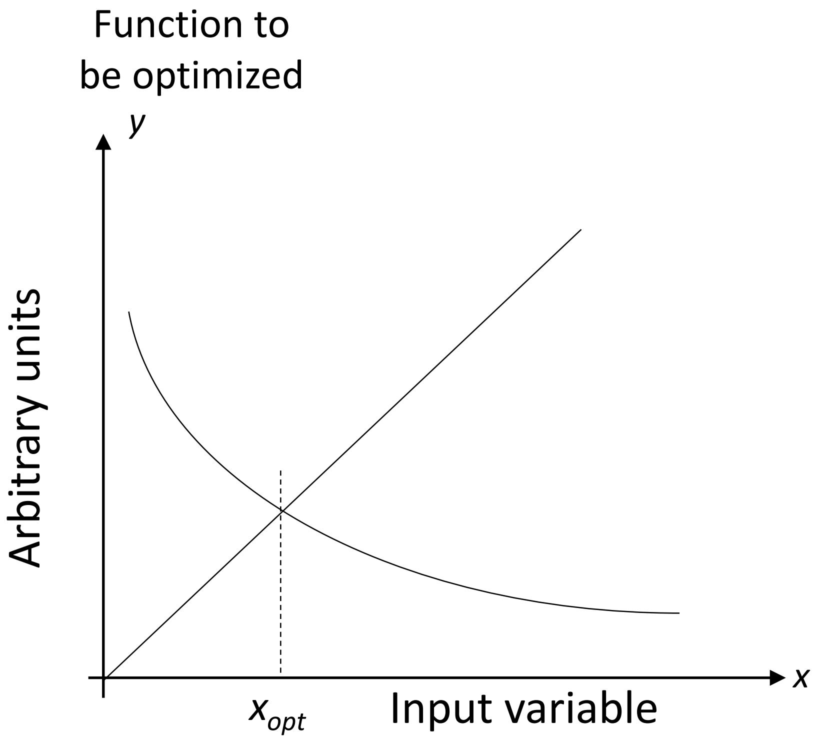
Lecture 15

Introduction to Integrated Circuit Design

What if . . . optimizations

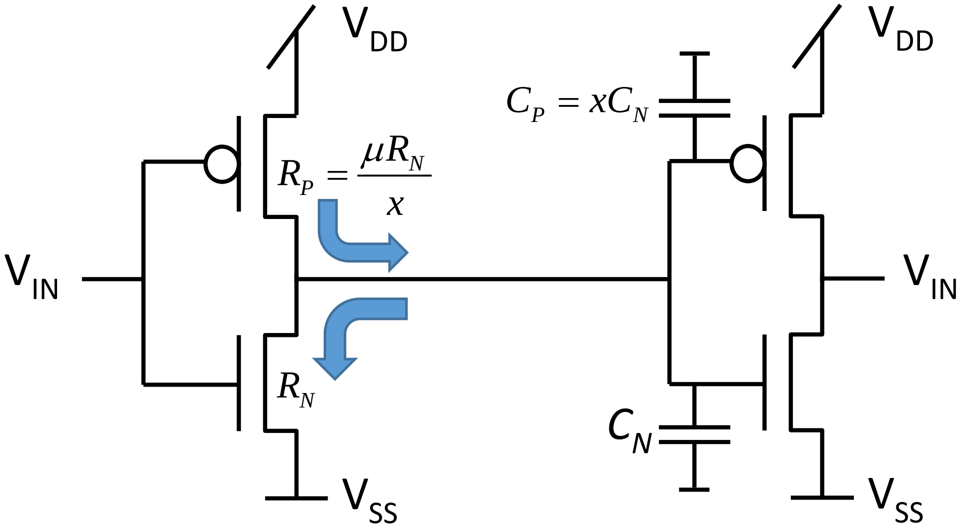
Optimization – Introduction



Inverter pair delay

Question: What if we want to minimize the inverter pair delay, how should we choose the width of the p-channel device wrt the width of the n-channel device - considering the difference in electron and hole mobility?

Answer: Assume R_P=μR_N for the same channel widths. Typically μ=2



Rise delay:

$$R_P\left(C_N + C_P\right) = R_N C_N \left(1 + x\right)\frac{\mu}{x}$$

Fall delay:

$$R_N\left(C_N + C_P\right) = R_N C_N \left(1 + x\right)$$

Normalized pair delay :

$$d = \left(1 + \frac{\mu}{x}\right)\left(1 + x\right) = 1 + \mu + x + \frac{\mu}{x}$$

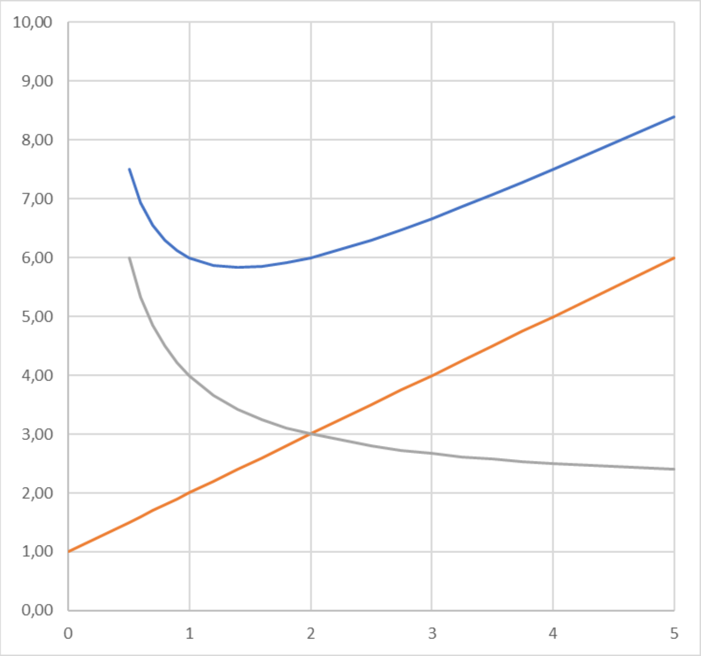
Minimum pair delay for:

$$x = \sqrt{\mu} = \sqrt{2}$$

Inverter pair delay

Question: What if we want to minimize the inverter pair delay, how should we choose the width of the p-channel device wrt the width of the n-channel device - considering the difference in electron and hole mobility?

