

XML Based File Format for QCADesigner

Timothy J. Dysart and Peter M. Kogge

Department of Computer Science and Engineering

University of Notre Dame

Notre Dame, IN 46556, USA

{tdysart, kogge}@cse.nd.edu

August 25, 2004

Abstract

One major aspect of all modern computer aided design (CAD) systems/packages is that they use a standard file format for communicating between each program. Important aspects of previous formats include the separation of technology and architecture along with simple interfaces for hierarchical design. Currently, the major quantum-dot cellular automata (QCA) CAD tool, QCADesigner, does not have a file format that is capable of handling these aspects. This work removes this restriction by creating an XML based template for technology files, architecture files, and cell library files that allow the architecture and technology to be separated and thus provide a simple, practical method of handling hierarchical designs. Additionally, QCADesigner has been modified to use these various file types.

1 Introduction

Software packages that design, layout, and aid in the fabrication of integrated circuits have simplified the design process and reduced the time to market for many circuits. Additionally, with an eye towards Moore's law, these packages developed mechanisms and file formats that allowed a circuit to be designed once and fabricated through a variety of processes and techniques. One of the first file formats for these packages was the Caltech Interchange Format or CIF [1]. This format used a metric called "lambda" that could vary in size due to the fabrication technology being used, and during the design phase, circuit components were separated by a specific number of units of lambda. In other words, CIF provided a separation between the architecture of the circuit and the technology used to fabricate it.

This philosophy is one of the main reasons for the creation of the quantum dot cellular automata (QCA) system file formats described in this paper that are based on the XML (eXtensible Markup Language) standard [2]. For a description of QCA basics, the reader is directed to [3]. To separate the architecture from the technology, two different file types are created. The first is an architecture file which contains the details of how the cells are connected, whereas the second is a technology file that describes features like the distance between cell centers and where the dots are located.

Another major goal of this work is to simplify hierarchical design capabilities and reduce the space needed for saving large designs. Figure 1 shows the definition of hierarchical design being used for this work in that a circuit being designed at a level can use any circuits from any lower level. For example, the CPU in level 3 can use any of the components in level 2, the gates in level 1, or the circuits in level 0. Similarly, the components in level 2 can use the gates in level 1 and the circuits in level 0. Including circuits from the same level can be done, but is risky as a loop can be created (i.e. the NAND Gate using the NOR gate and the NOR Gate using the NAND gate) which will be infinite and the design will not work. The hierarchical design capability is done by introducing a third file type for cell libraries which takes advantage of a standard for including XML based files within another XML based file. For the purposes of this report, the file types will be identified by their extensions which the architecture file extension being qca (note that this is small, not capital type) files, technology files will have the tqc extension, and cell library files have the qcl extension. Similarly, the collection of these file types will be named the XFFF (XML File Format Framework).

However, the development of a file format is pointless without a tool or set of tools that are designed to use it. Currently, QCADesigner is the only freely distributed tool for designing QCA circuits [4, 5]. Some of the features in QCADesigner include many of the basic cell layout functions (new cells, copy, delete, move, and so on), import/export of cell blocks, and various simula-

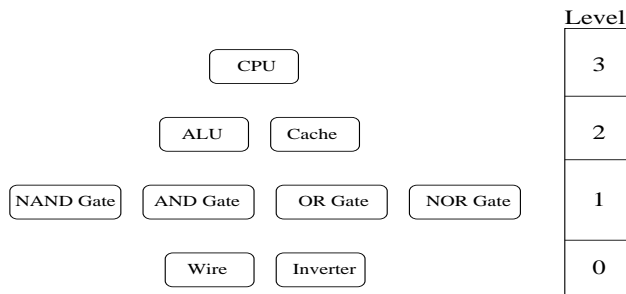


Figure 1: Design Hierarchy. At each level, the circuit can use components from any lower level.

tion methods. This report assumes that the reader is familiar with the basic operation and use of QCADesigner. This work modifies QCADesigner version 1.4.0 to use the XFFF and XFFF should be the standard in the next large revision of QCADesigner which will be 2.0. Source code for QCADesigner version 1.4.0 using the XFFF and sample files, similar to those presented here, are available online (www.nd.edu/~tdysart/xml).

