Semiautomatic Segmentation of Transistor Gates in Integrated Chips

Piali Das[1], Olga Veksler[1], Vyacheslav Zavadsky[2], and Yuri Boykov[1]

[1] University of Western Ontario
London, Ontario, Canada
{pdas4,olga,yuri}@csd.uwo.ca
[2] Semiconductor Insight Inc.
Ottawa, Ontario, Canada
vyacheslavz@semiconductor.com

Abstract. In recent years, interactive methods for segmentation are increasing in popularity due to their success in different domains such as medical image processing, photo editing, etc. In this paper we discuss a challenging industrial application of transistor gate segmentation in the images of integrated chips, which is essential for reverse engineering tasks. Segmentation in the domain of integrated chips is very difficult due to large variations in contrast and noise type and also due to extreme variation in the size of the transistor gates, which can range from a few pixels to a few thousands of pixels in length, and from one to several hundred pixels in width. We present a semiautomatic segmentation algorithm that produces reliable and accurate segmentation of a transistor gate from its background with the minimum guidance from the user, who just has to click on one pixel inside the transistor gate of interest. The algorithm is based on the powerful graph-cut interactive segmentation technique of Boykov and Jolly [1]. In order to obtain accurate and robust segmentation with such low user interaction, we make several assumptions based on our observations of the transistor gate images. The main assumption is that the transistor gates are approximately *compact* in shape, or can be approximated by several roughly collinear compact parts. To achieve robustness in segmentation, we incorporate the *compact* shape prior into the framework of [1]. The use of the *compact* shape prior allows us to introduce a parameter *bias* to bias the segmentation towards larger object boundaries, which counteracts the general tendency of the algorithm in [1] to produce smaller segments. In order to accommodate large variation in the quality of the images, most parameters in the algorithm are selected automatically to adapt to the current image. An application developed on the basis of our algorithm runs in real-time and is being used by Semiconductor Insight Inc.

1 Introduction

Segmentation is an important problem in computer vision and is often required as a preliminary step to solving various image analysis tasks. Segmentation is subjective, and thus it is ill-posed in a general setting. However for a particular application segmentation can become tractable, provided that enough problem specific assumptions can be made to simplify the problem.

In this paper, we describe a segmentation application for Semiconductor Insight, which is an engineering consultancy company specializing in intellectual property protection in the integrated circuit domain. To obtain images, the integrated circuit is delayered and SEM micro-photographed. The upper layers of the chip, that contain metal wiring, are typically high quality and can be segmented by automated means. The lower levels, that contain the actual dopant silicon implementation of transistors, are typically low quality, and could have substantial variation in brightness and contrast. Two of the most important parameters in IC circuitry are the length and the width of the transistors. They determine the circuitry power characteristics and are crucial for proper modeling and understanding of the functionality. Prior to the development of the application described in the paper, the width and length measurements were done by a human operator, boxing the actual gate in a computer application, which also involved time consuming panning and zooming across the image. We developed an interactive segmentation system for determining the length and the width of the transistor gates. The system requires the minimum possible user interaction and produces accurate and robust segmentation. The user just has to choose the target transistor by clicking inside it only once. Hence we refer to our application as semiautomatic segmentation.

We chose the graph-cut segmentation algorithm proposed by Boykov and Jolly in [1] as a basic framework for our application. The interactive graph cut [1] is one of the state-of-the-art methods for interactive segmentation. Unlike local methods like region growing or thresholding [2], algorithm in [1] is able to produce globally optimal segments. The global methods typically solve the problem by defining an objective function and optimizing it. Except the graph-cut based method of [1], none of the other optimization based methods like active contour (snake) [3, 4], level sets [5], normalized cut [6] guarantees a globally optimal solution.

