

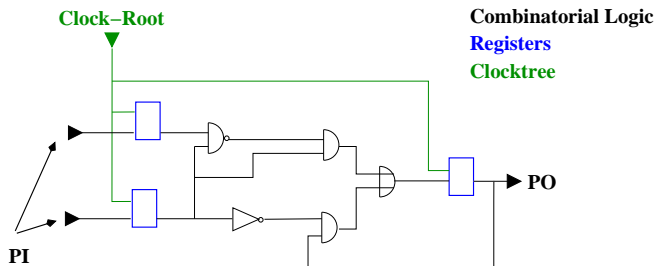
Logic Optimization in VLSI Design

Dieter Rautenbach

Forschungsinstitut für Diskrete Mathematik
Universität Bonn

21. Juni 2006

ASIC=Application Specific Integrated Circuit



We see $((\overline{(a \wedge b))} \wedge b) \vee (\overline{b} \wedge c)$

$$= ((\overline{a} \vee \overline{b}) \wedge b) \vee (\overline{b} \wedge c)$$

$$= ((\overline{a} \wedge b) \vee (\overline{b} \wedge b)) \vee (\overline{b} \wedge c)$$

$$= (\overline{a} \wedge b) \vee (\overline{b} \wedge c)$$

$$= \dots$$

Logic synthesis (in a perfect world)

- Specification of the desired function
 - HDL (= hardware description language)
 - RTL (= register transfer level)
- Technology mapping
 - Assign to each element of the RTL an actual block from the library
 - Compose/decompose gates
 - Optimize the logic (depth, size, wireability)
- Netlist
- Physical design
 - Placement, routing, clocking
 - Timing analysis and optimization \Rightarrow Timing Closure
- Sign-off and production

VHDL to RTL

```
ENTITY Example IS
  PORT (A, B, C, D, E
        : IN BIT;
        CLK
        : IN BIT;
        OUTPUT
        : OUT BIT;
        R
        : OUT BIT);
END Example;
```

```
ARCHITECTURE Behavior Of example IS
BEGIN
```

```
  PROCESS (CLK, A, B, C, D, E)
    VARIABLE S
      : BIT;
```

```
  BEGIN
```

```
    S := A or B;
    OUTPUT <= S or C;
```

```
    IF (not CLK'STABLE) and (CLK = '1') THEN
```

```
      IF (A = '0')
```

```
        THEN R <= S or D;
```

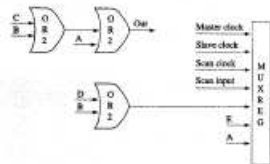
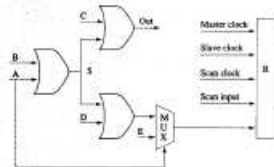
```
        ELSE R <= E;
```

```
      END IF;
```

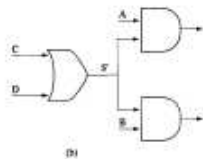
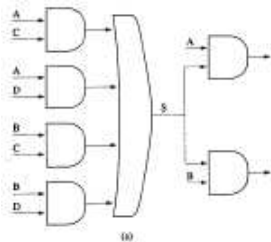
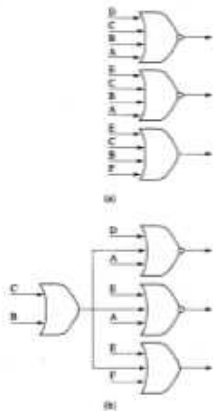
```
    END IF;
```

```
  END PROCESS;
```

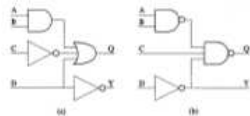
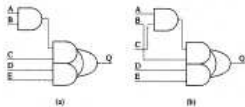
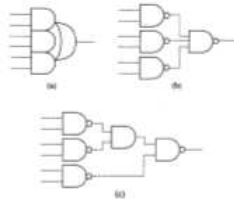
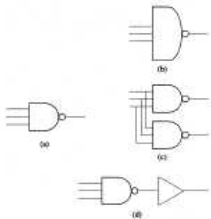
```
END Behavior;
```



Local logic changes



Local logic changes, II



Some observations

- Most of the essential decisions about the logic of the chip are taken long before any physical detail is known
- Changes (and hopefully improvements) are therefore based on crude estimates and of local nature
- No non-local optimization is applied/known
- Once the netlist is passed to physical design, changes in the logic are considered “dangerous” (Boolean equivalence)

Repeater trees distribute one signal via a logically trivial structure to many locations. Conversely, the task of logic design/redesign/optimization is to

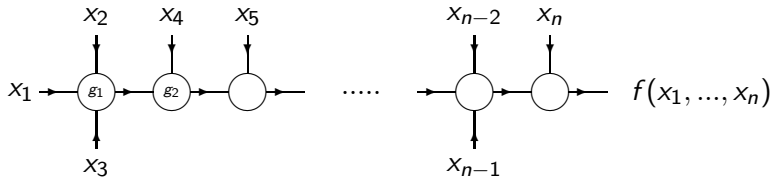
- combine (many) signals
- arising at various locations and points in time
- to (few) signals desired at various locations and points in time
- via a logically non-trivial structure.

