Boolean Decomposition of Binary Image Operators

HERALDO MACIEL FRANÇA MADEIRA, HÉLIO PEDRINI

UFPR – Universidade Federal do Paraná - Departamento de Informática
Caixa Postal 19081 - CEP: 81531-990 Centro Politécnico - Jd. Américas, Curitiba - PR, Brasil
{heraldo,helio}@inf.ufpr.br

Abstract. Boolean expansion of binary functions has been used in the representation of binary image operators. This representation scheme induced the creation of image processing architectures based on decision diagrams. In some situations, the application of this graph-based operator over the input images is faster than the corresponding conventional approach, in that the operators are constructively described by means of basic operators and set operations. However, when the conventional architecture exploits the bit parallelism of the microprocessor logical instructions, its performance often surpasses that of the graph-based system. This article proposes a novel approach, in which the decision diagrams guide the actions of the underlying conventional architecture, taking advantage of its parallelism, so that faster image processing can be achieved.
Keywords: Boolean decomposition; decision diagram; binary image operator; image processor architecture.

1 Introduction

Conventional binary image processors are implemented in software by supplying fast procedures for primitive operators, such as dilations and erosions, and tools for combining them by means of set theory operations. Usually, several pixels are treated simultaneously in these systems by using the microprocessor bitwise logical operations, resulting, thus, in faster implementations. Although the word length of these bitwise operations may be of several bytes, such image processing systems are called *byte-architecture*, in contrast to the *bit-architecture*, where pixels are processed one at a time.

Alternative architectures have been proposed. For example, in the context of automatic learning of operators, the operator is effectively treated when represented by data collections known as the *operator basis*[2, 3, 4]. Recently, specific operators [11] and a general-purpose architecture [10] based on a graph representation were proposed, offering superior performance over the conventional architecture in many situations.

Binary image operators can be represented by a special kind of direct acyclic graph known as *binary decision diagram*, or BDD [1]. The evaluation of such operators — that is, finding the image resulted from the application of the operator over the input image — is done by traversing the graph and performing an `if-then-else` test on the binary variable stored at each visited node. The value of this variable is directly related to the input image. The result of this traversal is one of the two possible final states of the graph, the sinks "0" and "1", which is then used to determine the output image pixel values.

Image processors using this approach evaluate the input image in time $\mathcal{O}(h \cdot N)$, where h is the longest path in the operator graph (*i.e.* the height of the decision diagram) and N is the input image size in pixels. It is not possible to perform a word-at-a-time parallel processing of pixels using this approach without traversing both branches of the `if-then-else` tests. Thus, although the graph-based architecture presents better performance than the conventional bit-system, in large word machines the conventional architecture is preferable for most operators.

This article demonstrates how the graph-based representation of operators can still be used to speed-up the conventional byte-architecture. An operator application time of $\mathcal{O}(n \cdot \frac{N}{w})$ will be achieved, where n is the size of the decision diagram, and w is the microprocessor word size. This approach leads to faster operator implementations when its graph has a reasonable width. The proposed system will be called *hybrid architecture*.

Following this introduction, Section 2 revisits the set theory approach to binary image processing. Section 3 presents the Boolean expansion and the graph-based representation of binary functions. Section 4 discusses the graph-based representation of windowed operators, how to compute it, and its first employment in the architecture of image processors. Section 5 presents the Boolean decomposition of operators and its use for faster software implementations of image processors. Section 6 shows some examples of operator implementation using the Boolean decomposition structure and makes an expected performance analysis in terms of the number of basic operations. Section 7 concludes the article.

2 Binary Image Operators

This section revisits the ground set theory approach to binary image processing. Binary images are modeled as sub-

sets of the image space E, which commonly is a rectangle in $\mathbb{Z}^2$. The space $\mathcal{P}(E)$ of all binary images constitutes, with the partial order $\subseteq$, a complete Boolean lattice [5], with universal bounds E and $\emptyset$, and having the usual $\cup$, $\cap$ and $\cdot^c$ as its supremum, infimum and complementation operations.