The rest of this report is outlined as follows. Section 2 describes the initial QCADesigner file format, while Sec. 3 describes the basics of the XML standard that are necessary to understand this work. Sections 4 and 5 cover the current architecture and technology files. The tools and libraries needed are in Sec. 6. Sec. 7 describes the integration between QCADesigner and the XFFF. Some ideas for future work and the conclusion are in Sec. 8

2 Initial QCADesigner File Format

Although the current file format for QCADesigner is adequate for small systems, it does have several limitations. These include the large file size, difficulties with future growth and standardization, no hierarchical capabilities, and no independence between the architecture and the technology. To demonstrate the large file size, refer to the example in Appendix A which is a project file containing just a single cell. This cell description is just under 1 Kbyte in size, and will grow close to linearly as more cells are added to the design. The reasoning for this is that although QCADesigner is fully capable of importing and exporting blocks of cells, its current file format is incapable of seeing these blocks as anything other than individual cells. Thus, each cell has to be saved individually rather than having a pointer in the file to the imported block. A more detailed discussion of file sizes and hierarchical design can be found in Sec. 4.2.

A bit of this single cell file can be found in Table 1 and it is easy to see how the architecture

1	[<i>QCELL</i>]
2	$x = 1.400000e + 02$
3	$y = 1.600000e + 02$
4	$top_x = 1.310000e + 02$
5	$top_y = 1.510000e + 02$
6	$bot_x = 1.490000e + 02$
7	$bot_y = 1.690000e + 02$
8	$cell_width = 1.800000e + 01$
9	$cell_height = 1.800000e + 01$
10	$orientation = 0$
11	$color = 2$
12	$clock = 0$
13	$is_input = 1$
14	$is_output = 0$
15	$is_fixed = 0$
16	$label = a$
17	$number_of_dots = 4$
18	[<i>QDOT</i>]
19	$x = 1.445000e + 02$
20	$y = 1.555000e + 02$
21	$diameter = 5.000000e + 00$
22	$charge = 8.000000e - 20$
23	$spin = 0.000000e + 00$
24	$potential = 0.000000e + 00$
25	[<i>#QDOT</i>]

Table 1: Part of the original QCADesigner file format.

and technology are not separated. One way is by noting that each cell has a cell width and cell height that is stored (lines 8 and 9), and these values are obviously technology dependent. Additionally, note that the cell contains values regarding the number of basic quantum dots (lines 17) it contains and then a substantial amount of information regarding each dot (lines 19-24). The newly developed qca file type does not contain this same data in such close combination in order to separate the technology from the architecture. Obviously, information such as clock zone and label are retained from the initial format for use in qca files since these values are not technology dependent.

The lack of a standard (or template) will limit the overall growth of QCA design tools. Since the current format is not based on a standard it may be difficult to adapt it to future design tools and may inhibit growth as QCADesigner is more fully developed. This is not to say that the current format

is pointless, but rather to point out its limitations as QCA CAD tools grow and evolve. The reason behind using a standard is that without one, each new design tool may develop its own file format, thus slowing the overall development process. Also, as the number of file formats increases, so will the number of converters needed. By creating a standard, the number of converters needed should be minimized and more time can be spent on developing better tools. By creating a file format based on a common standard like XML, tool development time can be reduced since libraries are available to simplify the development of file input and output code (which tends to be challenging to develop). Additionally, by using a standard the file input and output code can be reused among many projects, thus saving even more time.

3 XML Basics

There are a large number of resources available for learning about XML and several were used on this project including [6–9]. This section will define some of the basic terms used within the rest of this paper to provide the reader with some familiarity regarding the XML standard and how it works.

It is important to keep in mind that XML is not a file format or data structure in and of itself. Instead, it is a standard that provides a framework which a developer can use to develop a file format or, if the format is already provided (i.e. XHTML), a structured mechanism for creating files. From a different perspective, consider XML to be the same as the Ethernet standard. With Ethernet, the communications between computers is the same, but different manufacturers can produce different Ethernet cards. In other words, a file format like XHTML is to XML what an Ethernet card from company Y is to Ethernet.

Note to reader: *Italicized* text is used in the remainder of this report to denote parts of XML files within the text.

3.1 Elements and Attributes

Before creating a file or a file template the basic building blocks of XML should be understood first. To aid in this understanding, part of an architecture (or cell library) file is shown in Table 2. This can be a part of either file since the below subsection is a part of how QCA cells are saved. A more detailed discussion on what this part does is in Sec. 4.1. The basic building block is an element which contains a desired piece of information. For example, line 4 of the of this sample contains the element named *clock* and the value of that element is 0. One of the key things to note about an element is that it has a tag pair associated with the opening tag being `<clock>` and

1	<Initializers>
2	<orientation>0</orientation>
3	<color>2</color>
4	<clock>0</clock>
5	<iof polarize = "0">x</iof>
6	</Initializers>

Table 2: Part of an architecture/cell library file.