Answer: Assume $R_P = \mu R_N$ for the same channel widths. Typically $\mu = 2$



Rise delay:

$$R_P\left(C_N + C_P\right) = R_N C_N \left(1 + x\right)\frac{\mu}{x}$$

Fall delay:

$$R_N\left(C_N + C_P\right) = R_N C_N \left(1 + x\right)$$

Normalized pair delay :

$$d = \left(1 + \frac{\mu}{x}\right)\left(1 + x\right) = 1 + \mu + x + \frac{\mu}{x}$$

Minimum pair delay for:

$$x = \sqrt{\mu} = \sqrt{2}$$

Optimization – Energy*delay product

Question: What if we want to minimize the energy-delay product. How should we choose V_{DD} wrt to V_T?

Answer: It should be three times larger, i.e. $V_{DD} = 3V_T$

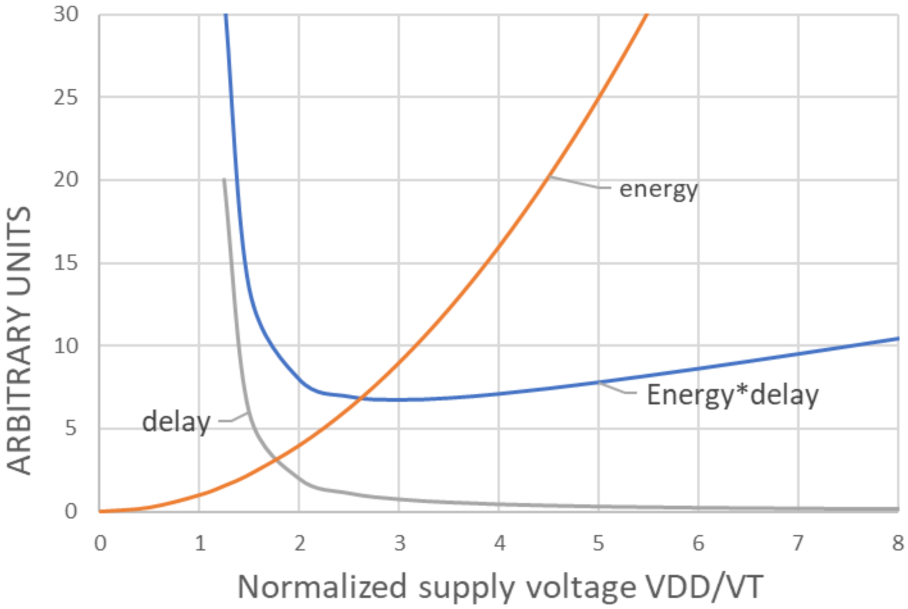
Energy: $E = CV_{DD}^2$

Delay: $t_{pd} \sim \frac{Q}{I_{DSAT}} \sim \frac{V_{DD}}{(V_{DD} - V_T)^2}$

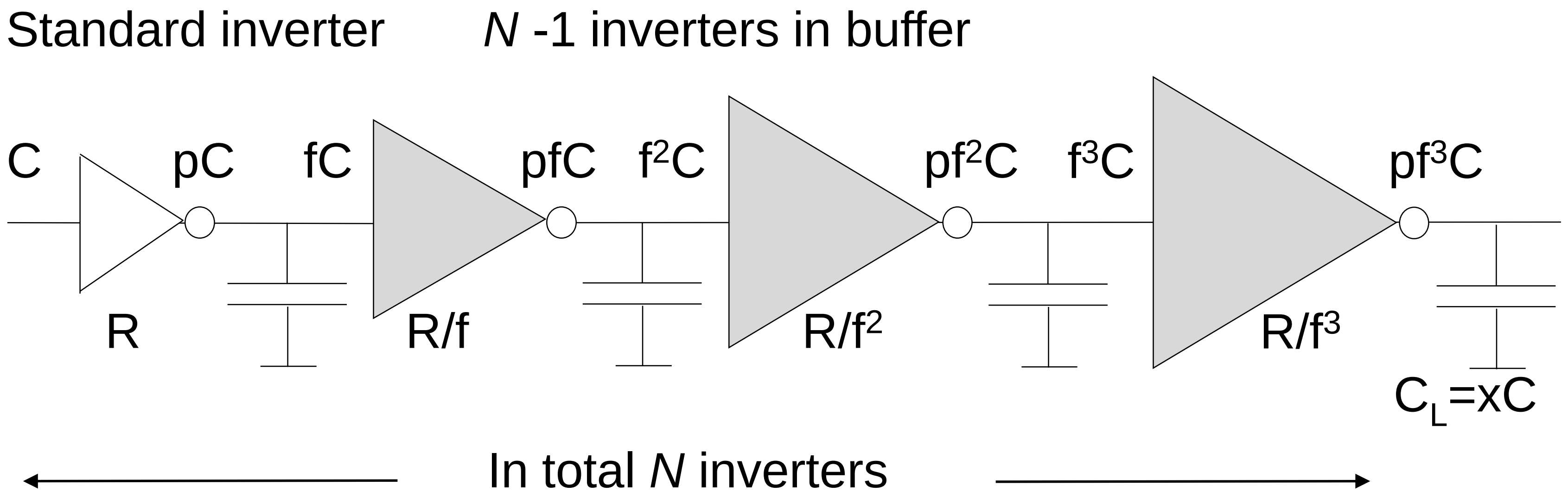
Energy*delay product

$$Et_{pd} \sim \frac{V_{DD}^3}{(V_{DD} - V_T)^2}$$

has a minimum for $V_{DD} = 3V_T$



Optimal tapering factor



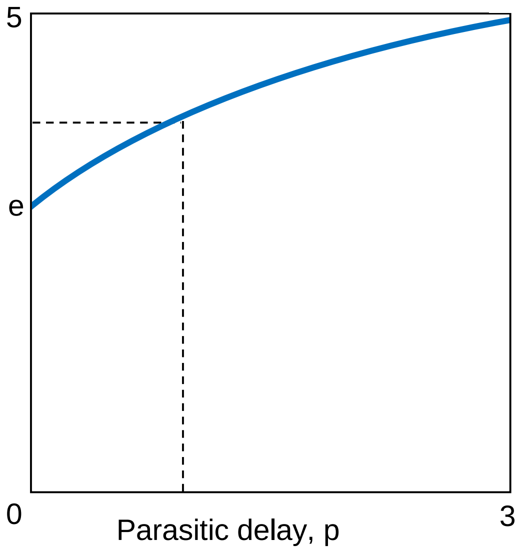
Relative delay=number of stages * stage delay: $d = N\left(p + f\right)$

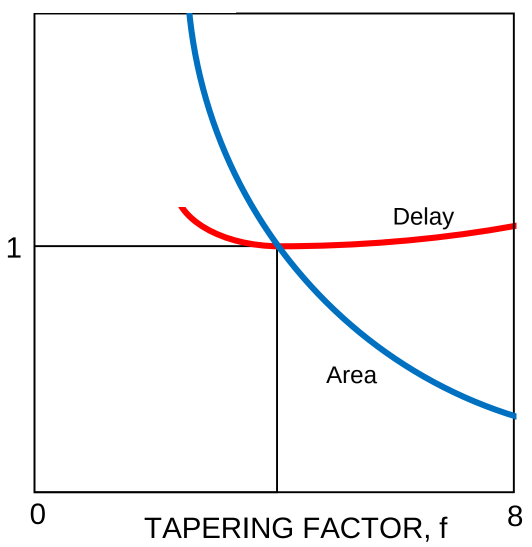
Minimum delay when all inverters have the same fanout: $f = \sqrt[N]{x}$

Number of stages: $N = \ln x / \ln f$

Relative delay: $d = \left(p + f\right)\ln x / \ln f$

Optimal tapering factor





For typical values of p, the optimum tapering factor is between 3.6 and 5. Typically a FO4!