The framework in [1] was proposed for a general segmentation and requires the user to mark a few object seeds and a few background seeds. In addition, if segmentation results are not satisfactory, the user has to correct them by adding more object and background seeds. We seek to reduce the interaction to the minimum, the user just needs to choose the transistor gate to be segmented by marking a single seed pixel inside it. The algorithm in [1] is not directly applicable with such a low user interaction. In order to make the graph cut framework applicable, we make several simplifying assumptions based on the observed images. Consider the sample images in the *Fig.*1. The common property of the transistor gates is that they are nearly rectangular in shape. Hence, we incorporate the *compact* shape prior in the framework of [1] which constrains the segmentation

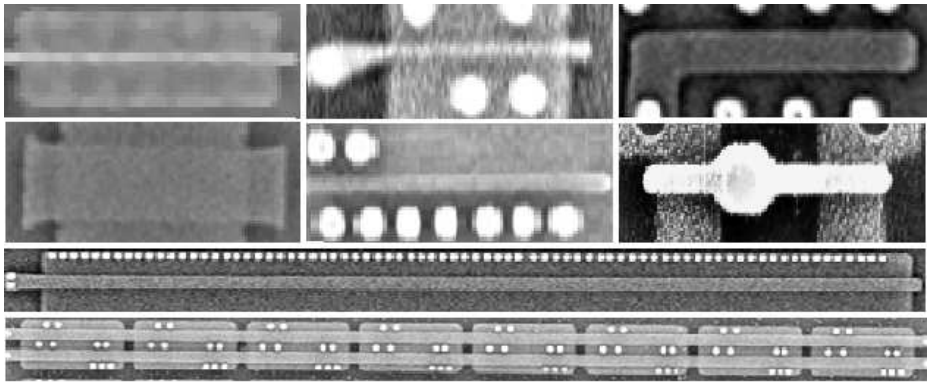

Fig. 1. Sample of the images provided by Semiconductor Insight Inc.

to follow the compact shape[3]. Another observation is that the transistor gates appear brighter than their background. Incorporating these assumptions into the framework of [1] helps achieve robust and accurate segmentation.

A major difficulty in our application arises from the large variability in the sizes of the transistor gates. They can range from 2 to 200 pixels in width and from 10 to a few thousands of pixels in length. In addition, there is a large variation in quality of the images. The contrast between the transistor gate and its background is frequently poor, and the intensity within a single transistor gate may vary significantly. Moreover, the noise level varies from image to image. *Fig.*1 shows some of the images. Accurate estimation and adaptation of the algorithm's parameters to a specific image is hence essential for robust segmentation.

In [1], the user is required to decide upon the quality of the segment and, if necessary, correct it by repeatedly adding new seeds and rerunning the graph-cut step. However the algorithm in [1] is sensitive to the choice of parameters, and if they are far from optimal, significant interaction may be required from the user to obtain the desired segmentation. Unfortunately, automatic parameter estimation for [1] is not a solved problem yet. In our application, we solve the parameter selection issue as follows. We devise a simple yet intuitive test for automatically checking the quality of the segment. If the current segment does not pass the quality check, we readjust the parameters and rerun the graph cut segmentation. We iterate this step using a search over parameter space until the resulting segment passes the quality check.

Another issue with [1] is its tendency to produce objects with shorter boundaries. For technical reasons, due to the compact shape prior, we can introduce a new parameter *bias* into the framework of [1] to bias segmentation towards larger

[3] We explain in section 3.4 what we mean by the word *compact*.

objects. The value of this parameter has a large influence on the segmentation results, and we choose it automatically as described in the previous paragraph.

The semiautomatic segmentation system developed on the basis of our algorithm for Semiconductor Insight Inc. has successfully replaced their existing time-consuming and tedious manual segmentation system. This paper is organized as follows. In Sec. 2 we review the graph cut segmentation framework of [1], in Sec. 3 we give the details of our algorithm for transistor segmentation, in Sec. 4 we present results.

