Chapter 6: Priority Encoders

Computer Structure - Spring 2004

Dr. Guy Even

Tel-Aviv Univ.

Leading One

Consider a binary string $x[0:n-1]$ (ascending indexes!).
DEF: The leading one of a binary string $x[0:n-1]$ is defined by

$$\text{LEADING-ONE}(x[0:n-1]) \triangleq \begin{cases} \min\{i \mid x[i]=1\} & \text{if } x[0:n-1] \neq 0^n \\ n & \text{otherwise.} \end{cases}$$

Example: Consider the string $x[0:6] = 0110100$. The leading one is the index $[1]$. Note that indexes are in ascending order and that $x[0]$ is the leftmost bit.
Claim: For every binary string $x[n-1:0]$

$$\text{LEADING-ONE}(\vec{a}) = \text{LEADING-ONE}(\vec{a} \cdot 1).$$

Unary Representation

DEF: A binary string $x[0:n-1]$ represents a number in unary representation if $x[0:n-1] \in 1^* \cdot 0^*$. The *value represented in unary representation* by the binary string $1^i \cdot 0^j$ is i.

Example: The binary string 01001011 does not represent a number in unary representation. Only a string that is obtained by concatenating an all-ones string with an all-zeros string represents a number in unary representation.

Parallel Prefix Computation

DEF: A parallel prefix computation OR circuit of length n is a combinational circuit specified as follows.

Input: $x[0:n-1]$.

Output: $y[0:n-1]$.

Functionality:

$$y[i] = \text{OR}(x[0:i]).$$

We denote parallel prefix computation OR circuit of length n by PPC–OR(n).

Priority Encoders

- A priority encoder is a combinational circuit that computes the leading one.
- We consider two types of priority encoders:
  - A unary priority encoder - outputs the leading one in unary representation.
  - A binary priority encoder -outputs the leading one in binary representation.

Unary priority encoder

DEF: A unary priority encoder U-PENC(n) is a combinational circuit specified as follows.

Input: $x[0:n-1]$.

Output: $y[0:n-1]$.

Functionality:

$$y[i] = \text{INV}(\text{OR}(x[0:i])).$$

Example:

- If $x[0:6] = 0110100$, then PPC–OR(7) outputs 0111111. U-PENC(7) outputs 1000000.
- If $\vec{x} \neq 0^n$, then U-PENC(n) outputs $y[0:n-1] = 1^j \cdot 0^{n-j}$, where $j = \min\{i \mid x[i]=1\}$.
- If $\vec{x} = 0^n$, then $\vec{y} = 1^n$ and $\vec{y}$ is a unary representation of n.

Binary Priority Encoder

DEF: A binary priority encoder B-PENC(n) is a combinational circuit specified as follows.

Input: $x[0:n-1]$.

Output: $y[k:0]$, where $k = \lfloor \log_2 n \rfloor$. (Note that if $n = 2^\ell$, then $k = \ell$.)

Functionality:

$$\langle \vec{y} \rangle = \text{LEADING-ONE}(\vec{x})$$

$n = 2^k \implies$ length of the output of a B-PENC(n) is $k+1$ bits; otherwise, the number n could not be represented by the output.

Example: Given input $x[0:5] = 000101$, a U-PENC(6) outputs $y[0:5] = 000111$, and B-PENC(6) outputs $y[2:0] = 011$.

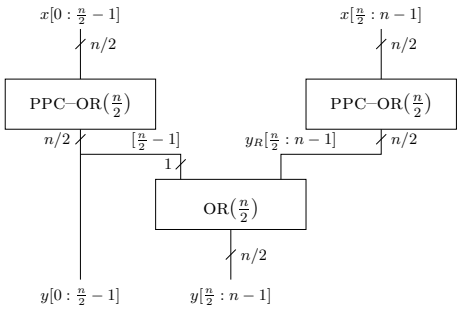
U-PENC(n) - Implementation

- Invert the outputs of a PPC–OR(n).
- Brute force design of PPC–OR(n):
  - separate OR-tree for each output bit.
  - delay = $O(\log n)$
  - cost = $O(n^2)$.
- How can we efficiently combine these trees? To be discussed in detail when we discuss fast addition.
- We now present a (non-optimal) design based on divide-and-conquer.

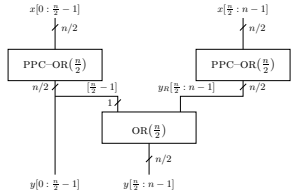
divide & conquer PPC–OR(n)

If $n = 1$, then $y[0] \leftarrow x[0]$.

If $n > 1$:



divide & conquer PPC–OR(n) - cont.



