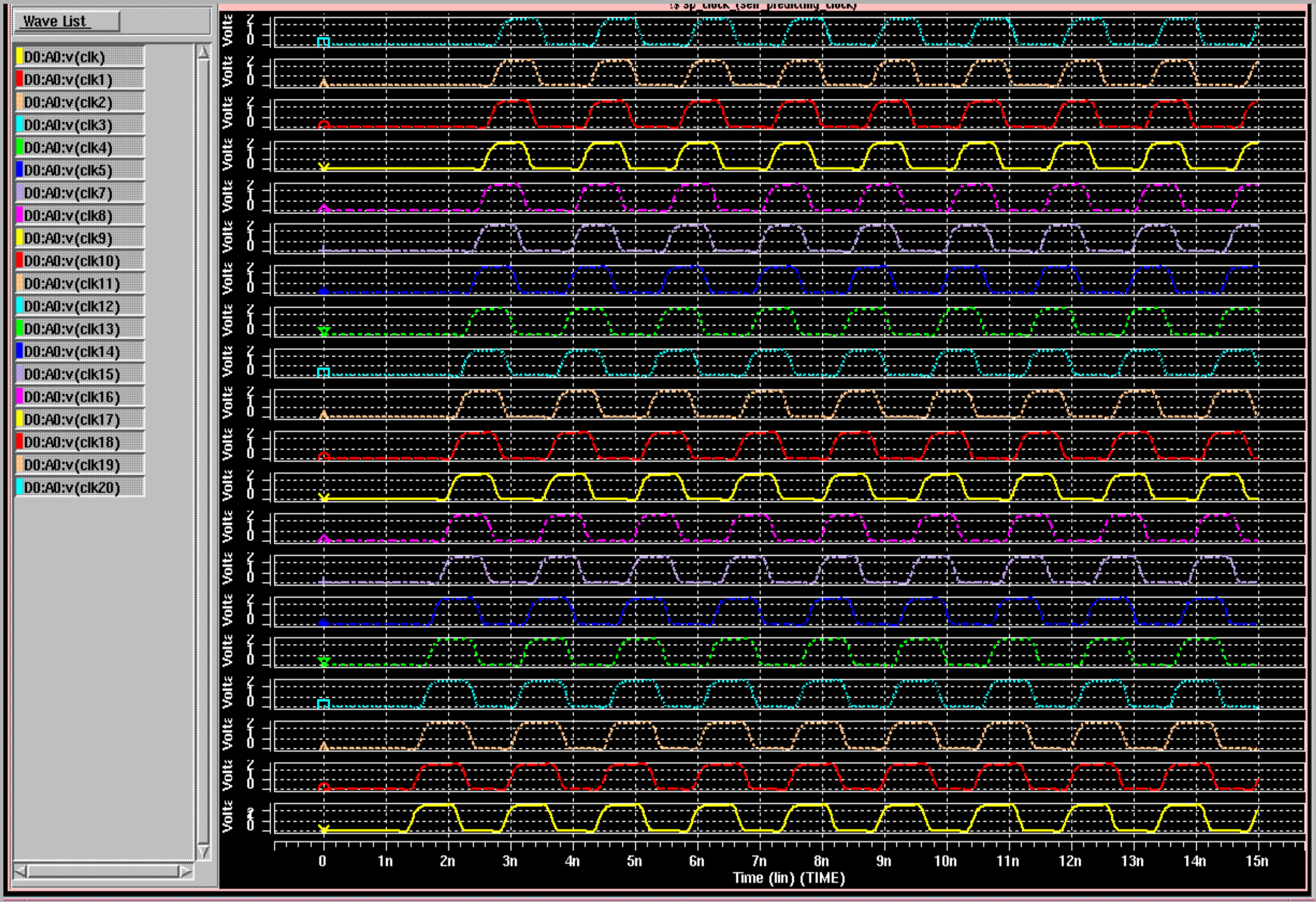


Clock Generation



- A new technique enables the design of a buffer having a delay roughly equal to a FO1 (static) inverter delay!
- Utilizes a novel DLL design
- Given 40% random variations in L, an average clock separation of 75 ps can be achieved, plus or minus 20 ps
  - FO4 for this 0.25 um process is 164 ps

