Vérification automatique de multiplicateurs avec les *BMDs

Pierre Senellart

10 décembre 2002

Introduction

- BDDs [1] very powerful tools for veifying arithmetic circuits.
- But exponential on multipliers.
- *BMDs [2] give a polynomial algorithm but need high-level information.
- Backward construction algorithm [3]

Shannon expansion of a function

$$f : \{0,1\}^n \longrightarrow \{0,1\}$$

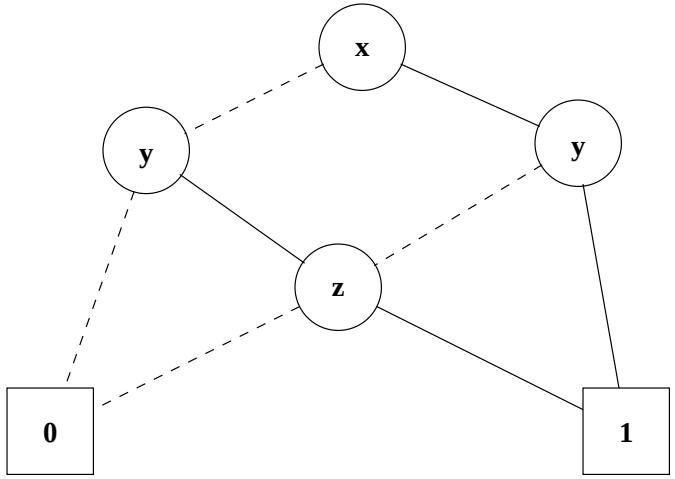
$$\begin{array}{lll} f_{\bar{x}_i} : & \{0,1\}^{n-1} & \rightarrow \quad \{0,1\} \\ & (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) & \mapsto \quad f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n) \end{array}$$

$$\begin{array}{lll} f_{x_i} : & \{0,1\}^{n-1} & \rightarrow \quad \{0,1\} \\ & (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) & \mapsto \quad f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n) \end{array}$$

$$f(x_1, \ldots, x_n) = x_i f_{x_i}(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) + \bar{x}_i f_{\bar{x}_i}(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$$

Ordered BDDs

- Unique representation of a boolean function ($\{0,1\}^n \rightarrow \{0,1\}$)
- Some initial total ordering of the variables
- Example: carry of a full adder



Use of BDDs

- BDDs very efficient for a large class of circuits
- But BDDs are inefficent on multipliers.

  Theorem. [Bryant,1986] *For any ordering of the variables, the BDD representation of some multiplier output will have a graph of exponential size.*

  Need for something else...

Moment decomposition of a function

$$f : \{0,1\}^n \longrightarrow \mathbb{N}$$

$$\begin{array}{rccl} f_{\bar{x}_i} : & \{0,1\}^{n-1} & \rightarrow & \mathbb{N} \\ & (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) & \mapsto & f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n) \end{array}$$

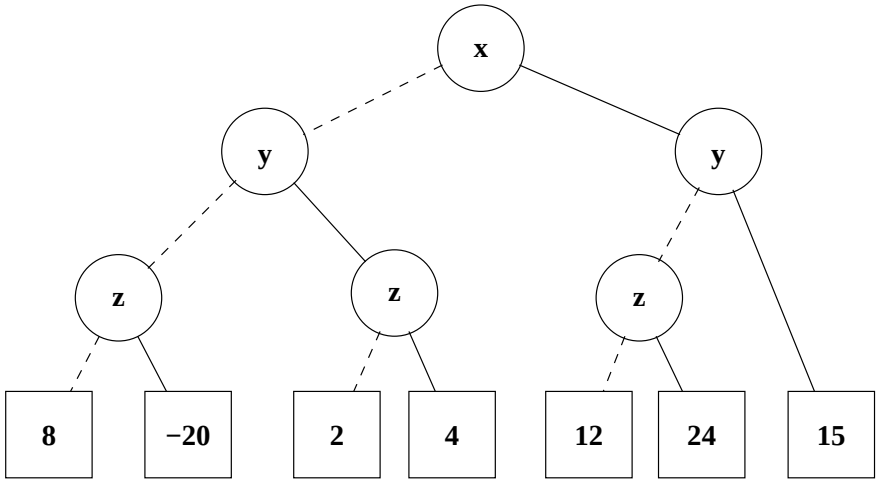
$$\begin{array}{rccl} f_{x_i} : & \{0,1\}^{n-1} & \rightarrow & \mathbb{N} \\ & (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) & \mapsto & f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n) \end{array}$$

Moment decomposition of a function (continuing...)