</clock> as the closing tag. Another useful aspect of an element is that it can have any number of elements embedded within it, such as the element *Initializers* does. In this case, the elements *orientation*, *color*, *clock* and *iof* are embedded within the *Initializers* tags. An element, like *iof* may also have attributes, which are fields within the opening tag such as *polarize* which has a value of 0. These attributes typically carry values of interest to that specific element. Of course, many attributes can be a sub-element of the element, but attributes tend to contain data that is considered to be important to a computer rather than a human.

3.2 Document Type Definitions

Once the specific data elements for the file are known, a file template, also known as a Document Type Definition (DTD), can be created to formalize the file structure. All three file types developed in this work have a unique DTD that defines how the files are to be structured. To draw an analogy, the technology DTD defines the tqc file format, which is similar to how Adobe has created a PDF definition that defines the pdf file format. All valid technology files conform to the rules set down in the technology DTD. The full DTD for technology files can be found in Appendix C, but a small section is show below in Table 3 to use as an example. Section 5 will demonstrate how a technology file uses the DTD.

The first line of the DTD can be ignored since it is fixed for all of the XML files used in this work. It should also be noted that a line starting with <! -- and ending with -- > denotes a comment line in XML files. Elements are declared by starting a line (part within the < and > symbols) with *!ELEMENT* as is done in lines 2, 5, 6, and 7. As can be seen in line 7, an element is declared in the first part of the line and the second part of this line is the element's name, in this case *width*. The third part of the line, between the parentheses, lists what the element *width* contains. For *width*, this is *#PCDATA*, which is parsed character data. Line 6 is the same, but defines *height* instead of *width*. Line 5, *CellInfo*, rather than containing *#PCDATA*, contains two subelements, *width* and *height*. Additionally, since *width* and *height* are each listed once without

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!ELEMENT TechFile (CellInfo, DotInfo, Other)>
3
4 <!-- Start Cell Info -->
5 <!ELEMENT CellInfo (height, width)>
6 <!ELEMENT height (#PCDATA)>
7 <!ELEMENT width (#PCDATA)>
8 <!-- Non-real attribute used as an example -->
9 <!ATTLIST width units CDATA #REQUIRED>

```

Table 3: Part of the Document Type Definition (DTD) for a technology file.

any other punctuation, each element is required to appear once and only once between *CellInfo* tags in the technology file. If *width* had a “+” after it, *width* would appear one or more times between the *CellInfo* tags in a technology file. Similarly, a “*” would allow for zero or more occurrences and a “?” would dictate zero or one occurrence. Lastly, on line 2, the element *TechFile* is defined as having one instance of each of the following elements: *CellInfo*, *DotInfo*, and *Other*. Definitions for *DotInfo* and *Other* are required, but are not shown since they are similar to that of *CellInfo*. Although it is not actually in the technology DTD, line 9 of the below example shows how an attribute named *units* for the element *width* is declared. The line says that an attribute is being declared (*!ATTLIST* for the *width* element and that the attribute’s name is *units*. The *CDATA* part of the line states that the attribute is character data and the *#REQUIRED* says that when the *width* element is seen, it must have an attribute named *units*. As was seen earlier, the opening tag for *width* would then be *<width units=0>*.

Now that the DTD for the technology file has been examined, a technology file using the DTD can be examined. Table 4 shows a valid XML file that is also well-formed. These terms will be explained in a bit more depth briefly. Lines 7 and 8 show the elements *height* and *width* and values (character data to be parsed) between the tags. Similarly, only single copies of *CellInfo*, *DotInfo*, and *Other* are shown. On line 5 is the opening tag for *TechFile* while the closing tag is on line 16. The important thing to note in this file is the information on lines 2 and 3. The first two items on line 2, *!DOCTYPE TechFile* inform the parser that a document of the type *TechFile* is about to be processed. It is a requirement that this document type and the root element are the same. The second part of lines 2 and 3, *PUBLIC* “ “ *“http://www.nd.edu/~tdysart/xml/tech.dtd”* tells the parser if the DTD is local or remote. *PUBLIC* refers to a remote DTD, while replacing it with *SYSTEM* refers to a local DTD. For a *PUBLIC* DTD, the first pair of quotation marks can remain empty while the second pair contains the URL where the DTD can be found. If this file

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE TechFile PUBLIC ""
3     "http://www.nd.edu/~tdysart/xml/tech.dtd">
4
5  <TechFile>
6     <CellInfo>
7       <height>3</height>
8       <width>3</width>
9     </CellInfo>
10    <DotInfo>
11      <!-- Details Omitted -->
12    </DotInfo>
13    <Other>
14      <!-- Details Omitted -->
15    </Other>
16  </TechFile>

```

Table 4: A valid technology file.

was using a local DTD, then only one pair of quotation marks is necessary and the path to the DTD is contained within them.