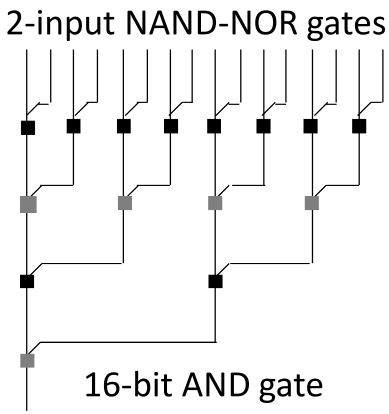
- A buffer with N-1 extra inverters consume an area $A_0 \approx f + f^2 + ... + f^{N-1}$
- **By choosing a somewhat larger tapering factor we can easily save more than 50% in area while only losing 10% in speed!**

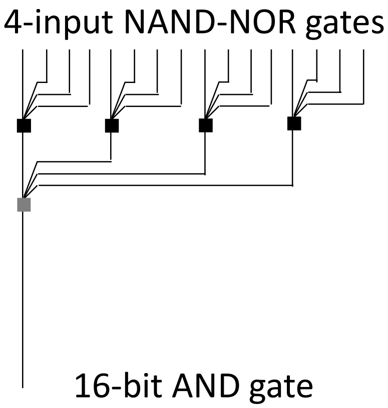
[From Hedenstierna & Jeppson 1987]

Optimization – Number of inputs

Question: What if we were to build a 16-bit AND gate using *n*-input NAND/ NOR-gate combinations, what would be the most efficient number of gate inputs to minimize the delay? (Reminder: $\overline{\overline{ab}+\overline{cd}} = abcd.$)

Answer: Is it with 2- or 4-input gates? Or with 8-input gates?





Optimization – Number of inputs

Question: What if we were to build a 16-bit AND gate using *n*-input NAND/ NOR-gate combinations, what would be the most efficient number of gate inputs to minimize the delay? (Reminder: $\overline{\overline{ab}+\overline{cd}} = abcd.$)

Answer: Is it with 2- or 4-input gates? Or with 8-input gates?

The NAND/NOR pair delay can be written d_{pair}=3*m*+1.

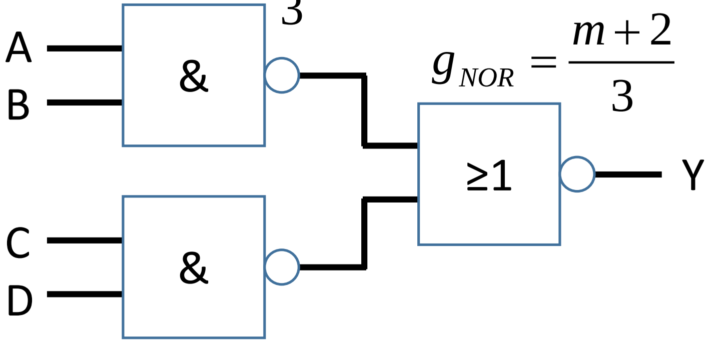
With a logical depth, *N*, given by n^N=16, the total normalized delay can be written

$$d_{AND16} = \frac{3m+1}{2} \times \frac{\ln 16}{\ln m}$$

which has a minimum for $m = \dfrac{3m+1}{3\ln m} = 3.$

$$p_{NAND} = m \qquad g_{NAND} = \frac{2m+1}{3}$$

$$p_{NOR} = m \qquad g_{NOR} = \frac{m+2}{3}$$



However, the number of inputs must in this case be a multiple of 2, so our choice is to use 2- or 4-input gates; both resulting in about the same delay, with a minor advantage for the 4-input solution.

Optimization – Number of inputs

Question: What if we were to build a 16-bit AND gate using *n*-input NAND/ NOR-gate combinations, what would be the most efficient number of gate inputs to minimize the delay? (Reminder: $\overline{\overline{ab}+\overline{cd}}=abcd$.)

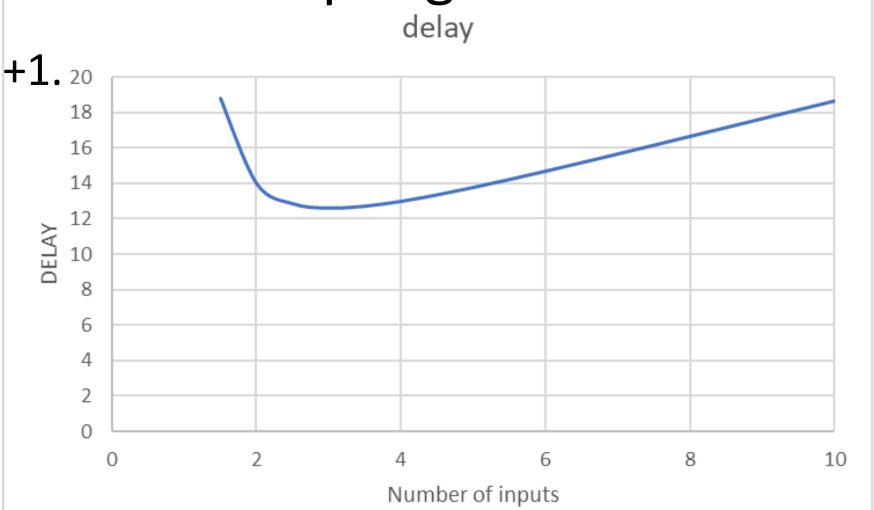
Answer: Is it with 2- or 4-input gates? Or with 8-input gates?

The NAND/NOR pair delay can be written d_{pair}=3m+1.