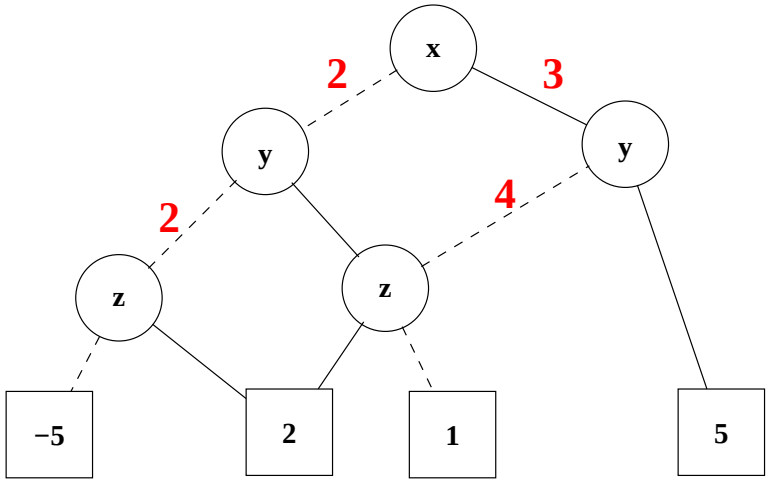
$$f_{\dot{x}_i} = f_{x_i} - f_{\bar{x}_i}$$

$$f(x_1, \ldots, x_n) = \underbrace{f_{\bar{x}_i}(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)}_{\text{constant moment}} + x_i \underbrace{f_{\dot{x}_i}(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)}_{\text{linear moment}}$$