We will focus on the “many” to “few” = 1 case, i.e redesign of critical paths.

Logic optimization of critical paths

- If timing problems remain after placement, routing, gate sizing, repeater tree insertion, ..., then these are (often) caused by 'logical depth'
- Static timing analysis calculates slacks for all relevant signals
- The so-called endpoint report indicates a list of critical paths
- Now most physical details can be much better estimated than during the initial logic synthesis
- Specifically, we can take the arrival times into account

A critical path



x_1

$g_1(x_1, x_2)$

$g_2(g_1(x_1, x_2), x_3)$

$\dots g_3(g_2(g_1(x_1, x_2), x_3), x_4) \dots$

$g_{n-1}(g_{n-2}(\dots g_3(g_2(g_1(x_1, x_2), x_3), x_4) \dots, x_{n-1}), x_n)$

For this critical path we have to find a logically equivalent yet better realization. Doing this, we can now take the arrival times of the signals x_i into account.

A Boolean circuit C is an directed acyclic graph whose vertices have been identified with

- inputs,
- outputs or
- evaluations of “simple” functions.

The set of simple functions is called the basis Ω of C .

Classical measures are size and depth of C

$\text{size}(C)$

$\text{depth}(C)$

- 3Sat
- Savage, Wegener, ...
- Circuits for special functions (e.g. binary addition)
- average complexity measures
- heuristics (IEEE, IBM's *Booledozer*, LLO (= local logic optimization,...))

Signal arrival times were not (efficiently) considered.

- Yeh et al., Generalized earliest-first fast addition algorithm, *IEEE Trans. Computers* **52** (2003)
- J. Liu et al., An algorithmic approach for generic parallel adders, in *Proc. ICCAD '03*

The delay of a Boolean circuit

Let C be a Boolean circuit with inputs

$$x_1, x_2, \dots, x_n$$

and let x_i have arrival time

$$t_i \in \mathbb{N}_0.$$

The delay of an input x_i in C is the sum of t_i and the maximal length of a directed path in C starting at x_i . The delay

$$\text{delay}(C)$$

of C is the maximal delay of an input of C .

Small depth \Rightarrow small delay?

$$\text{delay}(C) \geq \max\{\text{depth}(C), \max\{t_1, t_2, \dots, t_n\}\}$$

$$\text{delay}(C) \leq \text{depth}(C) + \max\{t_1, t_2, \dots, t_n\}$$

\Rightarrow

The delay of circuits of minimum depth (for a given function f) have only twice the minimum delay.

$$\text{delay}(f) := \min_C \text{delay}(C)$$

Lemma

If C is a Boolean circuit with “fan-in” at most r for a function depending on n inputs with arrival times

$$t_1, t_2, \dots, t_n \in \mathbb{N}_0,$$

then

$$\text{delay}(C) \geq \left\lceil \log_r \left(\sum_{i=1}^n r^{t_i} \right) \right\rceil.$$

Proof: (Kraft's inequality)

$$\sum_{i=1}^n r^{-(\text{delay}(C)-t_i)} \leq 1 \Rightarrow r^{\text{delay}(C)} \geq \sum_{i=1}^n r^{t_i}.$$

□

Theorem

If the Boolean function f depends on n inputs and for every assignment of arrival times there exists a Boolean circuit for f of fan-in at most 2 whose delay matches the lower bound, then

$$f(x_1, x_2, \dots, x_n) = y_1 \wedge y_2 \wedge \dots \wedge y_n$$

or

$$f(x_1, x_2, \dots, x_n) = y_1 \vee y_2 \vee \dots \vee y_n$$

or

$$f(x_1, x_2, \dots, x_n) = y_1 \oplus y_2 \oplus \dots \oplus y_n,$$

with $y_i \in \{x_i, \neg x_i\}$ for $1 \leq i \leq n$.