2 Segmentation with Graph Cuts

In [1] segmentation of an object from its background is stated as a binary labeling problem. Given a set of pixels P and a set labels $L = \{0, 1\}$, where labels 0 and 1 represent the background and the object, respectively, the goal is to find an assignment of labels to pixels $S = \{S_1, \ldots, S_p, \ldots, S_{|P|}\}$ that minimizes the energy function:

$$E(S) = \alpha \sum_{p \in P} D_p(S_p) + \sum_{\substack{\{p,q\} \in N \\ p<q}} V_{pq}(S_p, S_q), \tag{1}$$

N is the neighborhood system, which is often chosen as the standard 4-connected grid, and p and q are pixels. $D_p(S_p)$ is the penalty for assigning label S_p to the pixel p, and should be small if label S_p is likely for a pixel p. $V_{pq}(S_p, S_q)$ is the pairwise penalty for assigning labels S_p, S_q to neighboring pixels p and q, and should be large if $S_p \neq S_q$ and an object border is unlikely between p and q. D_p is called the regional term and it encodes the regional properties of the segment, while V_{pq} is called the boundary term and it encodes the boundary properties of a segment. The parameter α decides the relative importance between the regional and the boundary properties of the segment. In [1] it is shown how to find the global minimum of the energy in equation (1) using the min-cut/max-flow algorithm. We use the fast max-flow algorithm developed in [7].

3 Transistor Gate Segmentation

Our goal is to develop a semiautomatic segmentation system that segments a transistor gate in an IC image accurately with minimum guidance from the user, who just has to choose the transistor gate of interest by clicking inside it once. The graph cut algorithm [1] can not be directly applied because of several issues, which we address in our work. The main issues are as follows: automatic selection of background seeds, automatic parameter selection for the energy in Eqn. (1), counteracting the bias of [1] towards smaller segment boundaries, and reducing user interaction to a single seed. In this section, we address these issues and also discuss other modifications to [1] that improve segmentation robustness.

This section is organized as follows. Sec. 3.1 describes automatic background seed selection based on orientation estimation for transistor gates. Sec. 3.2 explains our automatic parameter selection for the energy in Eqn. (1), which also

leads to eliminating the need for user guidance beyond the initial object seed. Sec. 3.3 and 3.5 explain the regional and boundary terms that we use in equation (1). Sec. 3.4 explains the compact shape prior, and Sec. 3.6 explains how we perform segmentation in piecewise manner, which improves efficiency and allows segmenting transistor gates of shapes somewhat more general than compact.

3.1 Background Seed Detection and Orientation Estimation

In [1] the user is required to mark a few object seeds and a few background seeds. Since we don't require background seeds from the user, we need to identify them automatically. For this purpose we use our prior knowledge about the width of transistor gates, which ranges from 2 to 200 pixels, and about orientation, which is roughly horizontal or vertical. We first estimate the orientation by comparing the intensity variation along the horizontal and vertical directions in a small window around the user provided seed. We choose the orientation corresponding to the smallest intensity variation. Then we take the line parallel to the dominant direction and passing through the user marked seed. All pixels at distance 200 (the maximum width) away from that line can be safely assumed to be in the background, and we mark them as background seeds.

3.2 Eliminating User Guidance and Parameter Estimation

In [1], the user has to decide on the segment quality and, if necessary, correct it by repeatedly adding new seeds and rerunning the graph-cut step. However [1] is sensitive to the parameters choice, and if they are far from optimal, significant interaction may be required for acceptable segmentation. Unfortunately, parameter estimation for [1] is not yet solved. We solve the parameter selection issue as follows. We devise a simple yet intuitive test for automatically checking the quality of a segment. Our quality test requires the average intensity difference between adjacent pixels along the segment boundary to be greater than the intensity variation inside the object. If the current segment does not pass the quality check, we readjust the parameters and rerun the graph cut step. We search over parameter space until the resulting segment passes the quality check. Thus we eliminate the need for user guidance beyond the initial object seed.