BMDs and *BMDs



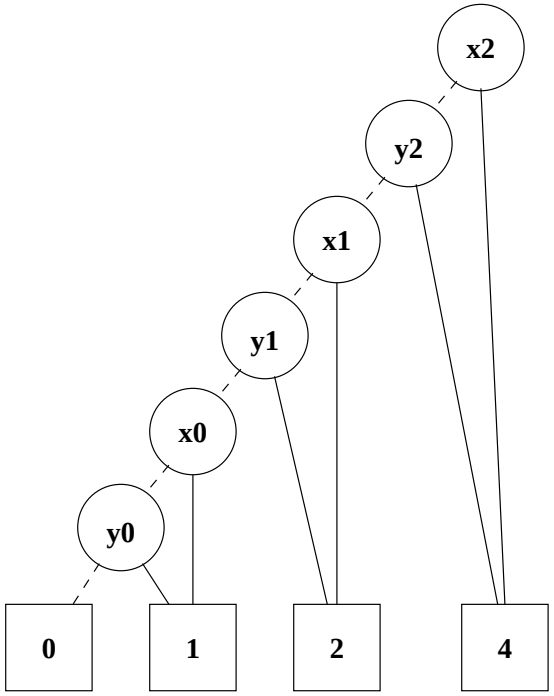
BMD



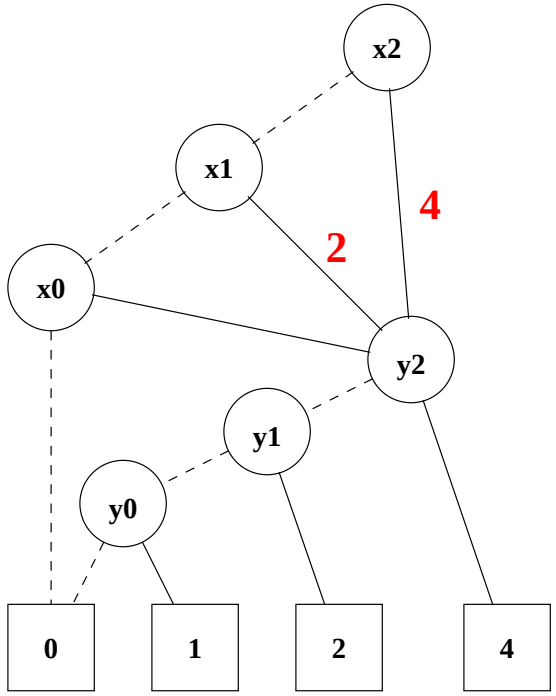
*BMD

$$f(x, y, z) = 8 - 20z + 2y + 4yz + 12x + 24xz + 15xy$$

Arithmetic operations



Addition



Multiplication

*BMDs of classical arithmetic operations are of linear size.

*BMD construction - Bryant's algorithm

Circuit: interconnection of components, described at both bit and word levels.

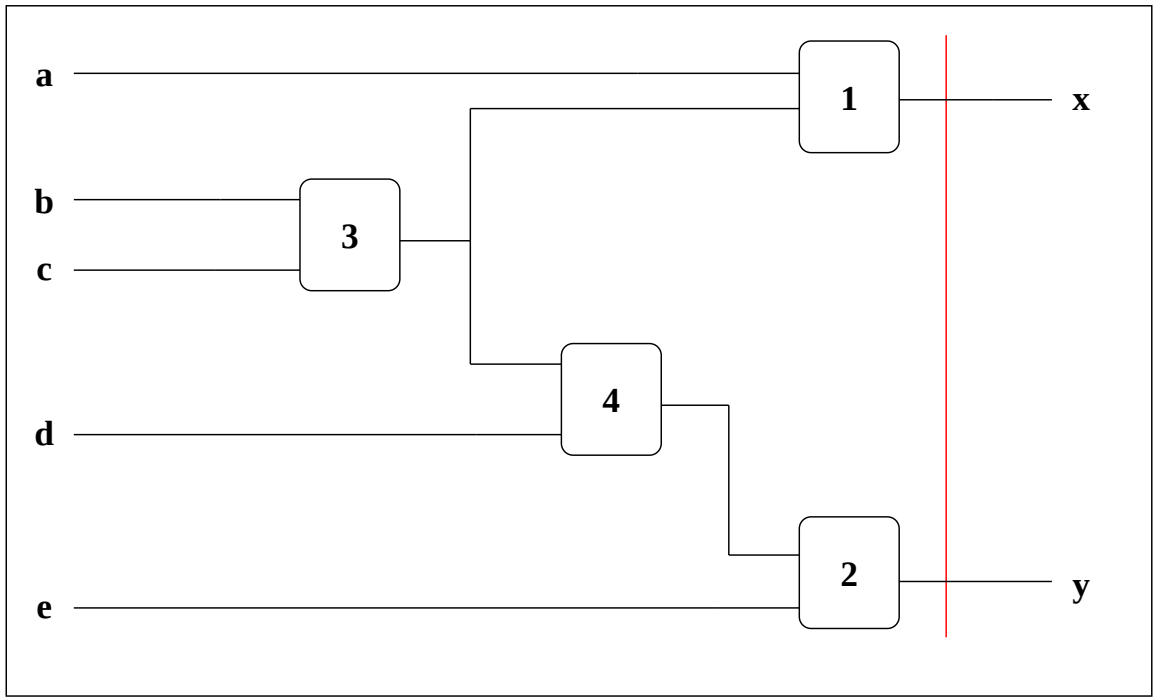
1. Check that the bit-level interpretation of a component matches its word-level interpretation by direct construction of a *BMD.

2. Check that the composition of word-level interpretations of components matches the specification of the circuit by direct construction of a *BMD.

Need of high-level knowledge about the circuit.