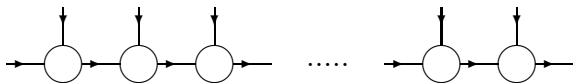
Proof of the theorem

“ \Leftarrow ” trivial

“ \Rightarrow ” Consider $\{t_1, t_2, \dots, t_n\} = \{1, 1, 2, 3, \dots, n-1\}$.

$$\Rightarrow \sum_{i=1}^n 2^{t_i} = 2^1 + 2^1 + 2^2 + \dots + 2^{n-1} = 2^n.$$

$$\Rightarrow \text{delay}(C) \geq n.$$



$$\Rightarrow \forall \pi \in S_n$$

$$f = g_1(x_{\pi(1)}, g_2(x_{\pi(2)}, (\dots, g_{n-1}(x_{\pi(n-1)}, x_{\pi(n)}) \dots)))$$

$$\Rightarrow 10 = 16 - 2 - 4 \text{ possibilities for } g_j.$$

⇒

$$f = y_{\pi(1)}^{\pi} \circ_1^{\pi} (y_{\pi(2)}^{\pi} \circ_2^{\pi} (\dots (y_{\pi(n-1)}^{\pi} \circ_{n-1}^{\pi} y_{\pi(n)}^{\pi}) \dots))$$

with $y_i^{\pi} \in \{x_i, \neg x_i\}$ and $\circ_i^{\pi} \in \{\wedge, \vee, \oplus\} \forall 1 \leq i \leq n$.

If

$$\circ_i^{\text{id}} \neq \circ_{i+1}^{\text{id}},$$

then (discuss)...

□

Resynthesis of a critical path

$$g_{n-1}(g_{n-2}(\dots g_3(g_2(g_1(x_1, x_2), x_3), x_4)\dots, x_{n-1}), x_n)$$

$$g_i \in \Omega = \{\vee, \wedge\} \text{ and fan-in} \leq 2$$

The hardest case is the function

$$f_0(x_1, y_1, x_2, y_2, \dots, x_n, y_n)$$

defined by

$$f_0(\dots) = (((\dots(((x_1 \wedge y_1) \vee x_2) \wedge y_2) \vee \dots) \vee x_n) \wedge y_n.$$

$$f_0 \sim \text{Carry bit of binary addition}$$

Binary addition

Given two n -bit binary numbers

$$u = (u_{n-1}, u_{n-2}, \dots, u_0)$$

and

$$v = (v_{n-1}, v_{n-2}, \dots, v_0),$$

i.e.

$$u = \sum_{i=0}^{n-1} u_i 2^i \text{ and } v = \sum_{i=0}^{n-1} v_i 2^i,$$

we want to calculate the binary representation of

$$u + v = \sum_{i=0}^{n-1} (u_i + v_i) 2^i = \sum_{i=0}^n s_i 2^i.$$

This requires the calculation of the carry bits

$$\begin{aligned}u_0 + v_0 &= c_1 2^1 + s_0 2^0 \\c_1 + u_1 + v_1 &= c_2 2^1 + s_1 2^0 \\c_2 + u_2 + v_2 &= c_3 2^1 + s_2 2^0 \\&\dots\end{aligned}$$

Binary addition, III

If we define

$$p_j = u_j \oplus v_j \text{ and } g_j = u_j \wedge v_j,$$

then

$$c_{j+1} = (p_j \wedge c_j) \vee g_j \text{ and } s_j = p_j \oplus c_j.$$

Note that

$$\begin{aligned}(p_2 \wedge ((p_1 \wedge x) \vee g_1)) \vee g_2 &= (p_2 \wedge (p_1 \wedge x)) \vee (p_2 \wedge g_1) \vee g_2 \\ &= ((p_2 \wedge p_1) \wedge x) \vee ((p_2 \wedge g_1) \vee g_2)\end{aligned}$$

and that

$$(p_2, g_2) \circ (p_1, g_1) := (p_2 \wedge p_1, (p_2 \wedge g_1) \vee g_2)$$

is associative (yet not commutative).

A useful identity

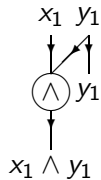
$$\begin{aligned}f_0(x_1, \dots, y_n) &= ((\dots(((x_1 \wedge y_1) \vee x_2) \wedge y_2) \vee \dots) \vee x_n) \wedge y_n \\&= \bigvee_{i=1}^n \left(x_i \wedge \bigwedge_{j=i}^n y_j \right) \\&= \overline{\left(\left(\bigvee_{i=1}^l \left(x_i \wedge \bigwedge_{j=i}^l y_j \right) \right) \wedge \left(\bigwedge_{j=l+1}^n y_j \right) \right) \vee \left(\bigvee_{i=l+1}^n \left(x_i \wedge \bigwedge_{j=i}^n y_j \right) \right)} \\&= \left(f_0(x_1, \dots, y_l) \wedge \left(\bigwedge_{j=l+1}^n y_j \right) \right) \vee f_0(x_{l+1}, \dots, y_n).\end{aligned}$$

Algorithm for f_0

Input: $n \in \mathbb{N}$ and $t_1, s_1, \dots, t_n, s_n \in \mathbb{N}_0$.
 t_i = arrival times of x_i and
 s_i = arrival time of y_i .

