CAD front-end for nanodevice design

Alexandre B. Guerra and Edval J. P. Santos

Laboratory for Devices and Nanostructures - Departamento de Eletrônica e Sistemas,
Universidade Federal de Pernambuco, Caixa Postal 7800, 50711-970, Recife-PE, Brazil

Abstract— **Algorithms for modelling quantum effects, such as the quantized conductance, are being developed by researchers worldwide. Our research group has developed a C-program for the simulation of the DC conductance of quantum devices. In this program, the device geometry has to be described in matrix format, which makes the data input process time consuming. To facilitate data input, we have developed a front-end which converts the design from a standard bitmapped format into the matrix format. This makes possible to use standard drawing software to design the device geometry, and use our program to transform the bitmapped file into the matrix the conductance simulation program understands.**

Keywords—**Nanodevice, CAD front-end, Nanotechnology**

I. Introduction

It would be too costly and a great waste of time to develop microelectronic processes and devices by trial and error. That is why is important to have mature simulation tools to predict the behavior of a device or circuit prior to fabrication. Using a process simulation tool, one can evaluate the construction of the physical structure of the device as a sequence of process steps, such as: photolithography, thin film deposition, etching, etc. Then is necessary to check on the device electrical performance. To this end, one may use a program for device simulation. If the device does not perform as desired, one can adjust its dimensions, doping levels, materials properties, etc. until the desired device performance is reached. Simulation tools can also be used to test novel device concepts.

The electrons that take part of the current flow, i.e., electrons on the conduction band, have energies near the Fermi level. As the critical dimensions of the microelectronic devices get closer and closer to the size of the Fermi wavelength, the electron present a wavelike behavior and its energy is quantized. The Fermi wavelength, λ_f, is related to the Fermi Energy through the de Broglie's relation in Equation 1.

$$\lambda_f = \frac{h}{\sqrt{2mE_f}} \tag{1}$$

where h is Planck's constant, m is the electron effective mass and E_f is the Fermi energy level.

Depending whether the device has one, two or three dimensions of the order of the Fermi wavelength, the device is classified as $2-D$, $1-D$ and $0-D$. A nanowire is a one-dimensional device and its electrical conductance displays a quantized behavior. This class of devices require a full quantum description for the simulation of its behavior, unlike traditional devices which can be simulated under the semi-classical approach. For quantum devices, as a first approximation, one can use the single electron approach, i.e., neglect many body effects, and solve Schrödinger equation to calculate the electrical properties.

From the solution of the wave equation, one can calculate the quantum transmission probability $T(E)$ and then apply the Landauer (1,2) relation given by Equation 2.

$$G = g_s G_Q \sum |T(E)| \tag{2}$$

where G_Q is the quantum conductance and g_s=2 is the spin degeneracy.

As the voltage applied at the terminals of the nanowire is increased, more open channels become available, increasing the conductance. In a quantum device, the channels are discrete energy levels, so a step-like behavior is expected, which has been confirmed experimentally.

In Figure 1, a few quantum devices are presented. In the figure one can see a quantum wire, an electronic Mach-Zender interferometer, an Aharonov-Bohm ring and a quantum constriction.

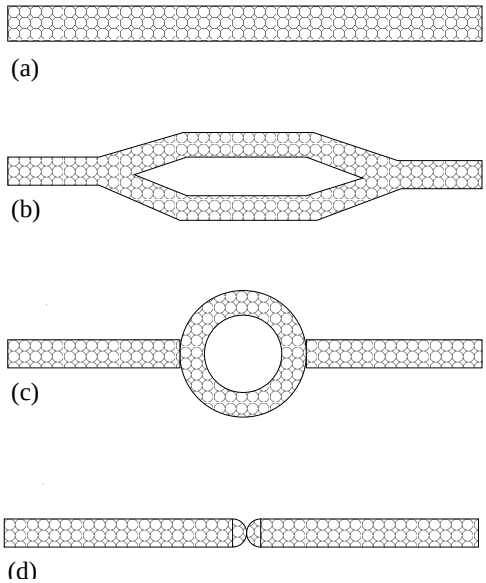

Fig. 1. Sample devices: (a) quantum wire; (b) Mach-Zender interferometer; (c) Aharonov-Bohm ring; (d) quantum constriction.