Question:

1. Prove the correctness of the design.
2. Extend design for values of n that are not powers of 2.
3. Analyze the delay and cost of the design.
4. Prove the asymptotic optimality of the delay of the design.

divide & conquer PPC–OR(n) - cost analysis

$$c(n) = \begin{cases} 0 & \text{if n=1} \\ 2 \cdot c(\frac{n}{2}) + (n/2) \cdot c(\text{OR}) & \text{otherwise.} \end{cases}$$

It follows that

$$\begin{aligned} c(n) &= 2 \cdot c(\frac{n}{2}) + \Theta(n) \\ &= \Theta(n \cdot \log n). \end{aligned}$$

Question: Prove a lower bound on $c(\text{PPC–OR}(n))$.

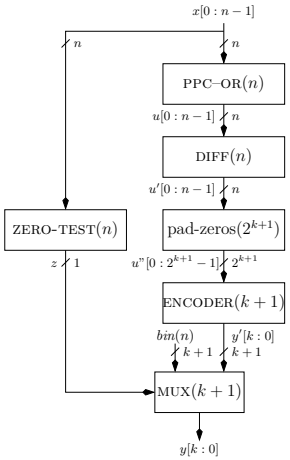
Promise: In the chapter on fast addition we will present a cheaper implementation of PPC–OR(n) (with logarithmic delay).

Implementation of a binary priority encoder

We present two designs for a binary priority encoder.

- design based on a reduction to PPC–OR(n).
- design based on divide-and-conquer

reduction of B-PENC(n) to PPC–OR(n)



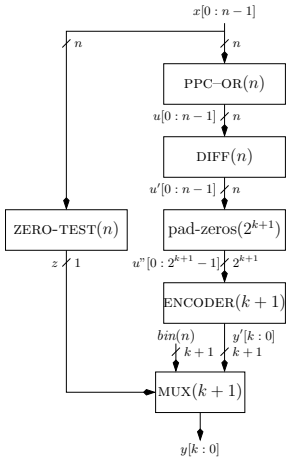
- apply PPC–OR(n)
- difference:

$$u'[i] = \begin{cases} u[0] & \text{if } i = 0 \\ u[i] - u[i-1] & \text{otherwise.} \end{cases}$$

if $\vec{x} \neq 0^n$, then $\vec{u'}$ is a 1-out-of-n representation of leading one's position.

- Pad & Encode.
- Select *bin*(n) if $\vec{x} = 0^n$.

cost analysis

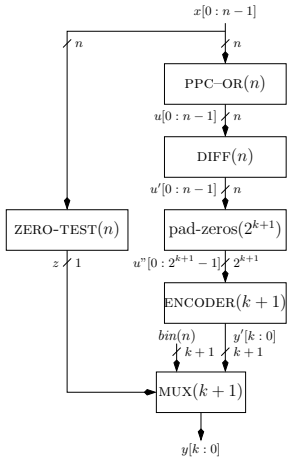


- zero cost: padding with zeros.
- logarithmic cost: multiplexer
- linear cost:
  - difference
  - encoder
  - zero tester
- $\Longrightarrow$

$$c(\text{B-PENC}(n)) = c(\text{PPC–OR}(n)) + O(n).$$

- If $c(\text{PPC–OR}(n)) = O(n)$, then also $c(\text{B-PENC}(n)) = O(n)$.

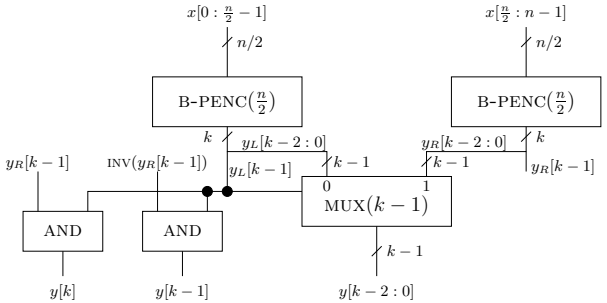
delay analysis



- zero delay: padding with zeros.
- constant delay:
  - difference
  - multiplexer
- logarithmic delay:
  - PPC–OR(n)
  - encoder
  - zero tester
- $\Longrightarrow$

$$d(\text{B-PENC}(n)) = O(\log n).$$

B-PENC(n): a divide-and-conquer design for $n = 2^k$

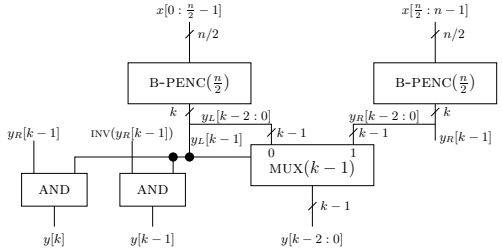


correctness

We prove correctness by induction.