Output: $C_0(t_1, s_1, \dots, t_n, s_n)$ für $f_0(x_1, \dots, y_n)$ and $\bigwedge_{j=1}^n y_j$.
 $C_{0, f_0}(t_1, \dots, s_n) = \dots$,
 $C_{0, \wedge}(t_1, \dots, s_n) = \dots$

Algorithm for f_0 , $n = 1$

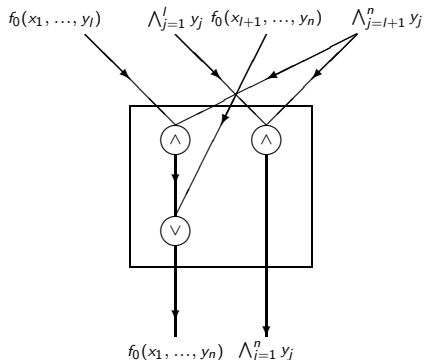


Algorithm for f_0 , $n \geq 2$

Choose $1 \leq l \leq n - 1$ such that

$$\max\{\text{delay}(C_{0,f_0}(t_1, \dots, s_l)) + 2, \text{delay}(C_{0,f_0}(t_{l+1}, \dots, s_n)) + 1\}$$

is minimal.



Lemma

- (i) *The algorithm works correctly.*
- (ii) $\text{size}(C_0(t_1, \dots, s_n)) = 4n - 3$.
- (iii) *In $C_0(t_1, \dots, s_n)$ all inputs have fan-out ≤ 3 and all gates have fan-out ≤ 2 .*
- (iv) $\text{delay}(C_{0,f_0}(t_1, s_1)) = \max\{t_1, s_1\} + 1$.
- (v) $\text{delay}(C_{0,\wedge}(t_1, \dots, s_n)) \leq \text{delay}(C_{0,f_0}(t_1, \dots, s_n)) - 1$.
- (vi) $\text{delay}(C_{0,f_0}(t_1, \dots, s_n))$ equals

$$\min_{1 \leq l \leq n-1} \max \left\{ \begin{array}{l} \text{delay}(C_{0,f_0}(t_1, \dots, s_l)) + 2, \\ \text{delay}(C_{0,f_0}(t_{l+1}, \dots, s_n)) + 1 \end{array} \right\}.$$

An interesting recursion

For $n \geq 2$ and $a, a_1, \dots, a_n \in \mathbb{N}_0$ let $\mathcal{D}(\dots)$ be defined as follows.

$$\mathcal{D}(a) = a \text{ and}$$

$$\mathcal{D}(a_1, \dots, a_n) = \min_{1 \leq l \leq n-1} \max\{\mathcal{D}(a_1, \dots, a_l) + 2, \mathcal{D}(a_{l+1}, \dots, a_n) + 1\}.$$

\Rightarrow

$$\text{delay}(C_{0,f_0}(t_1, \dots, s_n)) = \mathcal{D}(\max\{t_1, s_1\} + 1, \dots, \max\{t_n, s_n\} + 1)$$

An interesting recursion, II

$F_0 = 0$, $F_1 = 1$ and $F_k = F_{k-1} + F_{k-2}$ for $k \geq 2$.

$$Z(k) = \underbrace{(0, 0, \dots, 0)}_k$$

Lemma

Let $k \in \mathbb{N}_0$ and $l, n, m \in \mathbb{N}$.

Let $A \in \mathbb{N}_0^n$ and $B \in \mathbb{N}_0^m$.

- (i) $\max\{i \in \mathbb{N} \mid \mathcal{D}(Z(i)) \leq k\} = F_{k+1}$.
- (ii) $\mathcal{D}(A, l) \leq \mathcal{D}(A, Z(F_{l+1}))$.
- (iii) $\mathcal{D}(A, l, B) \leq \mathcal{D}(A, Z(F_{l+3} - 1), B)$.