For traditional microelectronic devices, there are many simulation tools available commercially.

These codes cannot simulate quantum devices correctly, as in such codes it is assumed that during collision events, the electrons behave very much like billiard balls (3,4). Hence new simulation tools are required. Algorithms for modelling quantum effects, such as the quantized conductance, are being proposed by researchers worldwide (5,6,7,8,9). Our research group has developed a C-program for the simulation of the DC conductance of quantum devices (9).

Our simulation program solves the wave equation by calculating the associated Green function. The device is divided in slices as shown in Figure 2. Each slice is stored as a column of the device matrix, where each matrix entry is related to the potential energy confining the electron. Each column (slice) of the device matrix is used to construct the Hamiltonian matrix for numerical calculation by using the tight-biding Hamiltonian Model. The n_{th} slice matrix is shown in Equation 3, $V_1..V_M$ in the diagonal is one slice of the device and one column of the device matrix. This makes the data input process time consuming. To facilitate data input, we have developed a front-end that converts the design from a standard bitmapped format into the matrix format that the program understands.

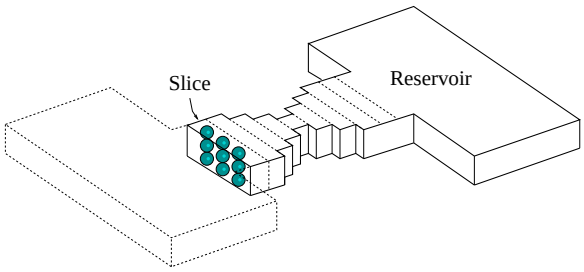

Fig. 2. Quantum device divided in slices for the simulation of quantum transport. The device is connected to two reservoirs.

$$H_n = \begin{pmatrix} V_1 - 4t & -t & 0 & \dots & 0 \\ -t & V_2 - 4t & -t & \dots & 0 \\ 0 & -t & V_3 - 4t & \dots & 0 \\ . & . & . & & . \\ . & . & . & & . \\ . & . & . & & . \\ 0 & 0 & 0 & \dots & V_M - 4t \end{pmatrix} \quad (3)$$

where V_i in the diagonal is the local potential energy profile at a given slice, and t is a coupling term.

II. The front-end

Our idea is to use a drawing program or structured design program with bitmapped format export facility. One such example, is the publicly available Gimp (10) under Linux.

As was mentioned in the introduction the device matrix is actually the potential energy which confines the electron inside the nanodevice under study. An infinite potential means that the electron has zero probability of being found in that region. For the sake of this paper, we assume that the confining potential may only have two values, $V = 1$ and $V = 0$. This is represented in the bitmapped file as two colors, black and white. The designer draws the structure at a graphical application and the front-end returns the matrix that is used by the computation algorithm (9). The bitmapped file format is selected because there is a direct relationship between the pixel and the data stored in the file. The bitmap file structure (11) is shown in Figure 3.

```
BITMAPFILEHEADER bmfh;(Bitmap-file header)
BITMAPINFOHEADER bmih;(Bitmap-information header)
RGBQUAD          aColors[];(Bitmap color table)
BYTE             aBitmapBits[];(Array of bytes)
```

Fig. 3. Description of the bitmapped file header.

The bitmap-file header, BITMAPFILEHEADER, contains information about the type, size, and layout of a device-independent bitmap file. The bitmap-information header, defined as a BITMAPINFOHEADER structure, specifies the dimensions, compression type, and color format for the bitmap. The color table, defined as an array of RGBQUAD structures, contains as many elements as there are colors in the bitmap.The bitmap bits, immediately following the color table, consist of an array of BYTE values representing consecutive rows, or scan lines, of the bitmap. Each scan line consists of consecutive bytes representing the pixels in the scan line, in left-to-right order. The scan lines in the bitmap are stored from bottom up. This means that the first byte in the array represents the pixels in the lower-left corner of the bitmap and the last byte represents the pixels in the upper-right corner.