3.3 Well-Formedness and Validity

One key distinction within XML is the idea of a valid or a well-formed file. A file can be well-formed and not valid, but it cannot be valid and not well-formed. In this example, the technology file is valid since it is both well-formed and has a DTD that it can be checked against. If it did not have a DTD to be checked against, it would only be well-formed. A file can be checked for well-formedness by pushing and popping tags off of a stack. Each time an opening tag is seen, it is pushed onto the stack. Each time a closing tag is found, the tag on the top of the stack is popped off and compared to the closing tag. Provided that all tags match (and the stack is empty as a result), the document is well-formed, else it is not. Determining if a file is valid or not can be done by comparing the file to the rules enforced by the DTD, but it is easier to use a tool such as xmllint [9].

```

1  <!ELEMENT QCELL (Location, Initializers, Info)>
2  <!-- Start Location info. -->
3  <!ELEMENT Location (x_center, y_center)>
4  <!ELEMENT x_center (#PCDATA)>
5  <!ELEMENT y_center (#PCDATA)>
6  <!-- Start Initializers -->
7  <!-- iof:  input, output, fixed, x = don't care -->
8  <!ELEMENT Initializers (orientation, color, clock, iof)>
9  <!ELEMENT orientation (#PCDATA)>
10 <!ELEMENT color (#PCDATA)>
11 <!ELEMENT clock (#PCDATA)>
12 <!ELEMENT iof (#PCDATA)>
13 <!ATTLIST iof polarize CDATA #REQUIRED>
14 <!-- Start Info Section -->
15 <!ELEMENT Info (label)>
16 <!ELEMENT label (#PCDATA)>

```

Table 5: Part of qca/qcl DTD's used for describing individual QCA cells.

4 Architecture and Cell Library Files

This section will outline the architecture (qca) and cell library (qcl) file formats for QCADesigner. These two formats are quite similar, and since the qcl format is extended for the qca format, it will be discussed first. One of the goals of this work was to separate the technology used to build the circuit from the structure of the circuit and it is important to note that these two formats are successful at doing this. Secondly, a mechanism for creating a hierarchical design is desired. This mechanism has been created and relies on the XInclude standard.

Note to reader: Throughout the remainder of this report, snippets from both DTDs and the three file types exist within the text. This has been done to show different points based on the context, and for completeness, the files that these snippets are taken from are enclosed in the appendices. It is assumed that the conversion from DTD to file based on the DTD and vice-versa is understood.

4.1 QCA Cell Representation

Prior to discussing the internals of either file format presented in this section, the representation of individual QCA cells will be described here. Below in Table 5 is the part of the qca and qcl DTDs that describe how individual cells are represented.

The *QCELL* element refers to the basic unit of QCA: the individual cell. The three subelements within the *QCELL* are the *Location*, *Initializers*, and *Info*. The *Location* element contains two elements that hold the x location and y location of the cell center, so the cell can be placed in the proper location based on the arbitrary scale used in QCADesigner. The *Initializers* include the cell *orientation* which refers to whether it is a 45 degree or 90 degree cell, the *color* the cell is when displayed by QCADesigner, the *clock* signal the cell is attached to, and *iof* which stands for input, output, and fixed. *iof* has an attribute named *polarization* which is used to hold the cell polarization if the cell is fixed. The last element of *QCELL* is *Info*. Currently, *Info* has only one element, *label* which is the cell's name if it has been given one.

4.2 Cell Libraries

A major contribution of this work is the creation of hierarchical design capabilities for QCA design tools, namely QCADesigner. The ability to create a hierarchical design is a requirement of any good CAD tool, and until now, this support was lacking in QCA design tools. This capability uses a standard for XML inclusions known as XInclude [10]. With XInclude, XML or text files can be added into another XML file through the use of a simple link. By linking several files together as was shown in Fig. 1, a hierarchical design is created.