With a logical depth, *N*, given by n^N=16, the total normalized delay can be written

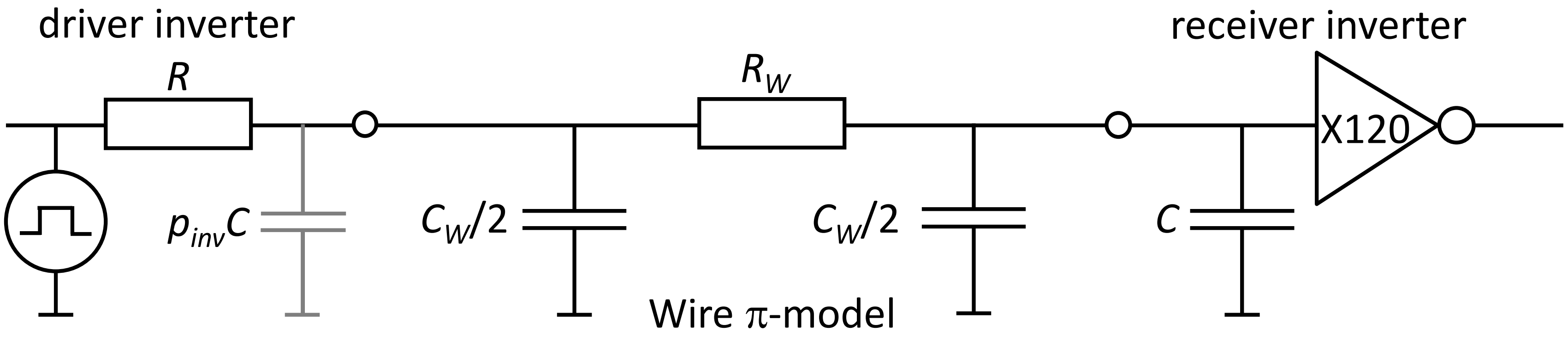
$$d_{AND16}=\frac{3m+1}{2}\times\frac{\ln 16}{\ln m}$$

which has a minimum for $m=\frac{3m+1}{3\ln m}=3.$



However, the number of inputs must in this case be a multiple of 2, so our choice is to use 2- or 4-input gates; both resulting in about the same delay, with a minor advantage for the 4-input solution.

Driving long wires



1. Start by defining wire effort $W_E = \dfrac{R_W C_W}{RC}$

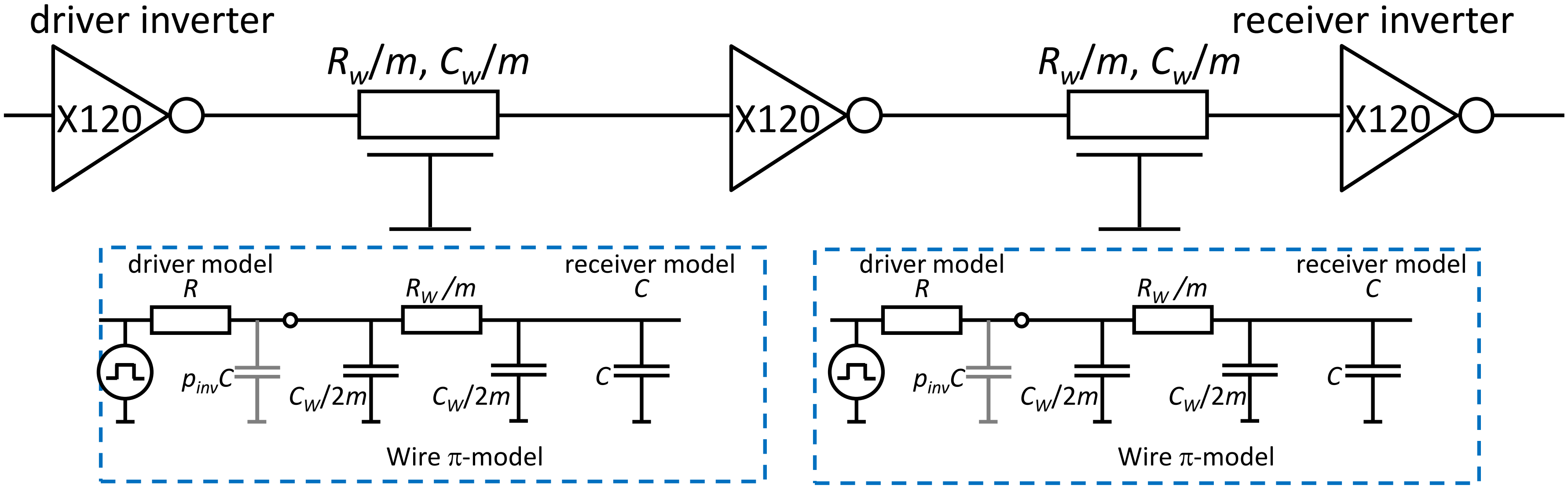
2. Replace wire with wire RC π-model!

3. Consider critical wire length for repeater insertion!

$$L_{crit} = \frac{2L}{\sqrt{W_E}} = 2\sqrt{\frac{RC}{rc}}$$

where *r* and *c* are wire resistance and capacitance per unit length!