A binary image processing program is modeled as the discrete set operator $\psi : \mathcal{P}(E) \to \mathcal{P}(E)$. The set Ψ of all such operators is also a complete Boolean lattice when one assigns for each $\psi_1, \psi_2 \in \Psi$,

$$\psi_1 \leq \psi_2 \Leftrightarrow \psi_1(X) \subseteq \psi_2(X), \forall X \in \mathcal{P}(E).$$

The supremum and infimum in this lattice verify

$$(\bigvee_{i\in I} \psi_i)(X) = \bigcup_{i\in I} \psi_i(X) \;\; and \;\; (\bigwedge_{i\in I} \psi_i)(X) = \bigcap_{i\in I} \psi_i(X),$$

where I is a set of indices.

The composition of ψ_1 by ψ_2, denoted $\psi_2\psi_1$ is the operator defined by $\psi_2\psi_1(X) = \psi_2(\psi_1(X)), \forall X \in \mathcal{P}(E)$. The identity ($\iota$) and negation ($\nu$) operators are defined by $\iota(X) = X$ and $\nu(X) = X^c, \forall X \in \mathcal{P}(E)$. Thus, $\nu\psi$ is the complement of ψ in the lattice $(\Psi, \leq)$.

The set operators $\delta \in \Psi$ that satisfy $\delta(\cup_{i\in I} X_i) = \cup_{i\in I}\delta(X_i)$, for every family $(X_i)_{i\in I}$ in $\mathcal{P}(E)$ are called *dilations*. Dually, the set operators $\varepsilon \in \Psi$ that satisfy $\varepsilon(\cap_{i\in I} X_i) = \cap_{i\in I}\varepsilon(X_i)$, for every family $(X_i)_{i\in I}$ in $\mathcal{P}(E)$, are called erosions. Dilations and erosions are characterized by *structuring functions* $b : E \to \mathcal{P}(E)$, and denoted δ_b and ε_b, respectively.

With erosions, dilations, the negation operator, composition and the basic set operations, one can build any operator of Ψ. *Morphological expressions* — that is, the valid sentences composed of the symbols $(\delta, \varepsilon, \nu, \wedge, \vee)$ and the parametric structuring functions — are the most common representation of operators. This representation scheme induced the architecture of the conventional binary image processors, also called *morphological machines*, or *MMachs* [2].

Practical implementations of MMachs uses, nevertheless, a sub-family of erosions and dilations: those that commute with the translation, as explained in the sequence.

Let $(E, +)$ be a commutative group, with the zero element $o \in E$, called *origin*. Let $h \in E$ and $X, Y \subseteq E$. The set $X_h = \{x + h : x \in X\}$ is the *translation* of X by h. The set operator τ_h defined by $\tau_h(X) = X_h, \forall X \in \mathcal{P}(E)$, is called the *translation* operator *by* h. An operator ψ is said *translation invariant* if it commutes with the translation, that is, if $\tau_h\psi = \psi\tau_h, \forall h \in E$.

The set operations $X \oplus Y = \cup_{y\in Y}(X_y)$ and $X \ominus Y = \cap_{y\in Y}(X_{-y})$ are the *Minkowski addition* and *subtraction*. The translation invariant dilations and erosions are characterized by sets $B \subseteq E$, called *structuring elements*, and, thus, denoted by δ_B and ε_B. They can be defined by

$$\delta_B(X) = X \oplus B \;\; \text{and} \;\; \varepsilon_B(X) = X \ominus B \;\; \forall X, B \in \mathcal{P}(E).$$

Note that $\delta_B = \vee_{b\in B}\, \tau_b$ and $\varepsilon_B = \wedge_{b\in B}\, \tau_{-b}$.

Let $W \subseteq E$ be a finite set called *window*. A set operator ψ is said to be *locally defined within* W if and only if

$$h \in \psi(X) \Leftrightarrow h \in \psi(X \cap W_h),\ \forall h \in E, \forall X \in \mathcal{P}(E).$$

We call *operator window* the minimal window in which the operator is locally defined. Thus, the translation invariant set operators ι, ν, τ_h, δ_B and ε_B have windows $\{o\}$, $\{o\}$, $\{-h\}$, $B^t = \{-b : b \in B\}$ and B, respectively.

Let Ψ_W be the set of all translation invariant operators that are also locally defined within some window W. These are the operators of interest in this article, and are called *windowed operators*, or W-operators. The lattice $(\Psi_W, \leq)$ is isomorphic to the complete Boolean lattice $(\mathcal{P}(\mathcal{P}(W)), \subseteq)$ [3]. This means that W-operators can be canonically (*i.e.*, completely and uniquely), represented by collections $\mathcal{X} \subseteq \mathcal{P}(W)$ of window configurations. One such representation is the operator *kernel*, given by $\mathcal{K}_W(\psi) = \{X \subseteq W : o \in \psi(X)\}$.