3.3 Regional Term

In this section, we explain the regional terms D_p's that we use in Eqn. (1). We first discuss D_p's for the object seed and the automatically detected background seeds. For the object seed pixel p, we set $D_p(0) = MaxInt$ and $D_p(1) = 0$, where $MaxInt$ is the maximum integer allowed. This insures that p will be assigned to the object in the optimal labeling. Similarly, for a background seed p, we set $D_p(1) = MaxInt$ and $D_p(0) = 0$. In [1] the seeds are also used to build the object and background models to be used for regional properties for other pixels. For more object data, we collect pixels from the region of size equal to the minimum possible transistor gate (2 by 10 in our application) and centered at

the object seed to build an object intensity histogram. Still the amount of data may be insufficient for an accurate intensity distribution model. Hence we use a weighted mixture of uniform distribution and a smoothed normalized histogram. For the background, we use a uniform distribution. The actual costs $D_p(S_p)$ are negative logarithms of the likelihoods. Therefore,

$$D_p(1) = -ln\left(\gamma P_{hist}(I_p) + (1-\gamma)\frac{1}{256}\right), \tag{2}$$

and $D_p(0) = -ln(1/256)$, where we assume that there are 256 gray levels in an image, and $P_{hist}(\cdot)$ is the object intensity likelihood (built from the histogram).

3.4 Compact Shape

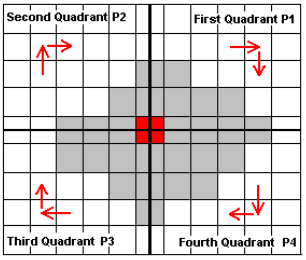

Fig. 2. Shows how segmentation is restricted in different quadrants. Object seed is red.

The transistor gates are roughly rectangular in shape, and thus if we impose a roughly rectangular shape prior on the object segment, we can significantly improve the robustness of our algorithm. We incorporate the so called *compact* shape prior. We use the word *compact* informally, borrowing the idea from [8], where they chose the word compact to reflect that for such segments, the perimeter to area ratio tends to be small. Consider *Fig.*2. It shows a square image region with the side equal to the maximum possible width of a transistor gate. The squares are the image pixels, and the dark square is the object seed. We divide this region into four slightly overlapping quadrants with respect to the seed, named P_1, P_2, P_3, and P_4. P_1 consists of all pixels above and to the right of the seed, P_2 consists of all the pixels above and to the left of the seed, P_3 consists of all the pixels below and to the left of the seed, and P_4 consists of all the pixels to the right and below the seed. We say that an object is compact if its boundary can be fully traced clockwise using only the edges in each quadrant shown in *Fig.*2. Thus in order for the object segment to be compact, we prohibit a certain set of label assignments to neighboring pixels. For example, for any neighboring pixels p and q in the first quadrant, we prohibit assigning 0 to p and 1 to q if p is either to the left or below q. Notation $p <_l q$ denotes that pixel p is to the left of q and notation $p <_a q$ means pixel p is above pixel q. If

l, l' are labels, we will denote the assignment of l to pixel p and l' to pixel q by $(p \leftarrow l, q \leftarrow l')$. Now we can define the set of prohibited assignments:

$$A^p = \begin{array}{l} \{p \leftarrow 0, q \leftarrow 1\} | p, q \in P_1 \cup P_4, p <_l q\} \cup \\ \{p \leftarrow 0, q \leftarrow 1\} | p, q \in P_2 \cup P_3, q <_l p\} \cup \\ \{p \leftarrow 0, q \leftarrow 1\} | p, q \in P_1 \cup P_2, q <_a p\} \cup \\ \{p \leftarrow 0, q \leftarrow 1\} | p, q \in P_3 \cup P_4, p <_a q\} \end{array}$$

An object segment is of *compact* shape if no prohibited assignments need to be made in its segmentation. In practice we found that incorporating the compact shape prior greatly improves the robustness of the transistor gate segmentation.