Simulation Results for Clock Scheme



Ultra High Speed Adder Design

- Use carry look-ahead since it makes effective use of NOR gates

$$\overline{C_1} ? \overline{\overline{\overline{g_0}} ? \overline{\overline{p_0} ? \overline{C_{in}}}}$$

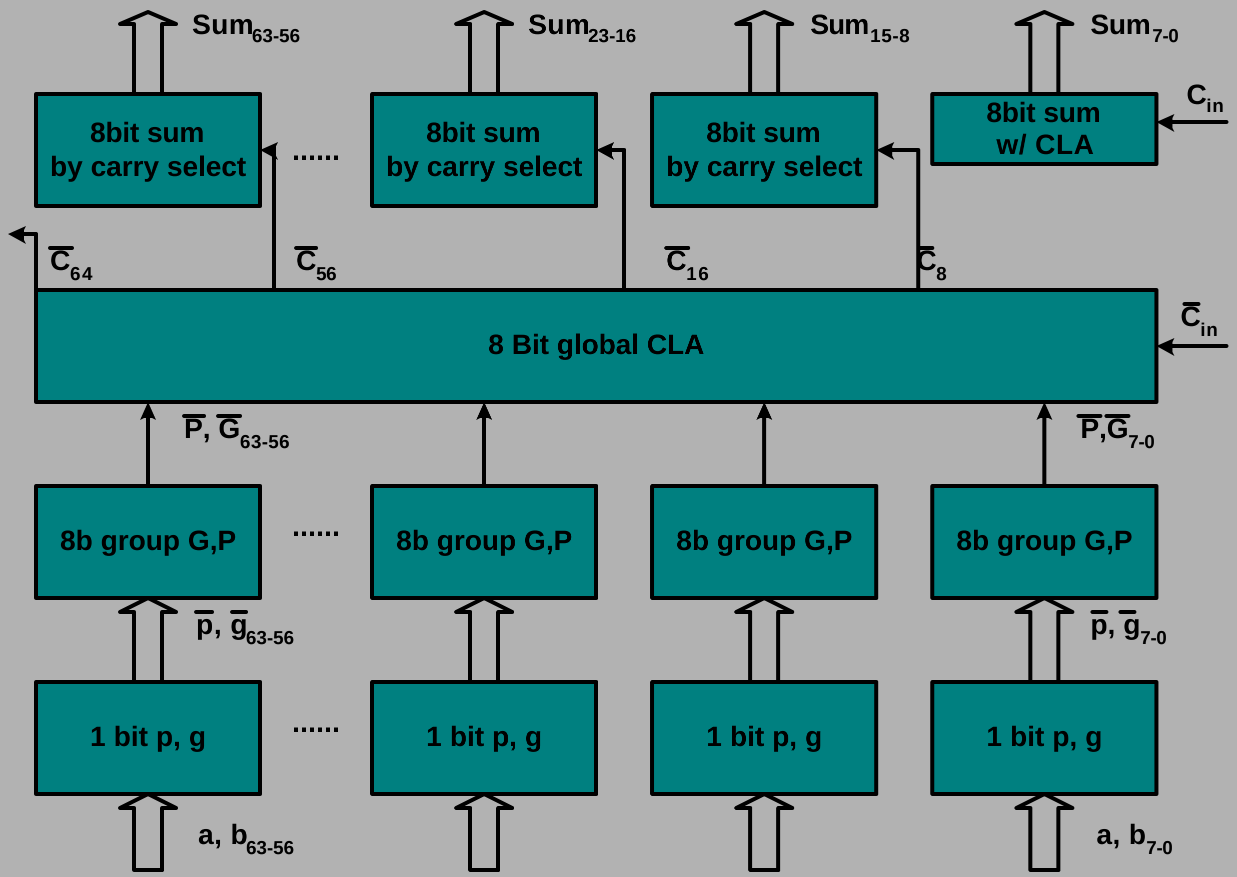
$$\overline{C_2} ? \overline{\overline{\overline{g_1}} ? \overline{\overline{p_1} ? \overline{g_0}} ? \overline{\overline{p_1} ? \overline{p_0} ? \overline{C_{in}}}}$$