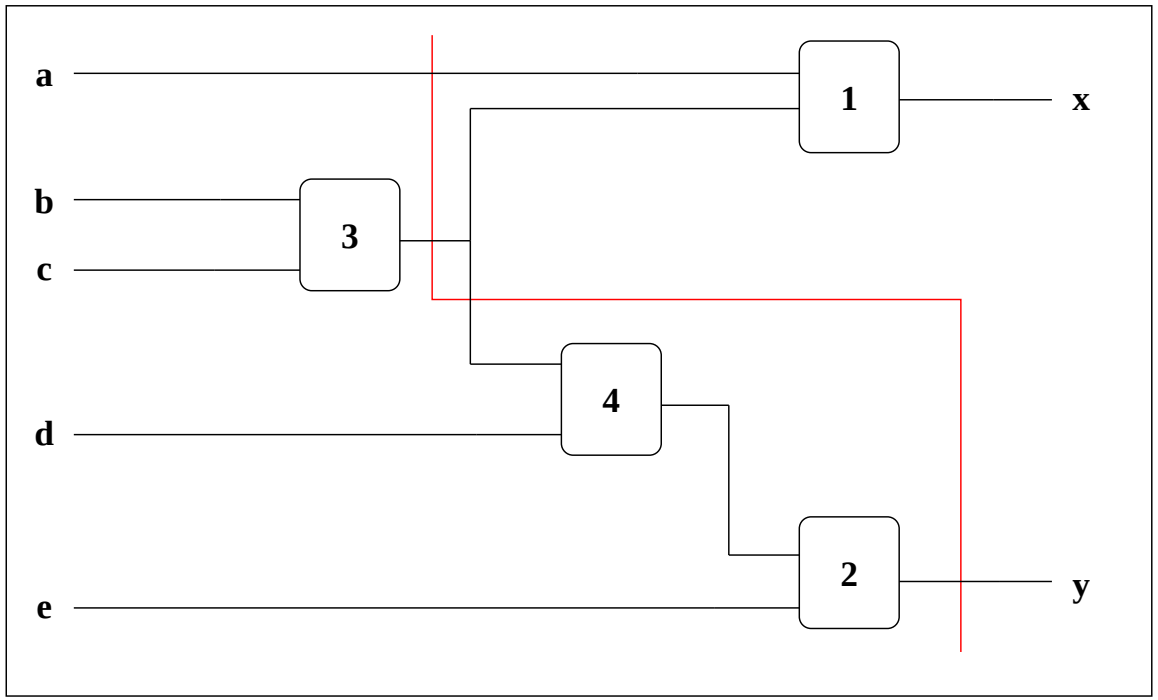
Direct construction of a *BMD of the entire circuit impossible due to the size of intermediary result.

Backward construction algorithm - step 1



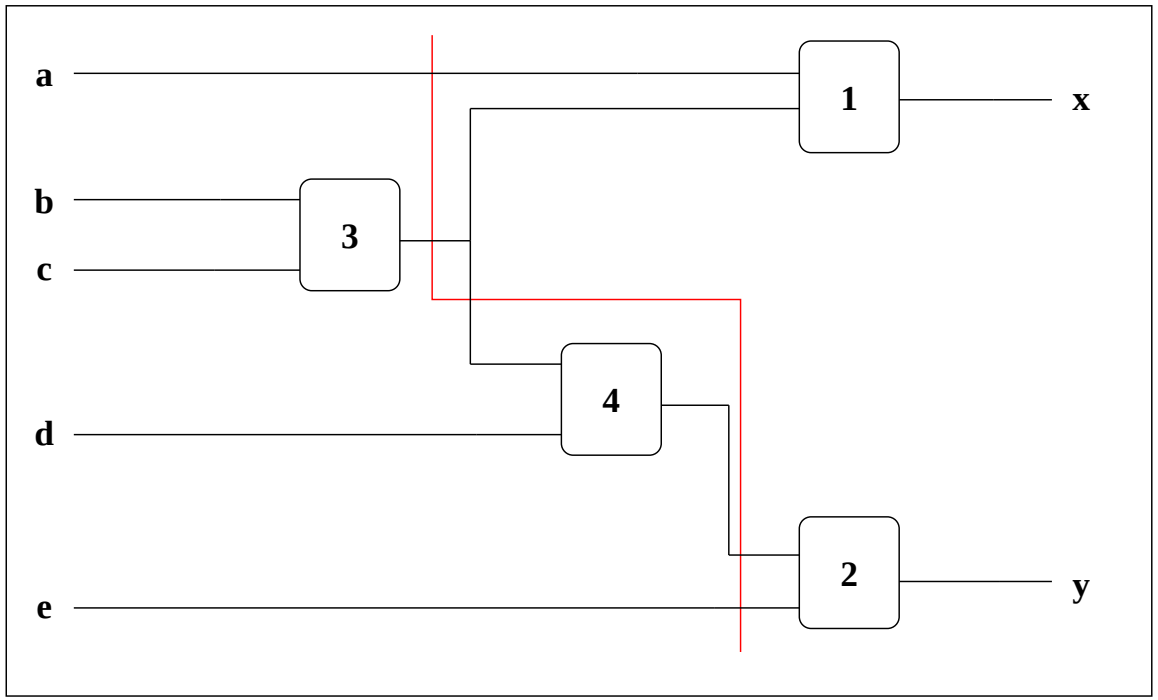
Beginning of the algorithm: the cut crosses all the primary outputs. The *BMD of the word-level interpretation of the output is constructed.