3.5 Boundary term

In this section we describe the boundary terms V_{pq} that we use in Eqn. (1). We assume that the intensity variation inside the transistor gate is smaller than the strength of intensity edges on its border. Another fact that we use is that the intensity edge between a transistor gate and its background almost always goes from light to dark. We also use the boundary term to incorporate the compact shape prior and to introduce a parameter *bias* in V_{pq} to encourage object segment towards a larger boundary. This parameter helps to counteract the well known bias of [1] towards a shorter boundaries. Hence V_{pq} is:

$$V_{pq}(S_p, S_q) = \begin{cases} 0 & \text{if } S_p = S_q \\ w_{pq} & \text{if } \{p \leftarrow S_p, q \leftarrow S_q\} \notin A^p \\ K & \text{if } \{p \leftarrow S_p, q \leftarrow S_q\} \in A^p \end{cases}, \quad (3)$$

where A^p was defined in section 3.4, the constant K is prohibitively large[4], and $w_{pq} = e^{-\frac{\Delta I^2}{2\sigma^2}} - bias$, where $\Delta I = max\{(I_p - I_q), 0\}$ encourages the intensity transition on segmentation border to be from light to dark. Parameter σ can be regarded as a measure of the noise level in the image. It affects the segmentation directly and hence a crucial parameter that needs to estimated correctly. When $\Delta I > \sigma$, the weight w_{pq} is typically small enough to allow a boundary. We compute σ as the average difference of the intensities of two adjacent pixels in a region around the object seed. The size of this region is same as the smallest possible object size which is known to us beforehand.

Parameter *bias* implements bias to a larger segmentation boundary. When the *bias* increases the boundary cost decreases. The value of *bias* has large influence on the segmentation results, and we automatically choose an appropriate value from a range by using the "quality check" as described in Sec. 3.2.

Now our energy function is fully specified, to minimize it globally and exactly with a graph cut, we just have to check that it is submodular, according to [9]. To be submodular, the binary terms of E(S) have to satisfy: $B_{pq}(0,0) + B_{pq}(1,1) \leq B_{pq}(1,0) + B_{pq}(0,1)$. The left hand-side is always 0, and the right hand-side is $w_{pq} + K$, which is always nonnegative since K is chosen to be very large.

[4] It is enough to make K equal to the cost of $E(S')$ where S' is any segmentation not containing prohibited assignments.

3.6 Piecewise Segmentation

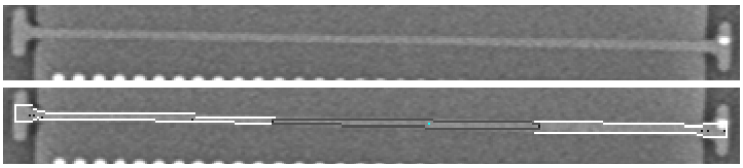

Fig. 3. Explains how the extension step works. The initial segment is outlined by black and the next segments are outlined with white.

In our application, we use piecewise approach to segmentation. First, we segment a piece of the transistor gate around the user provided seed, and if required, extend it piecewise along the dominant orientation, by repeatedly and automatically selecting a new object seed and running the graph cut until the whole transistor gate is segmented. *Fig.* 3 illustrates piecewise segmentation.

There are two main reasons for performing piecewise segmentation. The first reason is that a transistor gate may not be compact in shape, but rather consist of several roughly collinear compact pieces. The second reason is computational efficiency. There is a huge variability in transistor gate lengths, which can range from 10 pixels to a few thousand pixels. To segment the whole transistor gate, we would have to construct the graph of size equal to the biggest possible transistor gate, which would be too expensive if the actual transistor gate is medium/small. With piecewise segmentation, we run the graph cut on a series of much smaller graphs, adapting to the actual length of the transistor gate.

Our test to decide if the current segment has to be to be extended consists of measuring the dissimilarity between the intensity distribution of the current segment and that of the region just beyond the end of the segment. If the test fails, then the segment is extended. For extending, a new seed is selected which lies inside the current segment, close to the weak edge and approximately on the axis of the transistor gate in the dominant orientation.