$$\overline{C_3} ? \overline{\overline{\overline{g_2}} ? \overline{\overline{p_2} ? \overline{g_1}} ? \overline{\overline{p_2} ? \overline{p_1} ? \overline{g_0}} ? \overline{\overline{p_2} ? \overline{p_1} ? \overline{p_0} ? \overline{C_{in}}}}$$

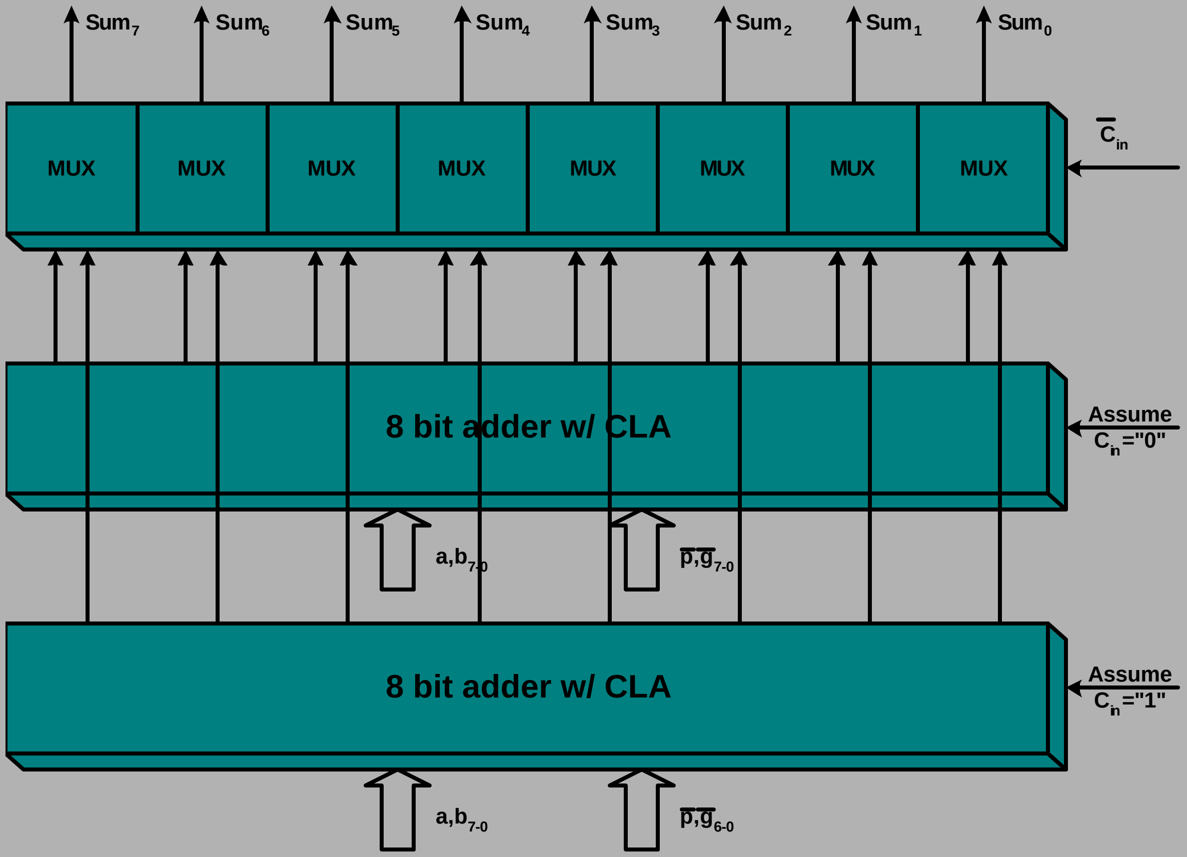
......