Proof of the lemma:

$$(i) m_k := \max\{i \in \mathbb{N} \mid \mathcal{D}(Z(i)) \leq k\}.$$

$$m(0) = 1$$

$$m(1) = 1$$

$$m(2) = 2$$

$$m(k) = m(k-1) + m(k-2)$$

for $k \geq 2$.

(ii) Let (A, I) be a counterexample of minimal length.

Case 1.

$$\mathcal{D}(A, Z(F_{I+1})) = \max\{\mathcal{D}(A_1) + 2, \mathcal{D}(A_2, Z(F_{I+1})) + 1\}$$

with $(A_1, A_2) = A$.

$$\begin{aligned} \Rightarrow \mathcal{D}(A, I) &\leq \max\{\mathcal{D}(A_1) + 2, \mathcal{D}(A_2, I) + 1\} \\ &\leq \max\{\mathcal{D}(A_1) + 2, \mathcal{D}(A_2, Z(F_{I+1})) + 1\} \\ &= \mathcal{D}(A, Z(F_{I+1})). \end{aligned}$$

Case 2.

$$\mathcal{D}(\dots) = \max\{\mathcal{D}(A, Z(F_{l+1} - r)) + 2, \mathcal{D}(Z(r)) + 1\}$$

for some $1 \leq r \leq F_{l+1} - 1$.

$$\Rightarrow \mathcal{D}(A, l) \leq \max\{\mathcal{D}(A) + 2, l + 1\}$$

$$\Rightarrow \text{If } \mathcal{D}(A) + 2 \geq l + 1 \Rightarrow$$

$$\mathcal{D}(A, l) \leq \mathcal{D}(A) + 2 \leq \mathcal{D}(A, Z(F_{l+1} - r)) + 2 \leq \mathcal{D}(A, Z(F_{l+1})).$$

$$\Rightarrow \text{If } \mathcal{D}(A) + 2 < l + 1 \Rightarrow$$

$$\mathcal{D}(A, l) \leq l + 1 \leq \mathcal{D}(Z(F_{l+1} + 1)) \leq \mathcal{D}(A, Z(F_{l+1})).$$

□

An interesting recursion, III

Theorem

$$\mathcal{D}(a_1, \dots, a_n) < \log_{\frac{\sqrt{5}+1}{2}} \left(\sum_{i=1}^n 2^{a_i} \right) + 2 \approx 1.44 \log_2 \left(\sum_{i=1}^n 2^{a_i} \right) + 2$$

for $a_1, a_2, \dots, a_n \in \mathbb{N}_0$.

Corollary

$$\text{delay}(C_0, f_0) \leq 1.44 \text{delay}(f_0) + 3.$$

$$\text{delay}(C(\dots)) \leq 1.44 \text{delay}(\dots) + 4.44 \text{ for } g_i \in \Omega = \{\vee, \wedge\}.$$

- $\mathcal{D}(a_1, \dots, a_n)$ describes the minimal “depth” of alphabetic code trees.

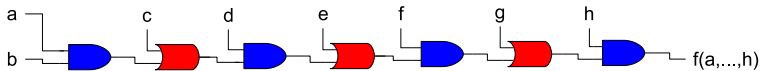
$$\mathcal{D}_d(\mathbf{a}) = \min\{\max\{\mathcal{D}_d(\mathbf{a}_i) + d_i \mid 1 \leq i \leq k\} \mid \mathbf{a} = \mathbf{a}_1\mathbf{a}_2\dots\mathbf{a}_k\}$$

$$(1.44 \rightarrow 1 + \epsilon)$$

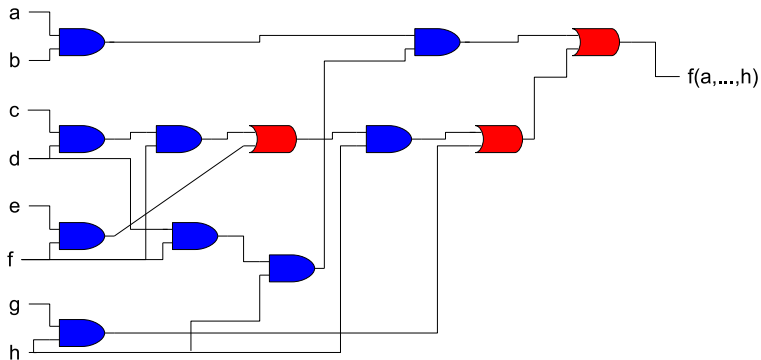
- Analysis of algorithms, random trees, Hu-Tucker coding, group testing, dichotomous search,...
- Kapoor and Reingold, Hinderer, Hwang,...