Another useful canonical representation is the collection of all maximal intervals in the kernel, called *basis* of the operator. An interval of window configurations with extremities A and B is defined by $[A, B] = \{X : A \subseteq X \subseteq B \subseteq W\}$. The basis representation is widely used in automatic design of operators by machine learning techniques [2, 3, 4].

3 Characteristic Function and Decision Diagrams

Besides the kernel and basis representations, W-operators can be canonically represented by functions $\phi : \mathcal{P}(W) \to \{0, 1\}$ and by functions $f : \{0, 1\}^{|W|} \to \{0, 1\}$, since $\{0, 1\}^{\mathcal{P}(W)}$ and $\{0, 1\}^{\{0,1\}^{|W|}}$ (the Boolean algebra $\mathbb{B}_{|W|}$) are Boolean lattices isomorphic to $\mathcal{P}(\mathcal{P}(W))$, and hence, to Ψ_W. The supremum, infimum and complement in those lattices are denoted by $+$, $\cdot$ and $\overline{\cdot}$.

The *characteristic function* of an operator ψ, is the function of $\{0, 1\}^{\mathcal{P}(W)}$ defined by $\phi(\psi)(X) = 1 \Leftrightarrow X \in \mathcal{K}_W(\psi) \Leftrightarrow o \in \psi(X), \forall X \in W$.

In the following, we define the characteristic *binary* function of a W-operator ψ, denoted $f(\psi) \in \mathbb{B}_{|W|}$. Let $W = \{w_1, w_2, \ldots w_{|W|}\}$. The *support* of a vector $\mathbf{x} = (x_1, x_2, \ldots, x_{|W|}) \in \{0, 1\}^{|W|}$, denoted $\underline{\mathbf{x}}_W$, is the set $X \subseteq W$ such that $w_i \in X \Leftrightarrow x_i = 1$. The characteristic binary function of an operator ψ, is the function of $\mathbb{B}_{|W|}$ defined by $f(\psi)(\mathbf{x}) = \phi(\underline{\mathbf{x}}_W), \forall \mathbf{x} \in \{0, 1\}^{|W|}$.

Binary functions of $|W|$ variables can be canonically represented in several ways. The sum-of-minterms form

of the characteristic function of an operator is structurally equivalent to its kernel. The set of prime implicants of this function is structurally equivalent to the operator basis.

A graph-based representation for binary functions, called *binary decision diagram* has been extensively used since the work of Bryant [7], when a canonical form and manipulation algorithms for it were defined. It is based in the *Boolean expansion* of a binary function, given as follows.

Let $f \in \mathbb{B}_{|W|}$, and x_i a variable in the domain of f. The Boolean expansion of f with respect to the variable x_i is

$$f \;=\; x_i \cdot f|_{x_i} + \overline{x_i} \cdot f|_{\overline{x_i}} \;=\; (x_i + f|_{\overline{x_i}}) \cdot (\overline{x_i} + f|_{x_i}).$$

The restrictions $f|_{x_i} = f(x_1, \ldots, x_i = 1, \ldots, x_{|W|})$ and $f|_{\overline{x_i}} = f(x_1, \ldots, x_i = 0, \ldots, x_{|W|})$ are, respectively, the *positive* and *negative cofactors* of f with respect to the variable x_i.

The cofactors can be expanded further with respect to other variables, and this process repeated until the remaining cofactors be the constant functions $\mathbf{0}$ and $\mathbf{1}$. The final expression, when parsed, results in an ordered binary tree, whose nodes are labeled with the variable indices, and the leaves with "0" or "1". The first child of each node is called positive, and the other child, negative.

A parsing tree for the function can be built with this procedure in a way that the paths from the root to the leaves visit the node indices in ascending order. Afterwards, all identical subtrees can be merged and all the nodes whose children are identical be eliminated. The resulting structure is a rooted directed acyclic graph called *reduced ordered binary decision diagram*, or ROBDD [7].

ROBDD is a canonical representation of binary functions. Several free software packages for ROBDD manipulation are available in the Internet. In [6, 7, 8] algorithms to compute the ROBDD of $f_1 + f_2$ and of $f_1 \cdot f_2$, given the ROBDDs of f_1 and f_2, are presented.