In the monochromatic bitmapped file format, the color table contains only two entries. Each bit in the bitmap array, the BYTE structure, represents a pixel. If the bit is clear, the pixel is displayed with the color of the first entry in the color table. If the bit is set, the pixel has the color of the second entry in the table.

The front-end algorithm begins by discarding the first 18 bytes of the file and reading the next four. These bytes are added into a single variable in order to obtain the image width. The process is repeated in the next four bytes in order to obtain the image height. With the image width, the algorithm computes the length of the image scan lines. The bitmap scan lines are stored in the disk as multiple of 4 bytes. So the algorithm rounds the image width divided by 32 (number of bits of 4 bytes) to the nearest multiple of 4 bytes forward plus infinity. With the length of the scan lines the algorithm goes to the 62nd byte of the file and begins to read the image data. To address each bit individually a byte mask is implemented. The bit information is stored in an auxiliary matrix. Then the auxiliary matrix is mapped into the final matrix. The final matrix discards the elements of the scan lines greater than the width of the image, to compensate the possible extra bits added at the image stored in the file bitmap format. The algorithm steps are summarized below.

1. Read the image width and height
2. Compute the scan line length
3. Read the image data
4. Build the auxiliary matrix
5. Build the final matrix
6. Call the computation algorithm

Fig. 4. An Aharonov-Bohm ring bitmap figure used as an input to the Front-end program .

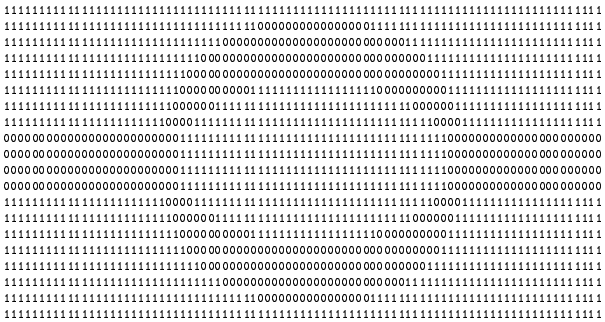

Fig. 5. The Aharonov-Bohm ring after being processed by the developed front-end.

As an example, consider the Aharonov-Bohm ring in Figure 4, as the image input. After running the front-end program, we have the final matrix shown in Figure 5.

III. Conclusions

With our front-end program the time consuming process of building the Hamiltonian matrixes was eliminated and this made computation more user friendly.

Acknowledgements

The authors would like to thank the *Instituto do Milênio* program of CNPq, Brazilian agency. One of the authors (ABG) would like to thank the scholarship funded by CAPES, Brazilian Agency.

References

[1] Rolf Landauer, *IBM J. Res. Develop.*, **1**, 223 (1957); *Phil. Mag.*, **21**, 863 (1970).

[2] H. Baranger and A. Douglas Stone, *Phys. Rev. B*, **40**, 8169 (1989).

[3] David K. Ferry and Robert O. Grondin, *Physics of submicron devices*, Plenum Press, New York (1991)

[4] Richard L. Liboff, *Kinetic theory: classical, quantum, and relativistic descriptions*, p. 130, Prentice-Hall Advanced Reference Series, New Jersey (1990).

[5] Felix A. Buot, *Physics Reports*, **234**, 73-174 (1993).

[6] T. Ando, *Phys. Rev. B*, **44**, 8017-8027 (1991).

[7] K. Nikolić and A. MacKinnon, *Phys. Rev. B*, **50**, 11008-11017 (1994).

[8] A. MacKinnon, *Z. Phys. B*, **59**, 385 (1985).

[9] Edval J. P. Santos, Simulation of the dc conductance of ballistic quantum devices, to appear in the Proceedings of the XVII SBMicro (2002)

[10] http://www.gimp.org

[11] James Murray, Encyclopedia of Graphics File Formats, 572-591 (1994)