$$M(n) = \min \max\{\alpha + M(k), \beta + M(n - k)\}.$$

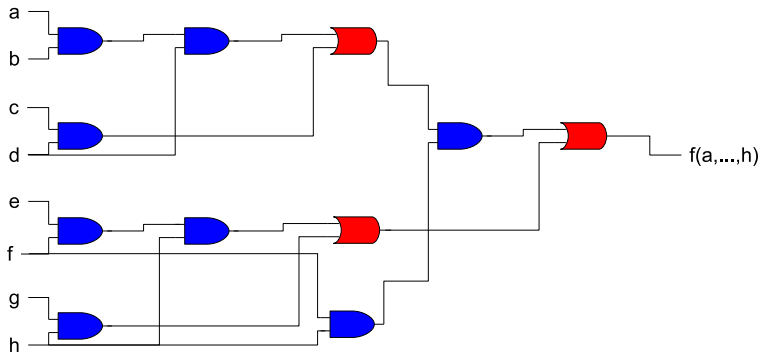
Examples



Examples



Examples



- Improve the algorithm with respect to
 - signal locations
 - using complex gates
 - bridge distance with inverter trees
 - avoidance of cloning
 - gate sizing
 - blockages
 - better delay modell
 - ...
- Resynthesis of larger pieces of the logic

The Prefix Problem

Given: An associative operation $\circ : D^2 \rightarrow D$
and inputs $x_1, x_2, \dots, x_n \in D$.

Result: A circuit with outputs
 $x_1 \circ x_2 \circ \dots \circ x_i \quad \forall 1 \leq i \leq n$.

The Prefix Problem, II

- Binary addition
- The above yields $\text{size} = O(n^2)$
- $\text{depth} = \log_2(n) + o(\log(n))$, $\text{size} = O(n \log(n))$
or
 $\text{depth} = 2 \log_2(n) + o(\log(n))$, $\text{size} = O(n)$.
- $\text{depth} = \lceil \log_2(n) \rceil + k$, $\text{size} = 2n \left(1 + \frac{1}{2^k}\right)$
 $\forall 0 \leq k \leq \lceil \log_2(n) \rceil$ (Ladner and Fischer)
- No construction considers arrival times

Construction for the Prefix Problem

Depending on the arrival times

$$t_1, t_2, \dots, t_n$$

of the inputs we will construct a circuit

$$P(t_1, t_2, \dots, t_n)$$

solving the prefix problem over the basis $\{\circ\}$.

For $n = 1$ the circuit $P(t_1)$ consists just of the input vertex x_1 having fan-out 1.

For $n \geq 2$ we apply the following steps.

Construction for the Prefix Problem, Step 1

Partition the set $\{1, 2, \dots, n\}$ into $l := \lceil \sqrt{n} \rceil$ sets

$$V_1 = \{1, 2, \dots, n_1\},$$

$$V_2 = \{(n_1 + 1), (n_1 + 2), \dots, (n_1 + n_2)\}, \dots,$$

$$V_l = \{(n_1 + n_2 + \dots + n_{l-1} + 1), \dots, (n_1 + n_2 + \dots + n_l)\}$$

such that $n_1 \geq n_2 \geq \dots \geq n_l$ and $n_1 - n_l \leq 1$.

Construction for the Prefix Problem, Step 2

For $1 \leq i \leq l$ we use the following dynamic programming approach to construct a circuit C_i over the basis $\{\circ\}$ calculating

$$y_i := \circ_{j \in V_i} x_j$$

In what follows C_{j_1, j_2} will denote a circuit calculating

$$\circ_{j=j_1}^{j_2} x_{(n_1+n_2+\dots+n_{i-1}+j)}$$

for $1 \leq j_1 \leq j_2 \leq n_i$.

If $j_1 = j_2$, then C_{j_1, j_2} consists just of the corresponding input vertex.

If $j_1 < j_2$, we recursively construct $C(j_1, j_2)$ using one \circ -gate joining the outputs of two circuits $C(j_1, l)$ and $C(l, j_2)$ such that

$$\max\{\text{delay}(C(j_1, l)), \text{delay}(C(l, j_2))\}$$

is minimized.

Let $C_i = C(1, n_i)$.

Clearly, C_i use $(n_i - 1)$ o-gates.

It follows as above that the computation of y_i by C_i terminates at time $t(y_i)$ with

$$t(y_i) \leq \log_2 \left(\sum_{j \in V_i} 2^{t_j} \right) + 2.$$

Construction for the Prefix Problem, Step 3

For $1 \leq i \leq l$ we recursively construct

$$P(t_{(n_1+n_2+\dots+n_{i-1}+1)}, \dots, t_{(n_1+n_2+\dots+n_i-1)})$$

and use these circuits to calculate all $(n_i - 1)$ prefixes on the inputs x_j for $j \in V_i \setminus \{n_1 + n_2 + \dots + n_i\}$.