Backward construction algorithm - step 2



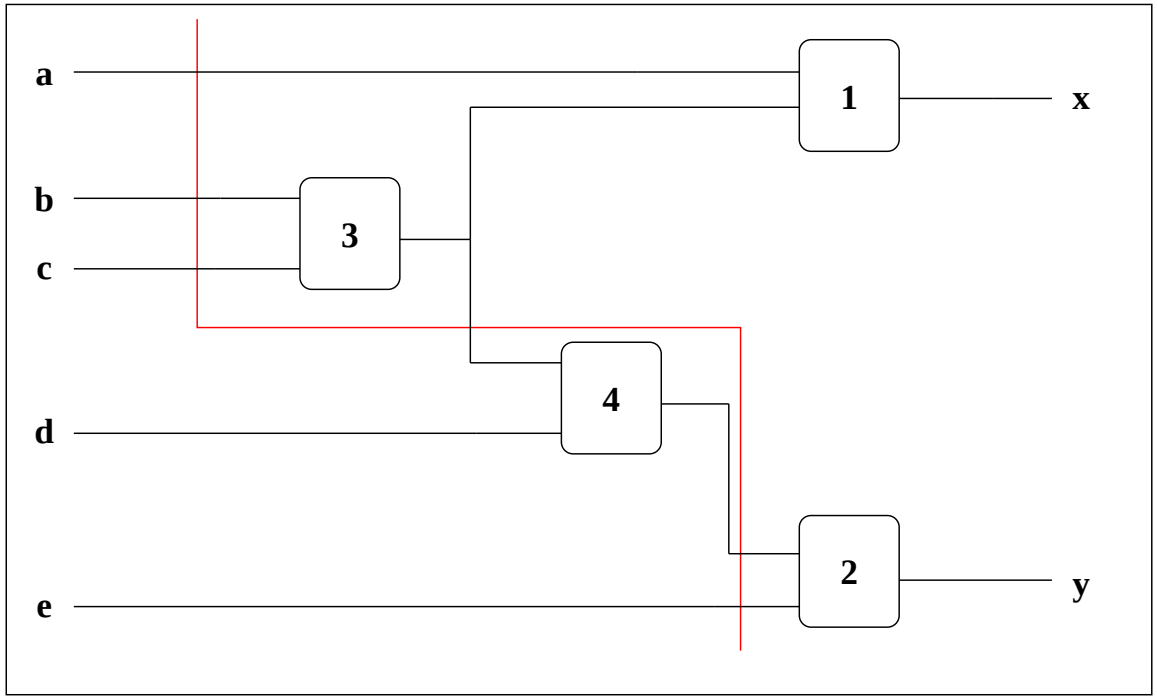
A gate just left to the cut is chosen and its output is substitued in the *BMD by the corresponding function of its inputs.