Repeater insertion

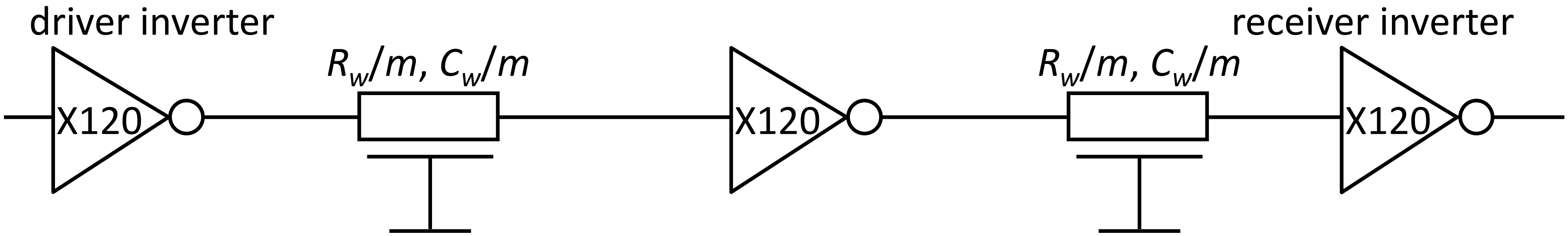


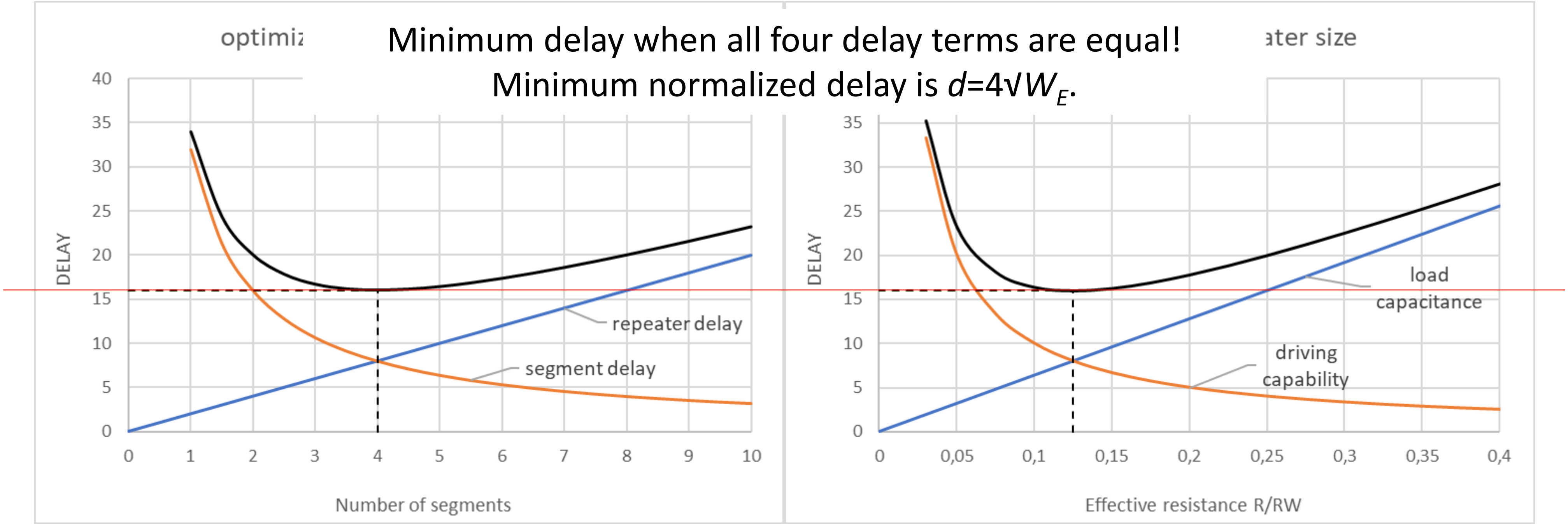
Normalized delay *m* segments $d = m\left(1 + p_{inv}\right) + W_E \dfrac{R}{R_W} + \dfrac{R_W}{R} + \dfrac{W_E}{2m}$

Find minimum $\dfrac{\partial d}{\partial R} = \dfrac{W_E}{R_W} - \dfrac{R_W}{R^2} = 0 \;\rightarrow\; R = \dfrac{R_W}{\sqrt{W_E}}$

Find minimum $\dfrac{\partial d}{\partial m} = \left(1 + p_{inv}\right) - \dfrac{W_E}{2m^2} = 0 \;\rightarrow\; m = \sqrt{\dfrac{W_E}{2\left(p_{inv} + 1\right)}} \approx \tfrac{1}{2}\sqrt{W_E}$

Repeater insertion



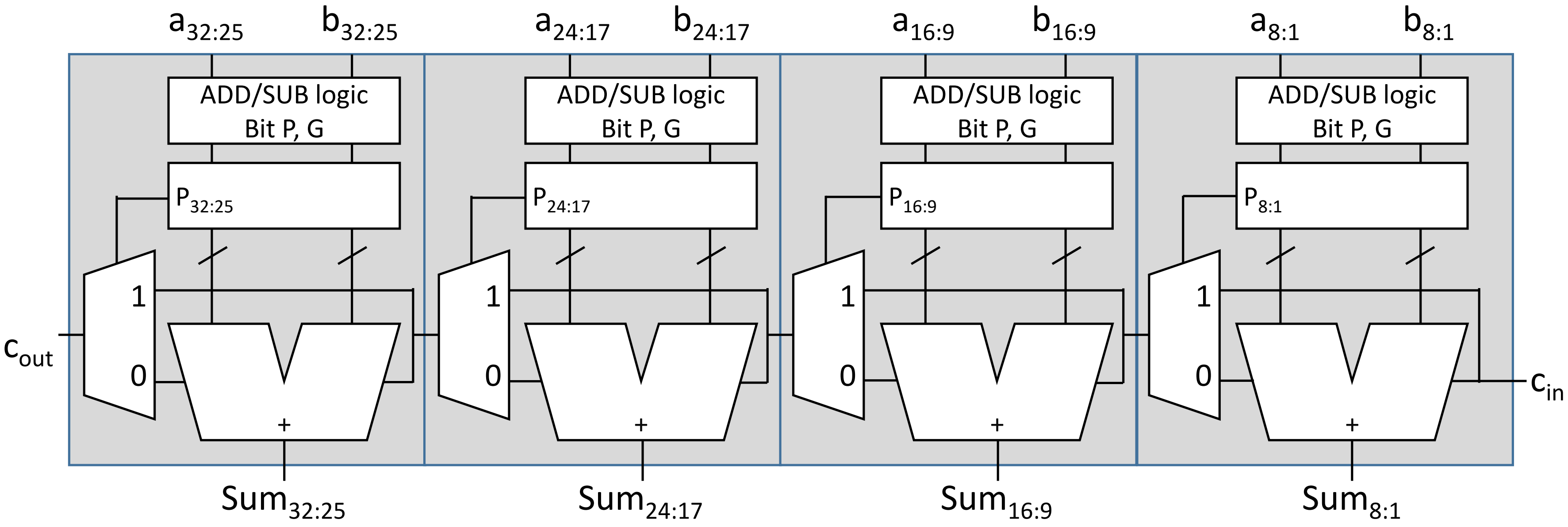


Minimum delay when all four delay terms are equal!

Minimum normalized delay is $d=4\sqrt{W_E}$.