Construction for the Prefix Problem, Step 4

We construct $P(t(y_1), t(y_2), \dots, t(y_{l-1}))$ to calculate all $(l - 1)$ prefixes on the inputs y_j for $1 \leq j \leq l - 1$ calculated by the circuits constructed in Step 2.

Construction for the Prefix Problem, Step 5

For $2 \leq i \leq l$ and $j \in V_i \setminus \{n_1 + n_2 + \dots + n_i\}$ we join the output $(y_1 \circ y_2 \circ \dots \circ y_{i-1})$ of the circuit constructed in Step 4 with the output

$$\bigcirc_{\substack{k \in V_i \\ k \leq j}} x_k$$

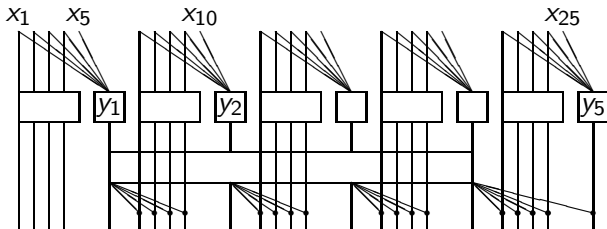
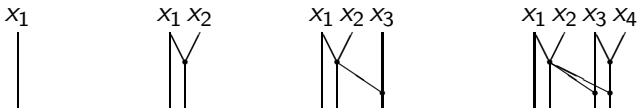
of the circuit constructed in Step 3 using one \circ -gate which calculates

$$\bigcirc_{k=1}^j x_k = (y_1 \circ y_2 \circ \dots \circ y_{i-1}) \circ \left(\bigcirc_{\substack{k \in V_i \\ k \leq j}} x_k \right).$$

Finally, we join the output $(y_1 \circ y_2 \circ \dots \circ y_{l-1})$ of the circuit constructed in Step 4 with the output y_l of the circuit constructed in Step 2 using one \circ -gate which calculates

$$\circ_{i=1}^n x_i.$$

Construction for the Prefix Problem



For $w \geq n \geq 1$ let $\text{size}(n)$ and $\text{delay}(w, n)$ denote the maximum size and the maximum delay of one of the constructed circuits with $t_i \in \mathbb{N}_0$ for $1 \leq i \leq n$ and $w = \sum_{i=1}^n 2^{t_i}$.

Lemma

For $w \geq n \geq 3$

$$\begin{aligned}\text{size}(n) &\leq (\lceil \sqrt{n} \rceil + 1) \text{size}(\lceil \sqrt{n} \rceil - 1) + 2(n - \lceil \sqrt{n} \rceil) \\ \text{delay}(w, n) &\leq \text{delay}(4w, \lceil \sqrt{n} \rceil - 1) + 1.\end{aligned}$$

Theorem

The prefix problem on inputs x_1, x_2, \dots, x_n with arrival times $t_1, t_2, \dots, t_n \in \mathbb{N}_0$ can be solved by

(i) a circuit over the basis $\{\circ\}$ with size $O(n \log(\log(n)))$ and delay

$$\log_2 \left(\sum_{i=1}^n 2^{t_i} \right) + 3 \log_2(\log_2(n)) + O(1);$$

(ii) a circuit over the basis $\{\circ, \text{id}\}$ satisfying the fan-out conditions ("1 for \circ and 2 for id ") with size $O(n \log(\log(n)))$ and delay

$$\log_2 \left(\sum_{i=1}^n 2^{t_i} \right) + \log_2(n) + 7 \log_2(\log_2(n)) + O(1).$$

$$\text{sel}(x, y, z) = (x \wedge y) \vee ((\neg x) \wedge z)$$

Theorem (Spira '71, Krapchenko '81,...)

For a Boolean function f (monotone Boolean function f) and a complete basis (the monotone basis)

$$\log_2(\text{formulasize}_\Omega(f) + 1) \leq \text{depth}(f)$$

and

$$\text{depth}(f) \leq \frac{\text{depth}_\Omega(\text{sel}) + 1}{\log_2 3 - 1} \log_2(\text{formulasize}_\Omega(f) + 1).$$

Theorem

Let $r \geq 2$ and let Ω be a set of boolean functions on at most r inputs. Let $\alpha \in \mathbb{N}$ be the minimum depth of a circuit over Ω for the function $\text{sel}(x, y, z) = (x \wedge y) \vee ((\neg x) \wedge z)$. For every function in Ω on $l \geq 2$ inputs, let Ω contain all functions on $(l - 1)$ inputs that arise by setting one of the inputs to 0 or 1.