Backward construction algorithm - step 3



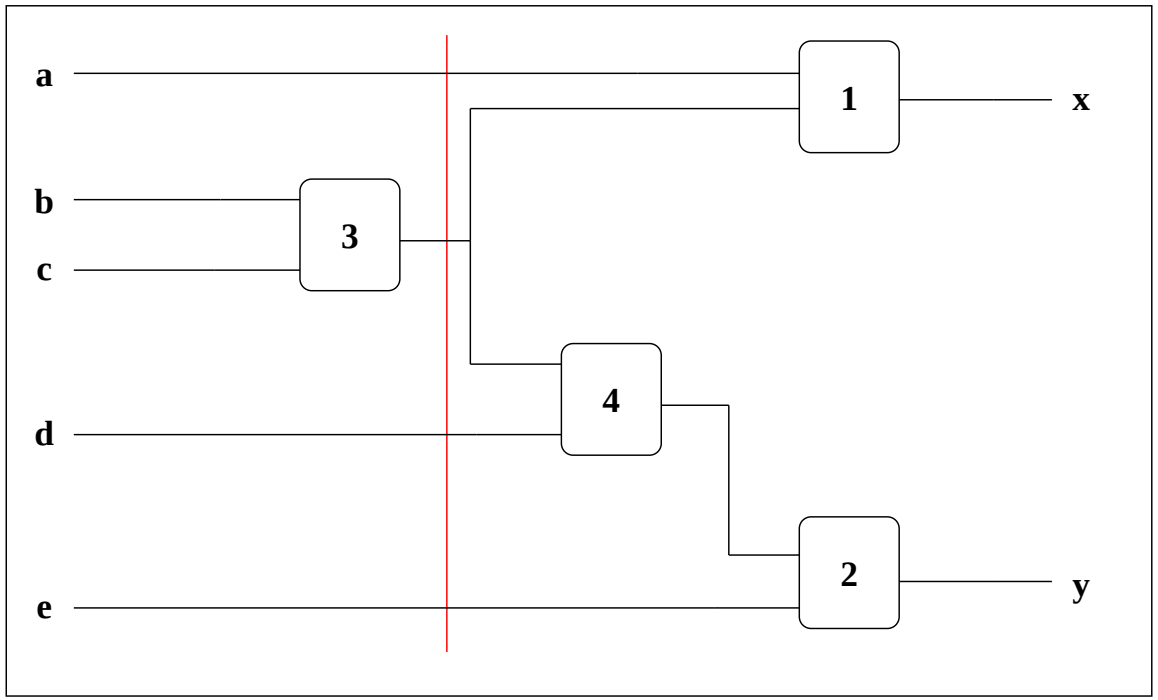
At any time, the *BMD expresses the word-level representation of the output as a function of the nets currently crossed by the cut.

Backward construction algorithm - step 4 (first try)