The cell library file format, with root element *IncludedCells*, is designed to hold a quantity of *QCELLS* and other cell library files that are linked in using the XInclude standard. The DTD for qcl files can be found in Appendix F and two sample qcl files are also in the appendices with an inverter in Appendix G which has nothing but *QCELLS* in it and a NAND gate in Appendix H that has both *QCELLS* and links to include the inverter (*CELL_LIBS* element). Table 6 is a sample qcl file with a link to another qcl file and a single *QCELL*. Although there is only one *QCELL* and one *CELL_LIBS* (qcl file) linked in, a qcl file can have as many (including zero) of each of these elements in it as is required by the design.

The *CELL_LIBS* element has three attributes with it. The *begin_x* and *begin_y* attributes state where the first cell in the linked in file (Inverter.xml in this case) is to be located. The other attribute, *rotate* is available for rotating the included cells around the unit circle (i.e. if *rotate* were 90, a cell initially at 0,50 would now be at 50,0). The lone element in *CELL_LIBS* is *xi:include* which has two attributes: *xmlns:xi* and *xi:href*. The value of *xmlns:xi* is fixed due to the XInclude standard while the value of *xi:href* points to the location of the file that is being included, ./Inverter.xml here. This location can be a local path or a URL (i.e. <http://www.nd.edu/~tdysart/xml/Inverter.xml> is valid provided that Inverter.xml exists at that location). One limitation of this linking mechanism is that the first cell or included file must have a starting location of 0,0. For this case, that means

```

1 <IncludedCells>
2 <CELL_LIBS begin_x="0" begin_y="0" rotate="0">
3 <xi:include xmlns:xi="http://www.w3.org/2001/XInclude"
4 xi:href = "./Inverter.xml" />
5 </CELL_LIBS>
6 <QCELL>
7 <!-- Details Omitted -->
8 </QCELL>
9 </IncludedCells>

```

Table 6: A sample cell library file.

```

1 <!ELEMENT QCADesignerProjectFile (INIT, (CELL_LIBS | QCELL)+ )>
2 <!ELEMENT INIT (QCAD_Version, Design_Opts)>
3 <!ELEMENT QCAD_Version (#PCDATA)>
4 <!ELEMENT Design_Opts (grid_spacing)>
5 <!ELEMENT grid_spacing (#PCDATA)>

```

Table 7: Part of an architecture Document Type Definition (DTD).

both the *begin_x* and *begin_y* values must be 0, and the location of the first cell in *Inverter.xml* must also be at 0,0. This limitation is caused by how QCADesigner handles these cell libraries. If the starting location was not 0,0 the block of cells in the library would move by the difference from 0,0 each time the design is opened, saved, and reopened.

4.3 Architecture Files

The DTD for the project file can be found in Appendix D. With the exception of the few lines in Table 7, the architecture file DTD is nearly the same as for the cell library DTD. The root element for an architecture file is *QCADesignerProjectFile*. The new element as compared to the cell library format is *INIT*. The elements of *INIT* are *QCAD_Version* and *Design_Opts* with *Design_Opts* having a single element underneath it named *grid_spacing*. *QCAD_Version* will tell QCADesigner which version of QCADesigner created the file while *grid_spacing* refers to the level of zoom that is to be used when opening the file.

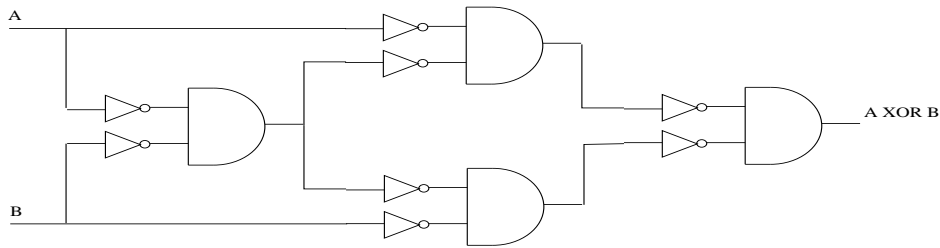


Figure 2: Exclusive OR gate using 4 NAND gates.

4.4 File Size

Being able to build small blocks and adding them into larger blocks is an obvious advantage of hierarchical design, but the other is that the reduction in space needed to store the design can be substantial. A simple example of this can be seen when implementing an XOR gate with 4 NAND gates as is shown in Fig. 2. As Fig. 3 shows, the circuit contains 158 cells, and the QCADesigner text based format requires just over 111 Kbyte of space for the file. Using the new qcl format, an inverter is created first which requires 9 cells and just under 3 Kbyte of space. Next, a NAND gate is created (inputs inverted) which has 10 cells and two inverters and takes up about 4 Kbyte of space when saved as a qcl file. Lastly, the XOR gate is created with four NAND gates and 46 other cells for wiring. This file requires about 16 Kbyte of space which could be reduced further by creating new qcl files that are wires and using them instead of individual cells. Thus, once the 1 Kbyte for the technology file is added in, the total space needed using the XML format is roughly 24 Kbyte or about 20% of what the old format required. As QCA system designs become larger, this size difference will increase due to the removal of technology specific information from the architecture files and if the advantage of cell library files is exploited.