32-bit carry skip adder

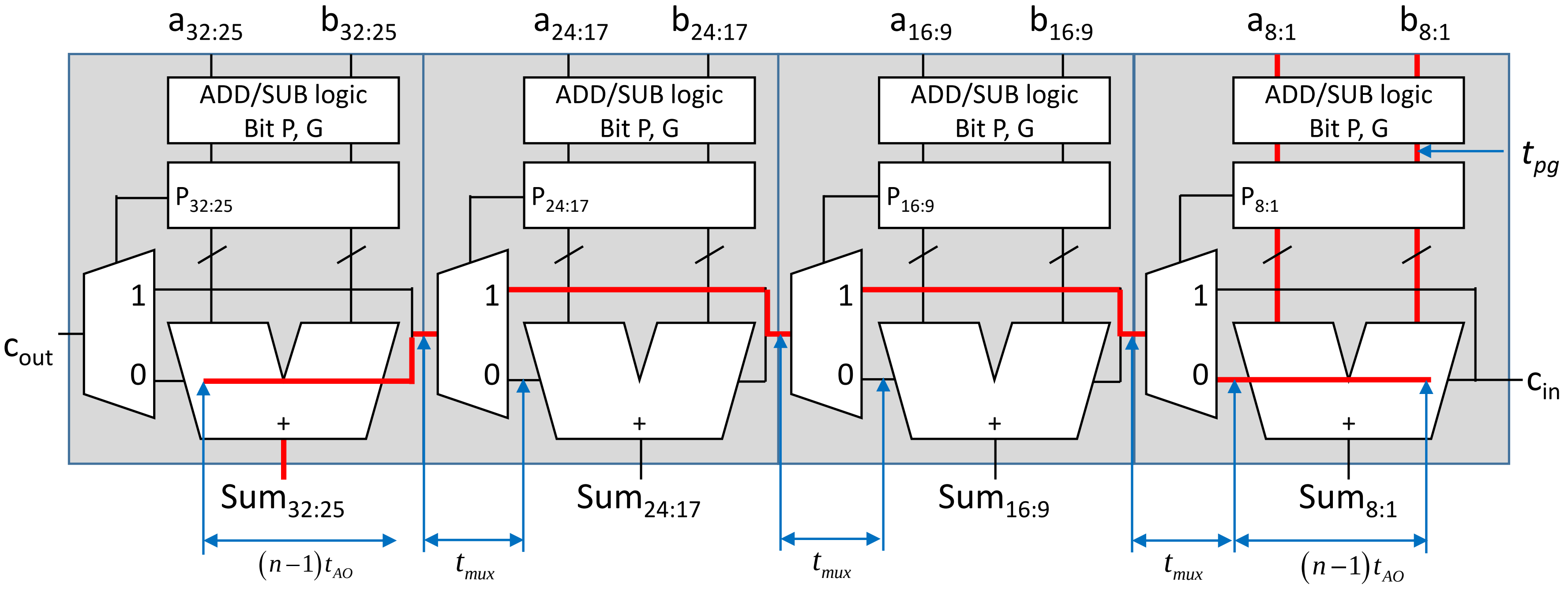
- Identify worst-case propagation delay for 32-bit adder!
- How to optimize a 32-bit adder built with *k* *n*-bit blocks?



The worst case delay, t_{skip}, is for calculating the most significant bit sum when a carry is generated in the least significant bit.

32-bit carry skip adder

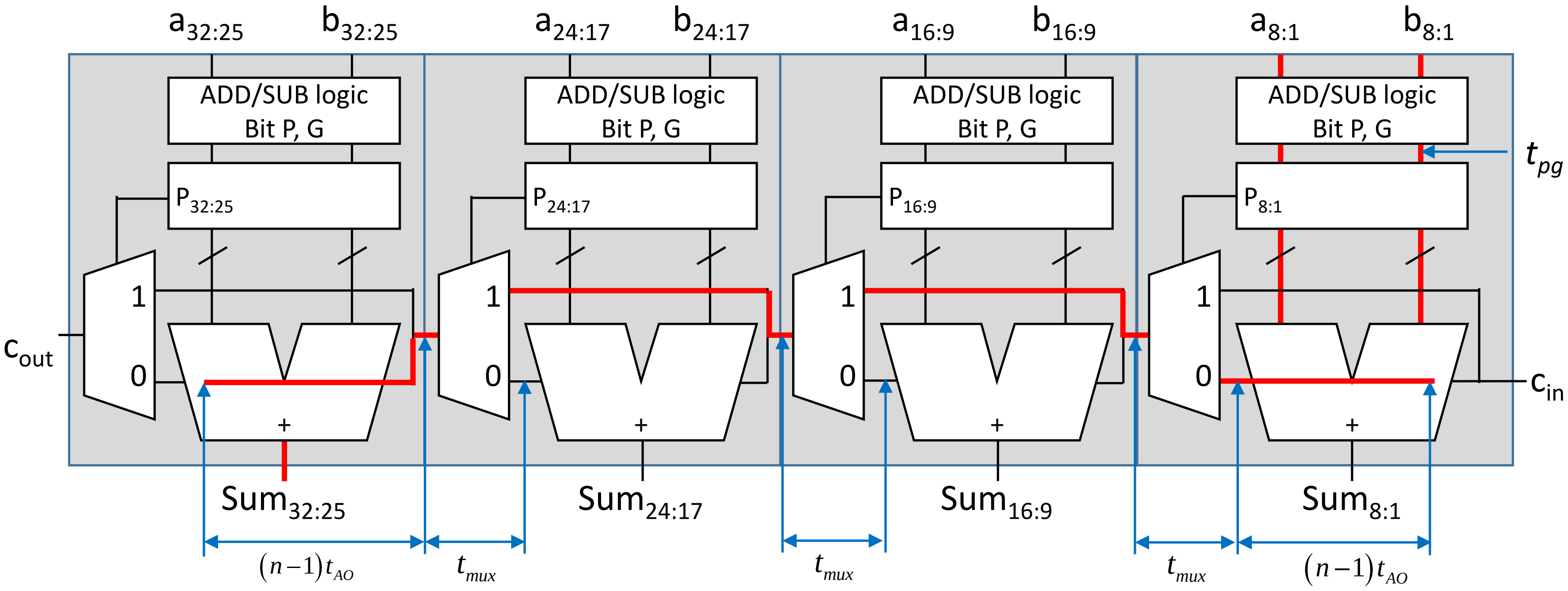
- Identify worst-case propagation delay for 32-bit adder!
- How to optimize a 32-bit adder built with *k* *n*-bit blocks?



The worst case delay, t_{skip}, is for calculating the most significant bit sum when a carry is generated in the least significant bit.

32-bit carry skip adder

- Identify worst-case propagation delay for 32-bit adder!
- How to optimize a 32-bit adder built with *k* *n*-bit blocks?