Figure 1 shows the ROBDDs of some binary functions. Solid lines point to the positive cofactors and dashed to the negative ones.

The evaluation of $f(x_1, x_2, \ldots, x_{|W|})$ can be stated as "`if` (x_i) `then` $f|_{x_i}$ `else` $f|_{\overline{x_i}}$". Therefore, the ROBDD can be "executed" as if it were a program consisting of nested `if-then-else` statements. The running time of this program is $\mathcal{O}(|W|)$, meaning that the function can be evaluated with at most one test per variable.

4 Graph Representation of Operators

This section discusses the graph-based architecture and the method to compute the operator graph.

Let $\leq$ a total order in E and $W = \{w_1, w_2, \ldots, w_{|W|}\}$ such that $w_1 \leq w_2 \leq \ldots \leq w_{|W|}$. Let $\mathbf{G}_W$ be the

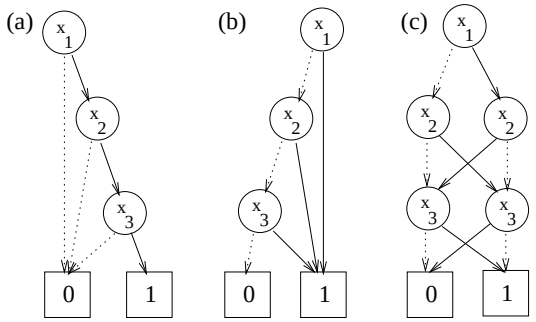

Figure 1: ROBDDs of some Boolean functions: (a) $x_1 \cdot x_2 \cdot x_3$; (b) $x_1 + x_2 + x_3$; (c) $x_1 \oplus x_2 \oplus x_3$.

space of all ROBDDs representing binary functions in the variables $x_{w_1}, x_{w_2}, \ldots, x_{w_{|W|}}$. As ROBDDs are canonical representation of binary functions, a bijective mapping between $\mathbf{G}_W$ and $\mathbb{B}_{|W|}$ exists.

Let $\mathcal{G} : \mathbb{B}_{|W|} \to \mathbf{G}_W$ denote the mapping that takes the ROBDD $\mathcal{G}(f)$ of the function f. Let $f_1, f_2 \in \mathbb{B}_{|W|}$. By defining $\mathcal{G}(f_1) \sqcup \mathcal{G}(f_1) = \mathcal{G}(f_1 + f_2)$ and $\mathcal{G}(f_1) \sqcap \mathcal{G}(f_1) = \mathcal{G}(f_1 \cdot f_2)$, we establish a lattice isomorphism between $\mathbf{G}_W$ and $\mathbb{B}_{|W|}$, and hence, between $\mathbf{G}_W$ and Ψ_W. The ROBDD of the characteristic function of an operator ψ is called *graph of* ψ, that is, $\mathcal{G}(f(\psi))$, and denoted by $\mathcal{G}(\psi)$

The graphs of the elementary operators ι, ν, τ_h, δ_B and ε_B are denoted by I, N, T_h, Δ_B and E_B. They have a very simple structure, as illustrated in Figure 2.

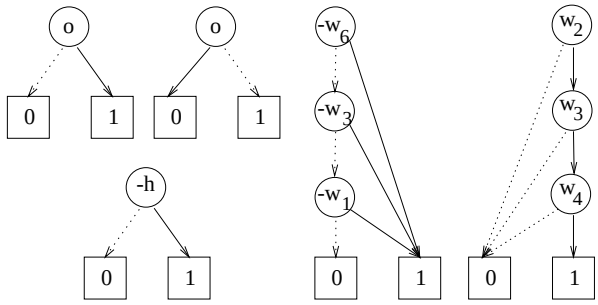

Figure 2: Graphs of ι, ν, τ_h, $\delta_{\{w_1, w_3, w_6\}}$ and $\varepsilon_{\{w_2, w_3, w_4\}}$.

Graphs of complex operators can be built by means of the procedures presented in [9], when one knows the operator description by a morphological expression. Those procedures compute the following graphs, given $\psi \in \Psi_W$, $\psi_1 \in \Psi_{W_1}$, $\psi_2 \in \Psi_{W_2}$, $h \in E$ and $B \subseteq W$:

- $\mathcal{G}(\delta_B \psi) \in \mathbf{G}_{W \oplus B^t}$
- $\mathcal{G}(\tau_h \psi) \in \mathbf{G}_{W_{-h}}$
- $\mathcal{G}(\varepsilon_B \psi) \in \mathbf{G}_{W \oplus B}$
- $\mathcal{G}(\psi_2 \psi_1) \in \mathbf{G}_{W_1 \oplus W_2}$.