4 Results

Fig. 4 shows segmentation results for the images in *Fig.*1. The central dots are the user entered seeds, and the other dots are the automatically chosen extension seeds. We set $\alpha = 0.007$, $\gamma = 0.4$, and they are fixed for the application. Parameters σ and *bias* are estimated adaptively for each image as already explained.

The application is able to segment transistor gates efficiently and with high accuracy. As we segment a transistor piecewise, we build relatively small 2D grid graph with 4-neighborhood connection. Hence the graph cut computations using the max flow algorithm of [7] runs very fast. The application is implemented with C++. On a P4, 2.8 GHz computer it takes less than 2 seconds to segment a transistor of size $120x2500$ pixels. Out of 100 random selections (some including

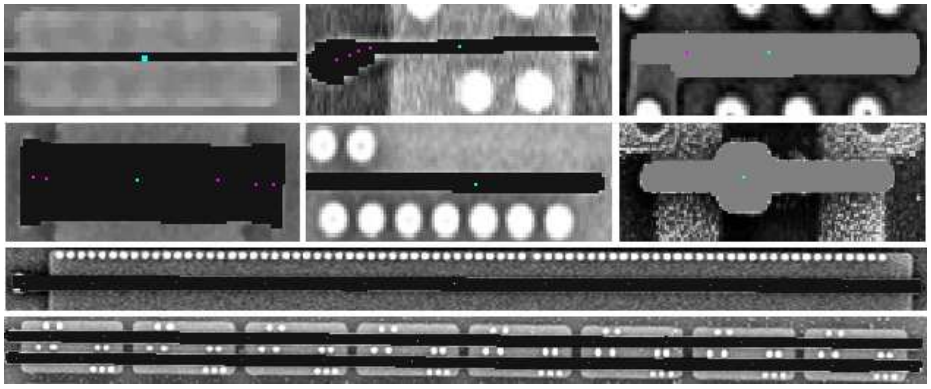

Fig. 4. Shows the segmentation results on the images displayed in 1.

very poor quality images with almost no contrast on the transistor gate border), 82 transistors were segmented accurately and 7 of them had the initial piece around the seed segmented correctly but the extension failed. If we include only the relatively better quality images, the accuracy is 95%. This application is currently being used by Semiconductor Insight Inc. for transistor segmentation replacing their manual segmentation system.

Acknowledgements

We would like thank Stephen Begg and Dale Carlson of Semiconductor Insights for developing interactive application incorporating the method.

References

1. Boykov, Y., Jolly, M.P.: Interactive graph cuts for optimal boundary and region segmentation. In: International Conference on Computer Vision(ICCV). Volume 20. (2001) 105–112
2. Gonzalez, Woods: Digital Image Processing. Second edn. Prentice Hall, Berlin Heidelberg New York (1996)
3. Kass, M., Witkin, A., Terzopoulos, D.: Snakes: Active contour models. International Journal of Computer Vision **2** (1998) 321–331
4. Cohen, L.: On active contour models and ballons. Computer Vision, Graphics and Image Processing **53** (1991) 211–218
5. Osher, S., Sethian, J.: Fronts propagating with curvature dependent speed: Algorithm based on hamilton jacobi formulations. JOurnal of Computational Physics **79** (1988) 12–49
6. Shi, J., Malik, J.: Normalized cuts and image segmentation. In: IEEE Conference on Computer Vision(ICCV). (1997) 731–737
7. Boykov, Y., Kolmogorov, V.: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. IEEE Transaction on PAMI **26** (2004)
8. Veksler, O.: Stereo correspondence with compact windows via minimum ratio cycle. IEEE Transaction on PAMI **24** (2001) 1654–1660
9. Kolmogorov, V., Zabih, R.: What energy function can be minimized via graph cuts? IEEE Transaction on PAMI **26** (2004) 147–159