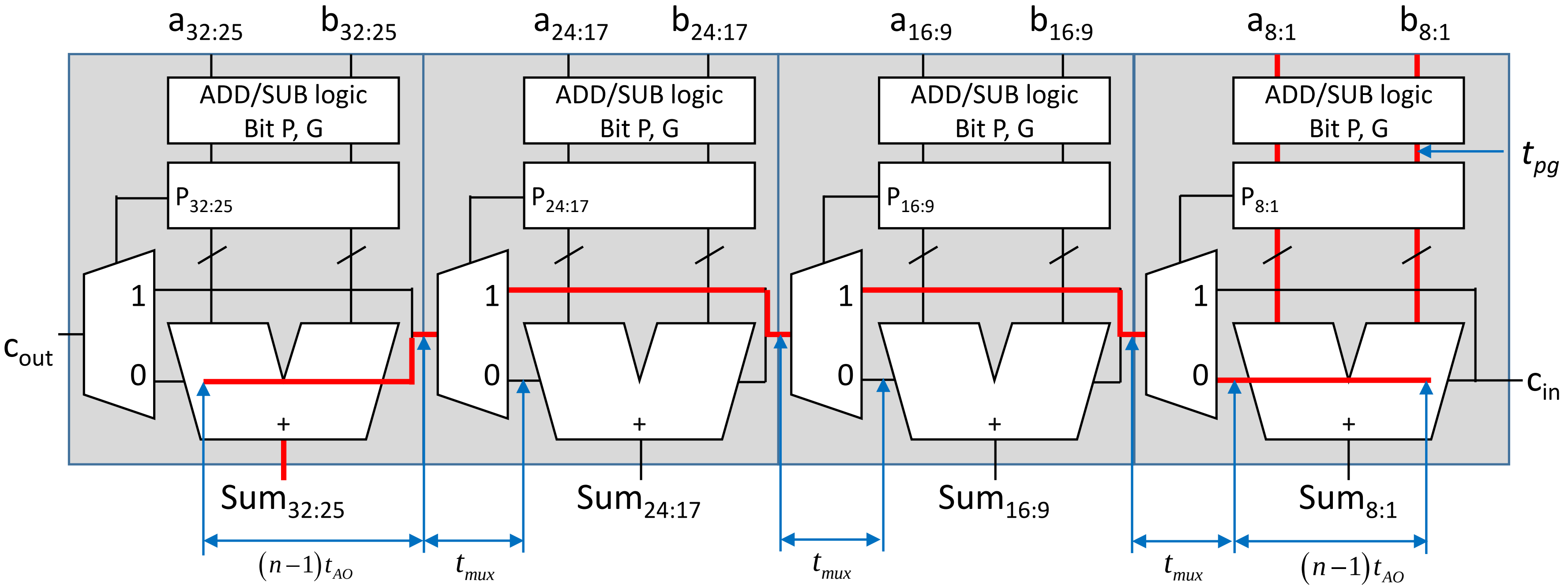


$$t_{skip} = t_{pg} + 2(n-1)t_{AO} + (k-1)t_{mux} + t_{XOR}$$

$$t_{skip} = t_{pg} + 2(n-1)t_{AO} + (32/n-1)t_{mux} + t_{XOR}$$

32-bit carry skip adder

- Identify worst-case propagation delay for 32-bit adder!
- How to optimize a 32-bit adder built with *k* *n*-bit blocks?



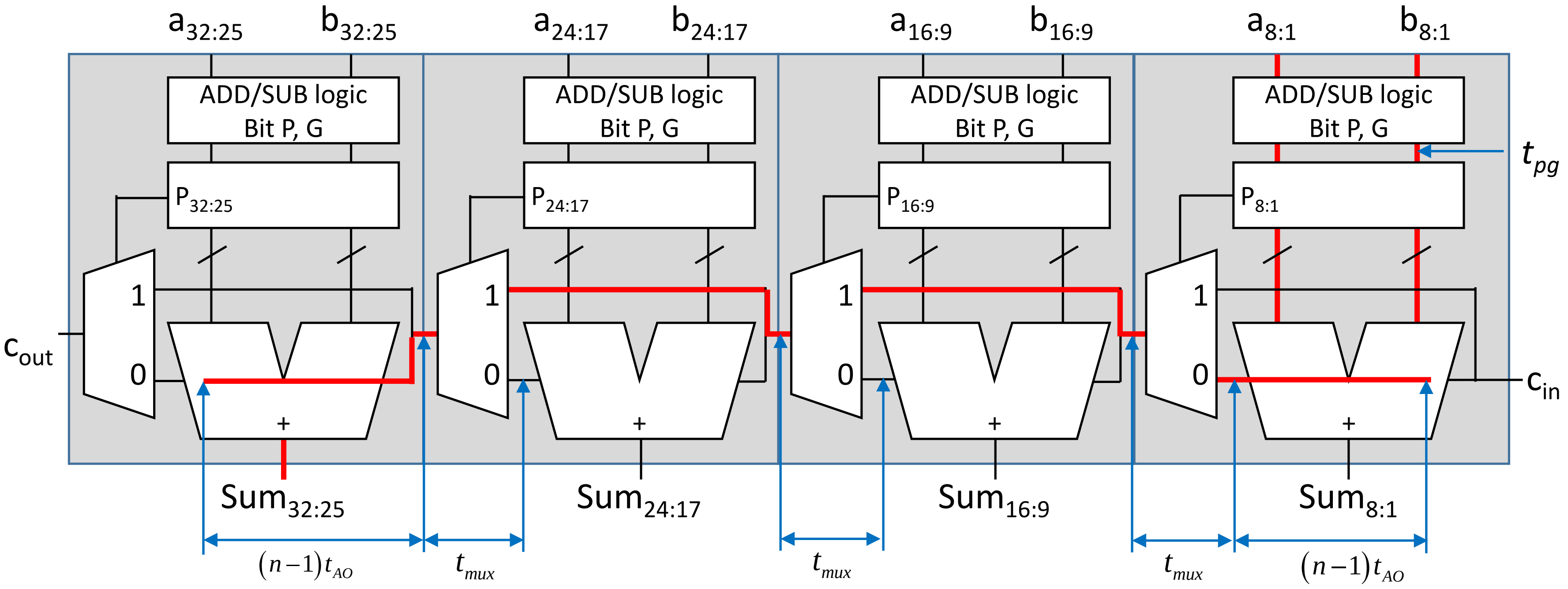
$$t_{skip} = t_{pg} + 2(n-1)t_{AO} + (k-1)t_{mux} + t_{XOR}$$