Let $G_i = \mathcal{G}(\psi_i)$. The graph $\mathcal{G}(\psi_2 \psi_1)$ will be denoted by $G_2 \cdot G_1$. Thus, the procedures above calculate $T_h \cdot G$,

$E_B \cdot G$, $\Delta_B \cdot G$ and the generic composition $G_2 \cdot G_1$. The procedures for $\sqcup$, $\sqcap$ and $\overline{G} = N \cdot G$ can be those given in [6, 7, 8]. With this notation, the procedure of converting a morphological expression into its graph is straightforward. For example, the graph of $\varepsilon_A \delta_A \wedge (\varepsilon_B \delta_B \vee \iota \vee \nu \wedge \tau_h)$ is obtained from the application of the following sequence of graph operations: $(E_A \cdot \Delta_A) \sqcap [(E_B \cdot \Delta_B) \sqcup I \sqcup (N \sqcap T_h)]$.

A morphological machine has been built having the ROBDD representation in its core [10]. A pre-processor calculates the operator graphs, translate them into `if-then-else`-formed programs, compile them, and store them in a library. The image processing itself is done by directly executing these programs as the machine primitives. Set operations, counters and loop control are also provided to allow the use of more complex operators. This approach has reported faster implementations than in the conventional bit-architecture for many operators [9].

5 Boolean Decomposition of W-Operators

This section presents the main contributions of this article: the Boolean decomposition of operators and the proposal of using it in morphological machines in order to achieve faster implementations.

Recall the Boolean expansion of a function $f \in \mathbb{B}_{|W|}$ with respect to the variable x_{w_i}:

$$\begin{aligned} f &= (x_{w_i} \cdot f|_{x_{w_i}}) + (\overline{x_{w_i}} \cdot f|_{\overline{x_{w_i}}}) \\ &= (x_{w_i} + f|_{\overline{x_{w_i}}}) \cdot (\overline{x_{w_i}} + f|_{x_{w_i}}). \end{aligned}$$

Let $\psi \in \Psi_W$ be the operator with characteristic binary function f. Let denote $\psi|_{w_i}$ and $\psi|_{\overline{w_i}}$ the operators whose functions are $f|_{x_{w_i}}$ and $f|_{\overline{x_{w_i}}}$, respectively. It is easily verified that x_{w_i} is the characteristic binary function of the operator τ_{-w_i}. Therefore, making use of the isomorphism between $\mathbb{B}_{|W|}$ and Ψ_W, we obtain what we call the *Boolean decomposition of ψ at w_i*:

$$\begin{aligned} \psi &= (\tau_{-w_i} \wedge \psi|_{w_i}) \vee (\nu\tau_{-w_i} \wedge \psi|_{\overline{w_i}}) \\ &= (\tau_{-w_i} \vee \psi|_{\overline{w_i}}) \wedge (\nu\tau_{-w_i} \vee \psi|_{w_i}). \end{aligned}$$

Similarly, in the space G_W we also define the *Boolean decomposition of the graph $G = \mathcal{G}(\psi)$ at w_i*:

$$\begin{aligned} G &= (T_{-w_i} \sqcap G|_{w_i}) \sqcup (N \cdot T_{-w_i} \sqcap G|_{\overline{w_i}}) \\ &= (T_{-w_i} \sqcup G|_{\overline{w_i}}) \sqcap (N \cdot T_{-w_i} \sqcup G|_{w_i}), \end{aligned}$$

where the restrictions $G|_{w_i}$ and $G|_{\overline{w_i}}$ are the graphs of the corresponding operators $\psi|_{w_i}$ and $\psi|_{\overline{w_i}}$. Note that, if w_i is the label of the graph root (the smallest variable index in G), then these graphs coincide with the successors of the root node.