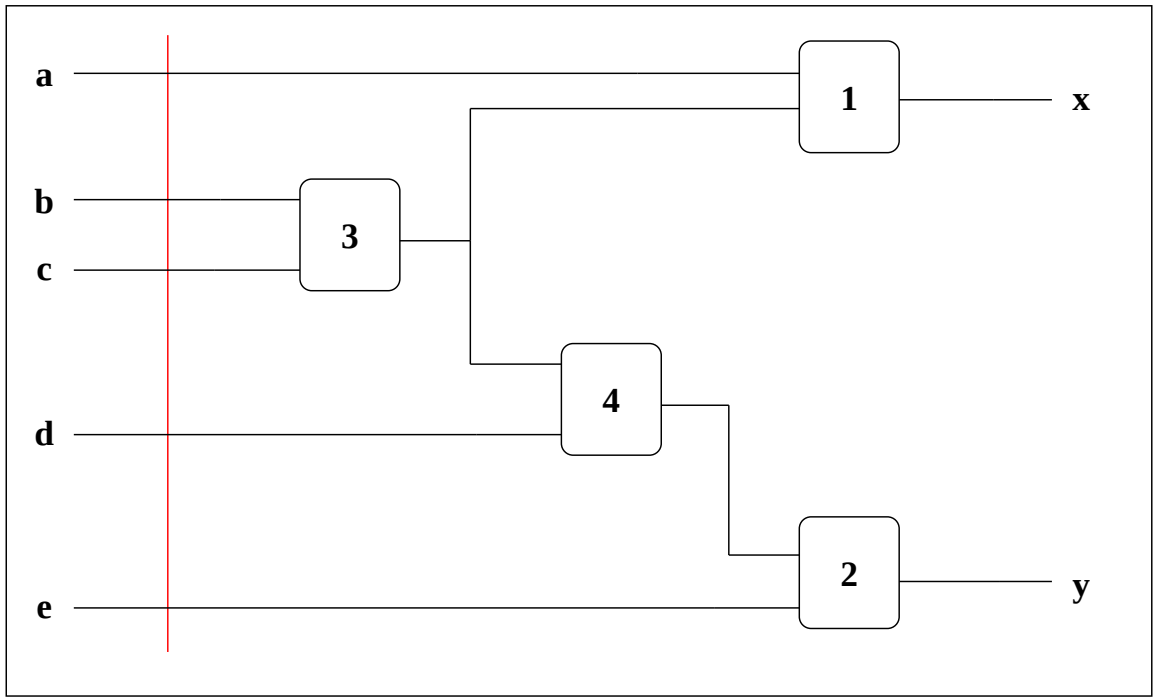
Problem: intermediary results must be kept!

Backward construction algorithm - step 4



Condition: a gate may be chosen only if its output is connected to only the input of the gates that have been already taken.

Backward construction algorithm - step 5



End of the algorithm: the cut crosses all primary inputs. The *BMD expresses the word-level representation of the output as a function of the inputs.

Add-step and carry-save multiplication

$$
\begin{array}{cccccc}
 & & & & 1 & 1 \\
\times & & & 1 & 1 & 1 \\
\hline
 & & & & 1 & 1 \\
+ & & & 1 & 1 & \cdot \\
+ & & 1 & 1 & \cdot & \cdot \\
\hline
 & & 1 & 0 & 0 & 1 \\
+ & & 1 & 1 & \cdot & \cdot \\
\hline
 & 1 & 0 & 1 & 0 & 1
\end{array}
$$

Add-step multiplication

$$
\begin{array}{cccccc}
 & & & & 1 & 1 \\
\times & & & 1 & 1 & 1 \\
\hline
 & & & & 1 & 1 \\
+ & & & 1 & 1 & \cdot \\
+ & & 1 & 1 & \cdot & \cdot \\
\hline
 & & & \{110, & 11\} & \\
+ & & 1 & 1 & \cdot & \cdot \\
\hline
 & & \{1001, & 1100\} & & \\
\hline
 & 1 & 0 & 1 & 0 & 1
\end{array}
$$

Carry-save multiplication

Experimental results

Number of bits	Time Add-step (s)	Time Carry-save (s)
4	1	3
8	12	58
16	161	1115
32	2083	
	$O(n^{3.7})$	$O(n^{4.3})$

(Lava, Sun Server)

Asymptotic $O(n^4)$ bound proved for a large class of classical multipliers [4]

What now?

- Backward Construction Algorithm: very efficient, in comparison with former methods
- Still, need of something better: $O(n^4)$ is too much!
- Completely different direction?

References

[1] Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, August 1986.

[2] Randal E. Bryant and Yirng-An Chen. Verification of arithmetic circuits with binary moment diagrams. In *Design Automation Conference*, pages 535–541, 1995.

[3] K. Hamaguchi, A. Morita, and S. Yajima. Efficient construction of binary moment diagrams for verifying arithmetic circuits. In *Proc. Int'l Conf. on CAD*, pages 78–82, 1995.

[4] Martin Keim, Michael Martin, Bernd Becker, Rolf Drechsler, and Paul Molitor. Polynomial formal verification of multipliers. In *VLSI Test Symposium*, Monterey, USA, 1997.