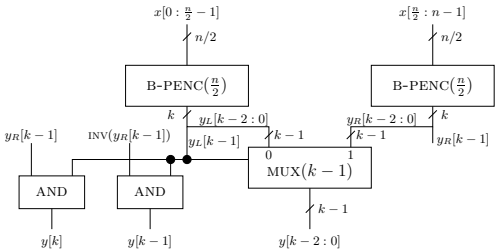
- Induction basis: $n = 1$ is trivial.
- Induction step deals with 3 cases:
  - leading one is in left half $x[0:\frac{n}{2}-1]$.
  - leading one is in right half $x[\frac{n}{2}:n-1]$.
  - $\vec{x} = 0^n$.

correctness: case $x[0:\frac{n}{2}-1] \neq 0^{n/2}$



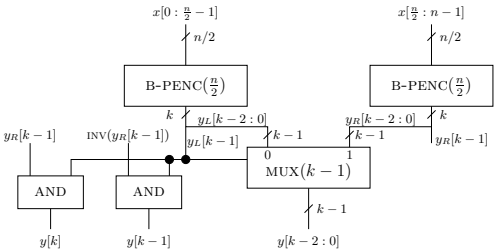
- Ind. Hyp. $\Rightarrow$ required output is $0 \cdot y_L[k-1:0]$.
- index of the leading one $< n/2 \Rightarrow \quad y_L[k-1] = 0$.
- $\Rightarrow \quad y[k] = y[k-1] = 0$ and $y[k-2:0] = y_L[k-2:0]$.
- $\Rightarrow$ output $\vec{y} = 0 \cdot y_L[k-1:0]$.

correctness: case $x[0:\frac{n}{2}-1]=0^{n/2}$ & $x[\frac{n}{2}:n-1]\neq 0^{n/2}$



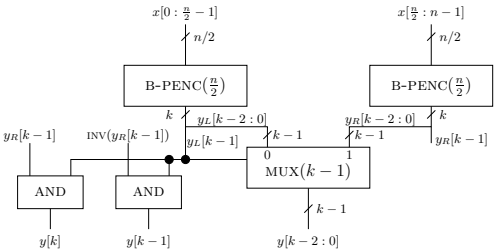
- Ind. Hyp. $\Rightarrow$ index of the leading one is $n/2+\langle\vec{y_R}\rangle$.
- $\Rightarrow$ required output is $0\cdot 1\cdot y_R[k-2:0]$.
- Ind. Hyp. $\Rightarrow$ $y_L[k-1]=1$ and $y_R[k-1]=0$.
- $\Rightarrow$ $y[k]=0$ & $y[k-1]=1$.
- $y_L[k-1]=1 \Rightarrow y[k-2:0]=y_R[k-2:0]$.

correctness: case $x[0:\frac{n}{2}-1]=0^{n/2}$ & $x[\frac{n}{2}:n-1]=0^{n/2}$



- required output is $1\cdot 0^k$.
- Ind. Hyp. $\Rightarrow$ $y_L[k-1:0]=y_R[k-1:0]=1\cdot 0^{k-1}$.
- $y_L[k-1]=y_R[k-1]=1 \Rightarrow y[k]=1$.
- $y_L[k-1]=y_R[k-1]=1 \Rightarrow y[k-1]=0$.
- $y_L[k-1]=1 \Rightarrow y[k-2:0]=y_R[k-2:0]=0^{k-1}$.

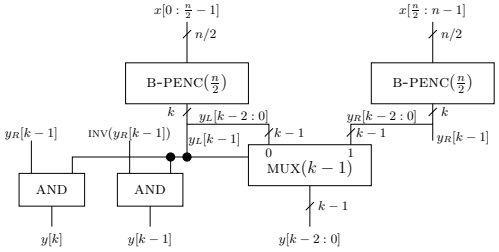
cost analysis



$$c(\text{B-PENC}(n)) = \begin{cases} c(\text{NOR}) & \text{if n=2} \\ 2\cdot c(\text{B-PENC}(n/2)) + 2\cdot c(\text{AND}) & \\ \quad +(k-1)\cdot c(\text{MUX}) & \text{otherwise.} \end{cases}$$

Solution is $c(n)=O(n)$, same recurrence as for ENCODER$''(k)$ with substitution $k=\log n$.

delay analysis



$$d(\text{B-PENC}(n)) = \begin{cases} t_{pd}(\text{NOR}) & \text{if n=2} \\ d(\text{B-PENC}(n/2)) + \max\{d(\text{MUX}), d(\text{AND})\} & \text{otherwise.} \end{cases}$$

$\Longrightarrow$ $d(\text{B-PENC}(n)) = O(\log n)$

Summary - priority encoders

- Two types of priority encoders: U-PENC(n) & B-PENC(n).
- Implementation of U-PENC(n):
  - brute force (separate OR-trees): cost $O(n^2)$, delay $O(\log n)$.
  - divide-&-conquer: cost $O(n\log n)$, delay $O(\log n)$.
  - to be shown: cost $O(n)$ and delay $O(\log n)$.
- Implementation of B-PENC(n):
  - reduction to U-PENC: overhead in cost - $O(n)$ & delay - $O(\log n)$.
  - divide-&-conquer: cost $O(n)$, delay $O(\log n)$.