From the disjunctive Boolean decomposition of ψ it follows that, $\forall X \in E$,

$$\psi(X) = (X_{-w_i} \cap \psi|_{w_i}(X)) \cup (\overline{X_{-w_i}} \cap \psi|_{\overline{w_i}}(X))$$

This expression can be interpreted as reproducing in $\psi(X)$ the pixels of $\psi|_{w_i}(X)$ at the positions set in X_{-w_i}, and the pixels of $\psi|_{\overline{w_i}}(X)$ at the other positions. An alternative notation to this is the *bitwise selection*, given by the "?:" ternary operation:

$$\psi(X) = X_{-w_i} \,?\, \psi|_{w_i}(X) : \psi|_{\overline{w_i}}(X).$$

Note that this is not an `if-then-else` in the usual sense, since both branches have to be computed, in order to get $\psi(X)$.

We are now ready to introduce an enhancement of MMachs, which consists of a procedure to drive the application of the operator over input images by exploiting this new decomposition. We suppose that the graphs of the operators are given — if not, they can be calculated from the morphological expression.

The procedure recursively traverses the operator graph, and at each internal node applies the Boolean decomposition expression to the images returned from its children nodes. The leaves are defined to return the empty (all pixels off) and full (all pixels on) images. The decomposition expression is calculated by the usual calls to the underlying conventional MMach (byte-architecture). The advantage of this method is its simplicity, since it demands no pixel-level routine development. The decomposition expression can also be supplied as a routine that extends the functionality of the MMach, allowing direct specification of ROBDDs by the operator designer. The main drawback of this approach is that one entire image is allocated for each instance of the recursion, but this is easily overcome as will be seen in the sequel.

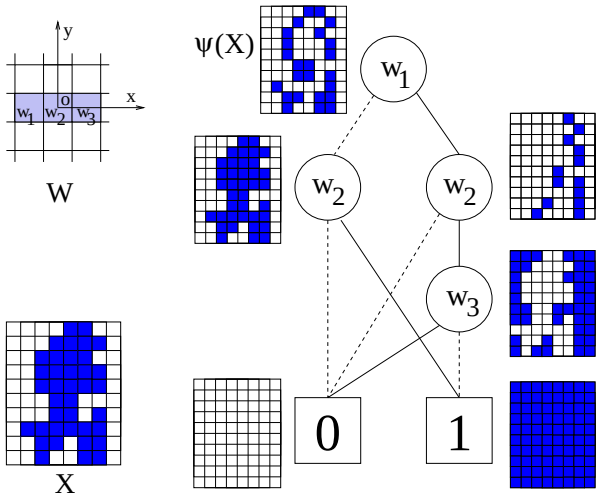

Figure 3: Graph of the vertical border detector ψ and the intermediate images used to compute $\psi(X)$.

Figure 3 illustrates the graph of the anti-extensive vertical border detector. Besides each node, it is shown the

processed image returned by it, given the input image X. The root node returns the image $\psi(X)$, as shown.

A first optimization of that procedure is possible, in which all needed translations of the image is precomputed, as well as their complements, and then the recursive procedure is built by making use of a fast implementation of the bitwise selection operation. A further optimization with respect to memory space can be achieved by doing the whole process in image chuncks, for example in one row at a time.

This new approach to morphological image processing, depending on the viewpoint, is regarded as an extended conventional MMach, or a parallel graph-based MMach. We call this approach *hybrid architecture*.

6 Expected Performance

This section shows the applicability of the architecture based on the Boolean decomposition structure. We present three classes of operators that run faster when decomposed in this way, when compared with the classical decomposition.

We assume that in any machine implementation the needed image translations ($\tau_{-w_i}(X)$) and their complements ($\nu\tau_{-w_i}(X)$) are previously calculated, or are calculated only once, when they appear. Hence, for a W-operator there are $|W|$ input image translations and $|W|$ complemented ones, independently of the decomposition used. So, for the sake of architecture comparisons, the number of operations $\cup$ plus the number of $\cap$'s will be considered as the algorithm cost. Thus, the costs of ε_B or δ_B are $|B|-1$. The cost of any operator described by the Boolean decomposition is the number of internal nodes n times the cost of the Boolean decomposition expression, that is, $3 \times n$.

Preliminarily, we analyse the cost of the hit-or-miss operator, which is extensively used in morphological image processing. It is defined as $\lambda'_{H,M} = \varepsilon_H \wedge \varepsilon_M \nu$. Its cost is, therefore, $|H| + |M| - 1$.