Then there is some constant $\beta = \beta(\Omega)$ with the following property: Let C be a read-once formula, i.e. a circuit for a boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ over Ω such that the fan-out of input vertices and gates is at most 1. For $1 \leq i \leq n$ let $t_i \in \mathbb{N}_0$ be the arrival time of the i -th input of f .

Then there is a circuit \tilde{C} for f over Ω such that

$$\text{delay}(\tilde{C}) \leq \frac{\alpha}{\log_r(r+1) - 1} \log_r \left(\sum_{i=1}^n r^{t_i} \right) + \beta. \quad (1)$$

We prove the result by induction over n .

Let $w = \sum_{i=1}^n r^{t_i}$ and note that $\frac{\alpha}{\log_r(r+1)-1} = \frac{\alpha}{\log_r\left(\frac{r+1}{r}\right)} > \alpha \geq 1$.

If $n = 1$, then

$$\frac{\alpha}{\log_r(r+1)-1} \log_r \left(\sum_{i=1}^n r^{t_i} \right) + \beta = \frac{\alpha}{\log_r(r+1)-1} t_1 + \beta \geq t_1 + \beta.$$

Therefore, there is some $\beta \in \mathbb{N}$ independent of C and f such that (1) holds for $n = 1$.

We may assume that $\beta \geq \alpha$.

Now let $n \geq 2$. The directed graph underlying C is a rooted tree T whose leaves are the inputs x_1, x_2, \dots, x_n . For every vertex u of T let $w(u)$ denote the sum of r^{t_i} where the sum extends over all i such that x_i lies in the subtree of T rooted at u .

Let the vertex u be chosen such that

(i) $w(u) > \frac{w}{r+1}$,

(ii) $w(u)$ is minimum subject to (i)

and u has maximum distance from the root subject to (i) and (ii).

It is easy to see that $w(u) < w$.

Let C_u denote the subcircuit of C corresponding to the subtree of T rooted at u . For $i \in \{0, 1\}$ let C_i denote the circuit that arises from C by replacing the output of u by the constant i .

Since C_u , C_1 and C_0 are circuits for functions defined on at most $(n - 1)$ inputs, we can apply the induction hypothesis to them.

This implies the existence of circuits \tilde{C}_u , \tilde{C}_1 and \tilde{C}_0 over Ω for the same functions whose delay is bounded as in (1).

Note that if g denotes the function computed at the vertex u , then clearly $f = \text{sel}(g, f|_{g=1}, f|_{g=0})$. Therefore, using \tilde{C}_u , \tilde{C}_1 , \tilde{C}_0 and the circuit for sel over Ω , we can construct a circuit \tilde{C} for f over Ω .

For this circuit \tilde{C} we have
 $\text{delay}(\tilde{C})$

$$\leq \alpha + \frac{\alpha}{\log_r(r+1) - 1} \max \{ \log_r(w(u)), \log_r(w - w(u)) \} + \beta. \quad (2)$$

By the choice of u , we have $w - w(u) \leq \frac{wr}{r+1}$.

If $w(u) \leq \frac{wr}{r+1}$, then (2) implies

$$\begin{aligned} \text{delay}(\tilde{C}) &\leq \alpha + \frac{\alpha}{\log_r(r+1) - 1} \log_r \left(\frac{wr}{r+1} \right) + \beta \\ &= \frac{\alpha}{\log_r(r+1) - 1} \log_r(w) + \beta. \end{aligned}$$

Hence, we may assume that $w(u) > \frac{wr}{r+1}$.

In this case u must be a leaf of T and C_u has delay $\log_r(w(u))$.
Therefore, we can strengthen (2) as follows.

$$\begin{aligned} & \text{delay}(\tilde{C}) \\ & \leq \alpha + \max \left\{ \log_r(w(u)), \frac{\alpha}{\log_r(r+1) - 1} \log_r(w - w(u)) + \beta \right\}. \quad (3) \end{aligned}$$

If the right-hand term yields the maximum in (3), then we can proceed as before. Hence, we may assume that the left-hand term yields the maximum in (3) and trivially we obtain

$$\text{delay}(\tilde{C}) \leq \alpha + \log_r(w(u)) \leq \frac{\alpha}{\log_r(r+1) - 1} \log_r(w) + \beta,$$

which completes the proof. \square

Thank you for your attention!!