$$\overline{C_7} ? \overline{\overline{\overline{g_6}} ? \overline{\overline{p_6} ? \overline{g_5}} ? \text{......}\overline{\overline{p_6} ? \overline{p_5} ? \overline{p_4} ? \overline{p_3} ? \overline{p_2} ? \overline{p_1} ? \overline{p_0} ? \overline{C_{in}}}}$$

64-bit Adder Architecture



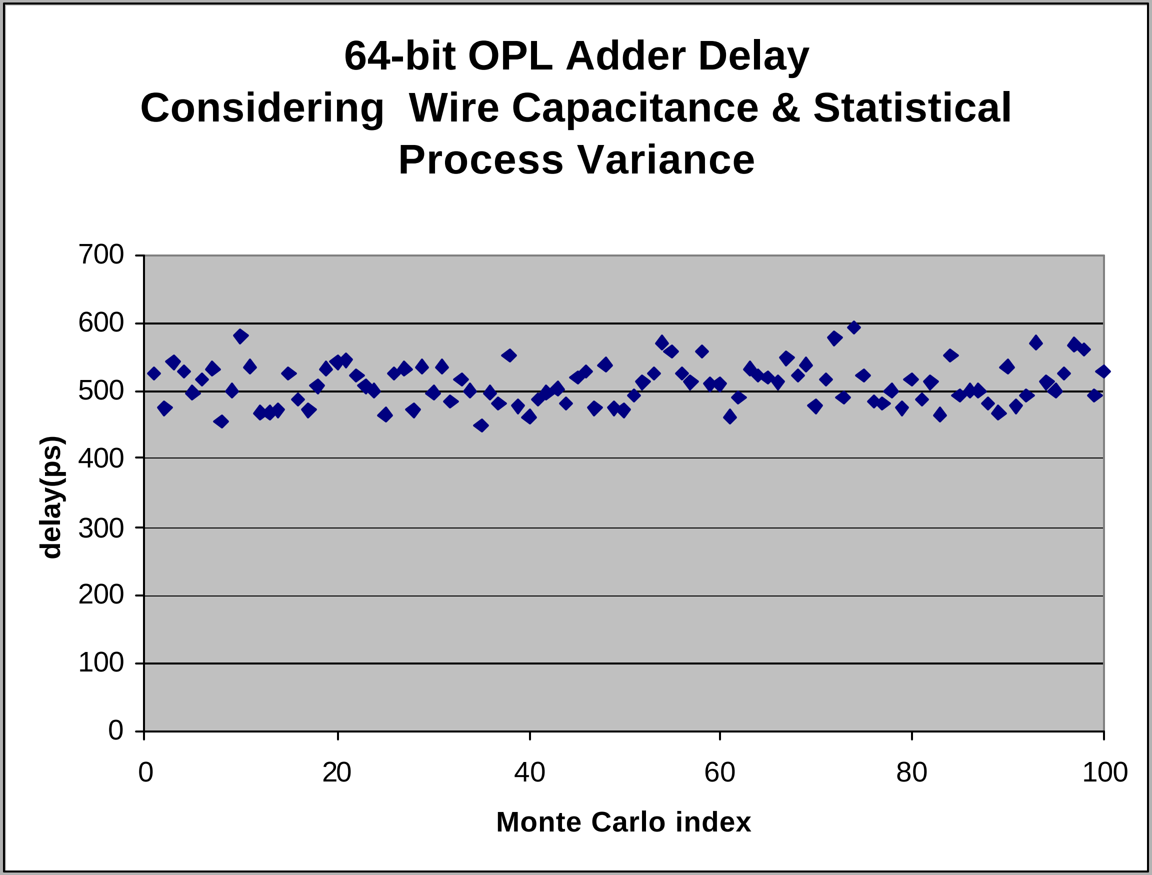
8-Bit Sum using Carry Select



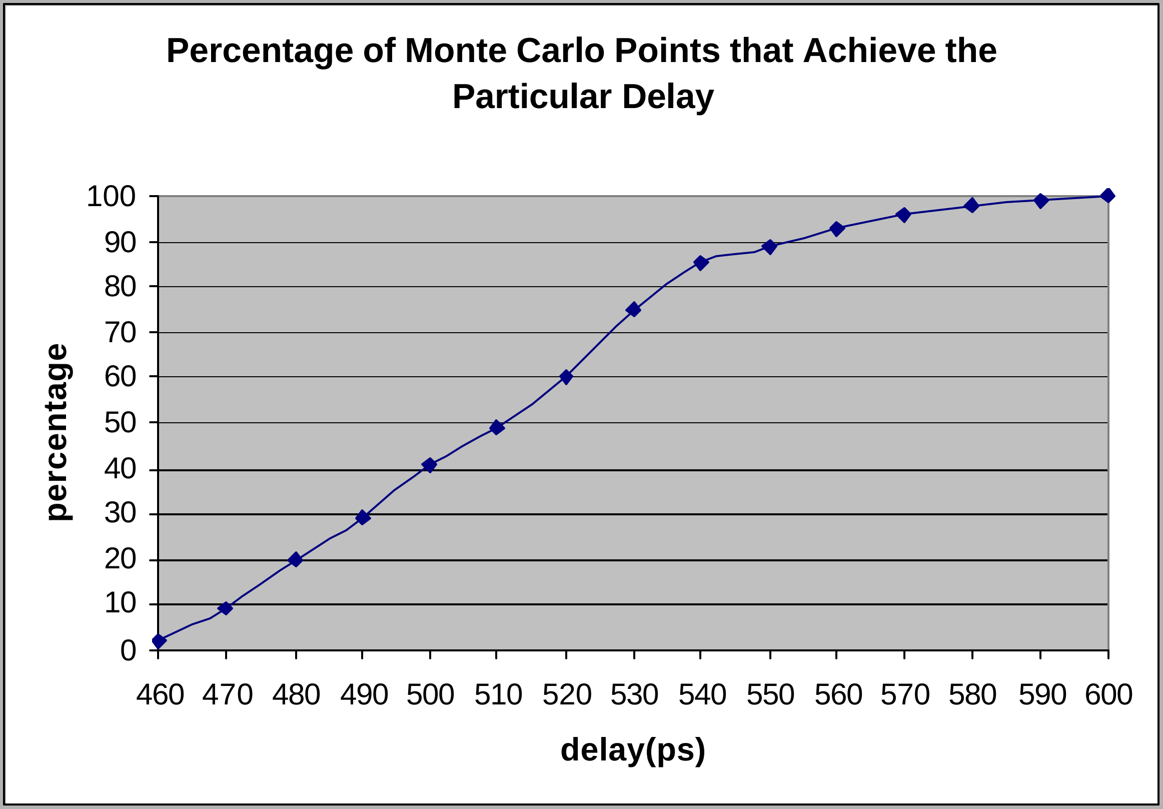
64-Bit Adder Results

64-bit Adder	Process	Delay	Divided by FO4 Inv Delay
OPL	.25? m	460 ps	2.8
David Harris, Stanford	.6? m		6.4
S.Naffziger, HP	.5? m	930 ps	7.0

64-bit Adder: Statistical Analysis



64-bit Adder: Statistical Analysis



OPL Summary

- OPL appears to be fastest known logic technique
  - Patent application has been filed
- Applicable to: static CMOS, pseudo-nMOS, or dynamic logic
- Speeds up underlying logic family by at least 2X
- Developed new, yet faster logic technique – OPL-differential logic (may apply for additional patent)
- OPL 64-bit adder: worst-case delay of 2.8 FO4 INV delays (best previously reported: 6.4 – 7.0)
- Very applicable to random logic blocks
- Analyzed OPL’s performance with respect to PVT variations and coupling noise
- Developed reliable clock generation scheme for OPL circuits
- Intel has a team working on OPL circuit development
  - OPL verified for a sub-100nm process