The first operator is the parity operator, named π_W. This operator examines the input image under its window W and turns on the output pixel at the appropriate position if the observed image configuration through the window has an odd number of points. Its description by the classical morphological decomposition is a supremum of $2^{|W|-1}$ terms of the form $\lambda'_{A_i,\overline{A_i}}$, where the A_i's are the subsets of W with odd size and $\overline{A_i} = W \cap A_i^c$. Thus, its cost is exponential in the size of W. Except for the case of a very small window ($|W| \leq 3$), it has worse performance than the implementation by the Boolean decomposition, since the graph of this operator has only $2 \cdot |W| - 1$ nodes. Figure 4 illustrates the set of the A_i's for π_W and the operator graph.

The second operator is the symmetry detector. This operator examines the input image under its window W and

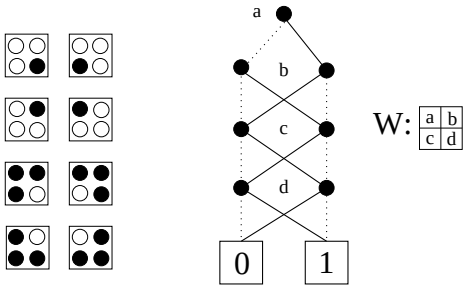

Figure 4: Structuring elements and graph for the parity operator ($|W| = 4$).

turns on the output pixel at the appropriate position, if the observed window configuration is symmetric with respect to the center. It is defined by the formula:

$$\sigma_W = \vee_{B \in \mathcal{S}} \lambda'_{B,\overline{B}},$$

where $\mathcal{S}$ is the set of the $2^{\frac{|W|}{2}}$ possible symmetric structuring elements contained in W.

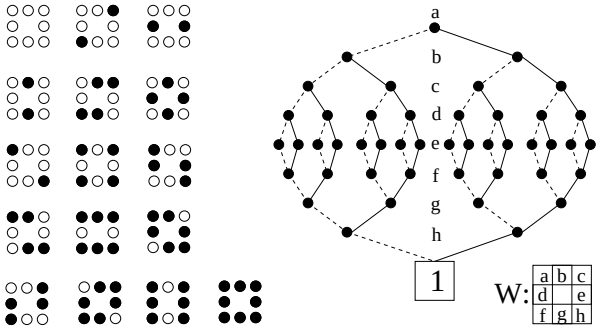

Figure 5: Structuring elements and graph for the symmetry detector ($|W| = 8$).

Figure 5 shows the set $\mathcal{S}$ as well as the operator graph, for a window with eight elements. Arcs to the sink "0" were omitted in the drawing.

The cost of $\lambda_{B,\overline{B}}$ is $|W| - 1$. Hence, the cost of σ_W is $(|W| - 1) \cdot |\mathcal{S}| + (|\mathcal{S}| - 1)$, that is, $|W| \cdot 2^{\frac{|W|}{2}} - 1$. The corresponding graph-based operator has $3 \cdot 2^{\frac{|W|}{2}} - 3$ nodes, then costing $9 \cdot (2^{\frac{|W|}{2}} - 1)$. For windows with ten or more elements, the performance of the graph-based implementation of σ_W gets better than the implementation by the flat decomposition above.

The third operator is the median filter. It is defined as:

$$\mu_B(X) = \left\{x \in E : |X_{-x} \cap B| \geq \frac{|B|+1}{2}\right\},\ \forall X \subseteq E$$

The kernel of μ_B consists of all subsets of B that have $\frac{|B|+1}{2}$ elements or more, supposing $|B|$ odd. Its basis consists of the $\binom{|B|}{\frac{|B|+1}{2}}$ intervals $[A, B]$ such that $|A| = \frac{|B|+1}{2}$.

The morphological expression derived from the basis of the median filter is: $\mu_B = \vee\{\varepsilon_A : A \subseteq B, |A| = \frac{|B|+1}{2}\}$. The implementation of this expression will cost $\binom{|B|}{\frac{|B|+1}{2}} \cdot \frac{|B|+1}{2} - 1$. We note that this implementation is too costly. For example, if $|B| = 9$ there are 126 erosion terms, and μ_B costs 629 operations.