$$t_{skip} = t_{pg} + 2(n-1)t_{AO} + (32/n-1)t_{mux} + t_{XOR}$$

$$n_{opt} = 4\sqrt{\frac{t_{MUX}}{t_{AO}}}.$$

32-bit carry skip adder

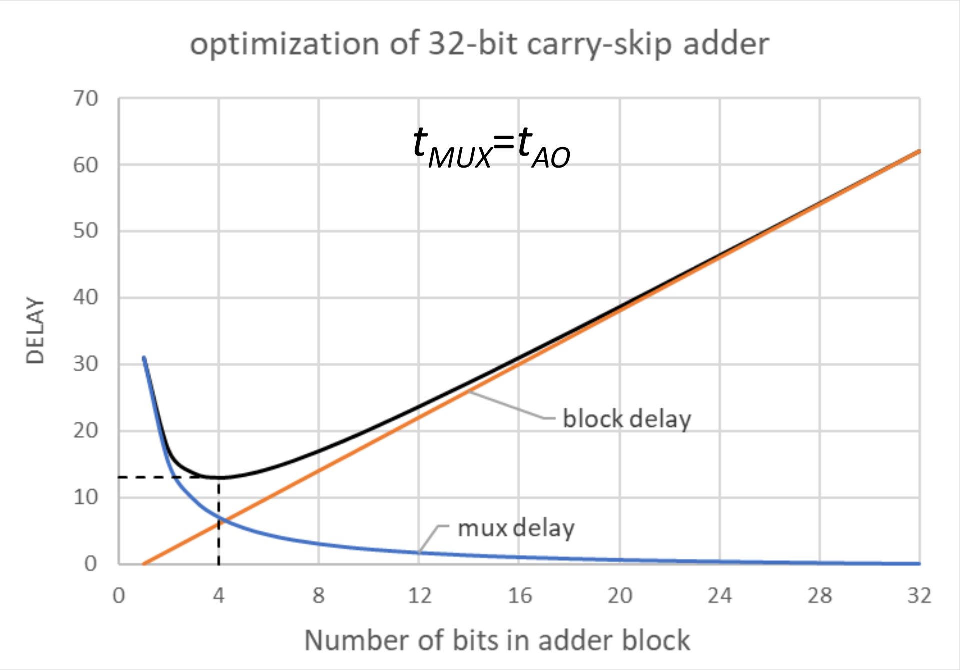
- Identify worst-case propagation delay for 32-bit adder!
- How to optimize a 32-bit adder built with *k* *n*-bit blocks?

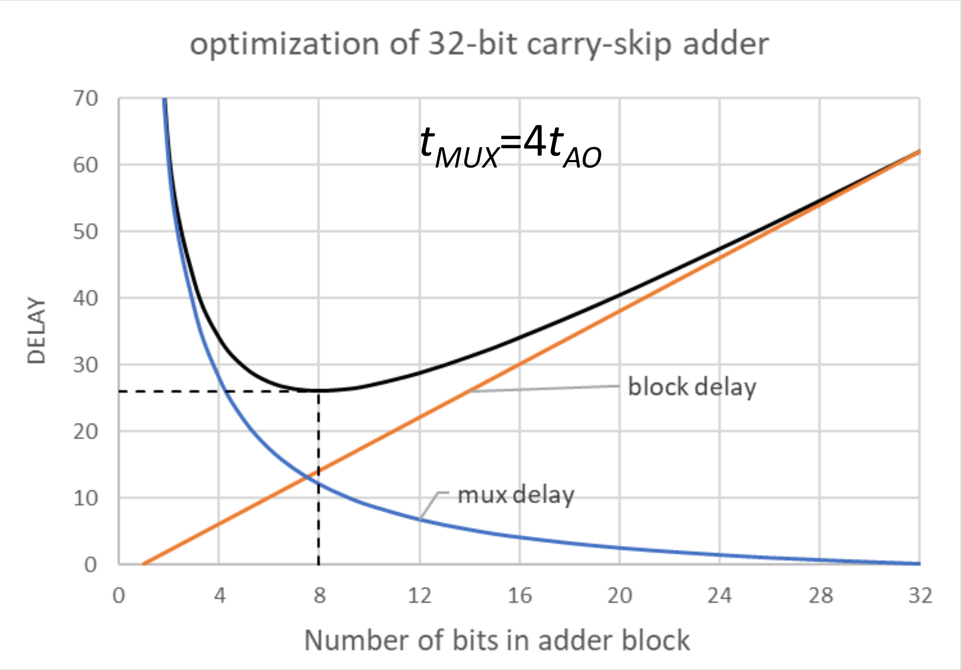


If efficient muxes with t_{MUX}=t_{AO} could be used, 4-bit blocks would be most efficient.
If muxes are slow due to their complexity, say t_{MUX}=4t_{AO}, then 8-bit blocks would be most efficient.

32-bit carry skip adder

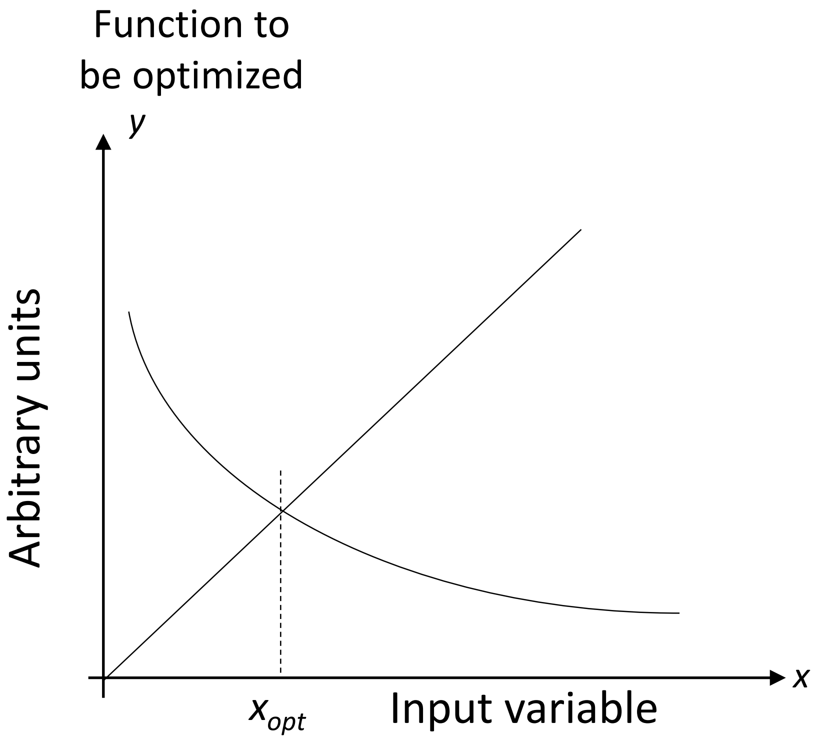
- Identify worst-case propagation delay for 32-bit adder!
- How to optimize a 32-bit adder built with *k* *n*-bit blocks?





If efficient muxes with $t_{MUX}=t_{AO}$ could be used, 4-bit blocks would be most efficient.
If muxes are slow due to their complexity, say $t_{MUX}=4t_{AO}$, then 8-bit blocks would be most efficient.

Optimization – Summary



End of ”what if” lecture!

Q & A?