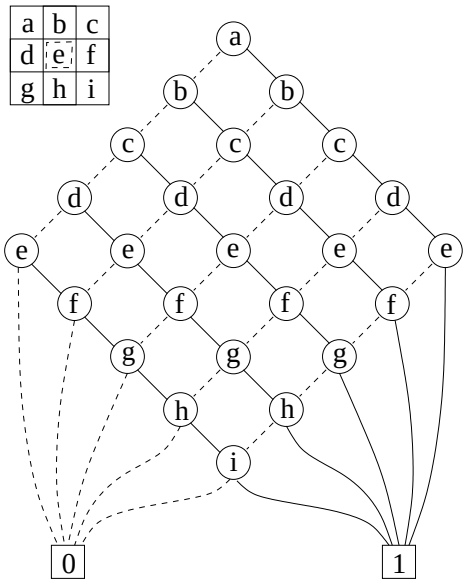

Figure 6: ROBDD for a 3×3 median filter.

The ROBDD for the median filter has a very simple structure, and the example for $|B| = 9$ is shown in Figure 6. The cost of μ_B in the proposed MMach is $3 \cdot (\frac{(|B|+1)}{2})^2$, which is obviously much more efficient than the implementation based on the basis decomposition. Thus, for $|B| = 9$ μ_B costs 75 operations. In a graph-base MMach it would cost 9. In a 32-bit machine, the expected speed-up of the new approach is $(9 \cdot 32)/75 = 3.84$.

7 Conclusion

Using the lattice isomorphism that exists between the space of W-operators and the space of the binary functions of $|W|$ variables, a decomposition structure in Ψ_W equivalent to the Boolean expansion was derived. This was named the operator "Boolean decomposition at the point h", with $h \in W$.

The components of the decomposition expression are the operators τ_h and $\nu\tau_h$, which are trivially calculated, and the restrictions $\psi|_{w_i}$ and $\psi|_{\overline{w_i}}$, which are directly available in a certain recursive procedure. The combining operations are also simple to calculate (infimum and supremum). All of these operations can exploit the natural parallelism of the microprocessor logical instructions and, thus, calculate several pixels at a time.

The mentioned recursion is guided according to the graph of the operator, that is, its ROBDD, whose construction also comes from the Boolean expansion. Depending on the width of the microprocessor word and on the number of nodes of the graph, the operator implementation using this decomposition can have superior performance than the usual, and than its implementation in graph-based architectures, as well.

Acknowledgements

The authors would like to thank UFPR for the support during the development of this work.

References

[1] S. B. Akers. Binary Decision Diagrams. *IEEE Transactions on Computers*, C-27(6):509–516, June 1978.

[2] J. Barrera, F. S. C. da Silva, and G. J. F. Banon. Automatic Programming of Binary Morphological Machines. In *Image Algebra and Morphological Image Processing*, volume 2300 of *Proc. of SPIE*, pages 229–240, San Diego, 1994.

[3] J. Barrera and G. P. Salas. Set Operations on Closed Intervals and Their Applications to the Automatic Programming of Morphological Machines. *Electronic Imaging*, 5(3):335–352, July 1996.

[4] J. Barrera, R. Terada, R. Hirata Jr, and N. S. T. Hirata. Automatic Programming of Morphological Machines by PAC Learning. *Fundamenta Informaticae*, 41(1-2):229–258, Jan. 2000.

[5] G. Birkhoff. *Lattice Theory*. American Mathematical Society, Providence, Rhode Island, 3rd edition, 1967.

[6] K. S. Brace, R. L. Rudell, and R. E. Bryant. Efficient Implementation of a BDD Package. In *Proc. of 27th ACM/IEEE Design Automation conference*, pages 40–45, 1990.

[7] R. E. Bryant. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, Aug. 1986.

[8] K. Karplus. Representing Boolean Functions with If-Then-Else DAGs. Technical Report UCSC-CRL-88-28, UCSC, Nov. 1988.

[9] H. M. F. Madeira and J. Barrera. Incremental Evaluation of BDD-Represented Set Operators. In P. C. Carvalho and M. Walter, editors, *Proc. XIII Brazilian Symposium on Computer Graphics and Image Processing*, pages 308–315, Gramado, Oct. 2000. IEEE.

[10] H. M. F. Madeira, J. Barrera, R. Hirata Jr, and N. S. T. Hirata. A New Paradigm for the Architecture of Morphological Machines: Binary Decision Diagrams. In J. Stolfi and C. L. Tozzi, editors, *Proc. XII Brazilian Symposium on Computer Graphics and Image Processing*, pages 283–292, Campinas, Nov. 1999.

[11] L. Robert and G. Malandain. Fast Binary Image Processing Using Binary Decision Diagrams. *Computer Vision and Image Understanding*, 72(1):1–9, October 1998.