5 Technology File

To handle the separation of architecture and technology, a technology file must be created which contains technology specific information. The basic technology file DTD is in Table 8 and the full one (with comments) is in Appendix C and a tqc file can be found in Appendix B. As is seen, the relevant pieces of information include cell *height* and *width*, *dot diameter*, and the cell center to cell center distance which is called *lambda*. Two other elements in the DTD are the *angle*, which is defined in Fig. 4, and the cell center to dot center distance called *cd_dist*. The one assumed piece of technology information is that the cell has four dots. Currently, this seems to be the relevant

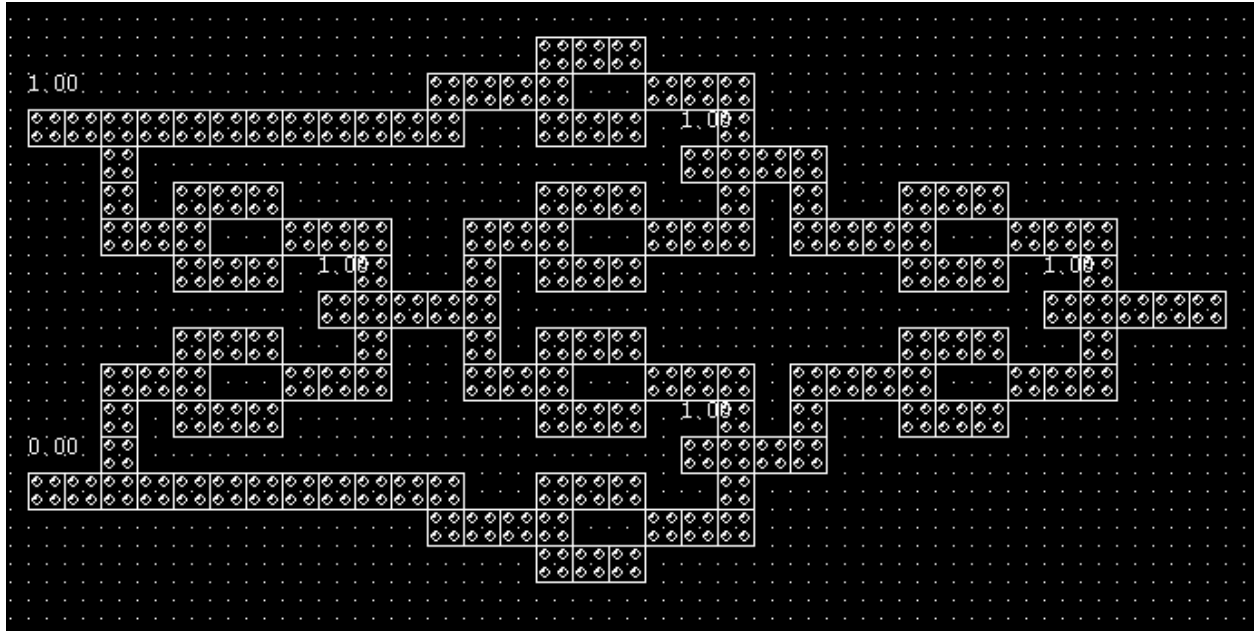


Figure 3: Exclusive OR gate using 4 NAND gates with QCA cells. The inputs (fixed) are on the left and the output is on the right. The AND gates are created by fixing an input to -1. Also, this picture does not show the relevant clocking zones.

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <!ELEMENT TechFile (CellInfo, DotInfo, Other)>
3
4  <!ELEMENT CellInfo (height, width)>
5  <!ELEMENT height (#PCDATA)>
6  <!ELEMENT width (#PCDATA)>
7
8  <!ELEMENT DotInfo (diameter)>
9  <!ELEMENT diameter (#PCDATA)>
10
11 <!ELEMENT Other (cd_dist, angle, lambda)>
12 <!ELEMENT cd_dist (#PCDATA)>
13 <!ELEMENT angle (#PCDATA)>
14 <!ELEMENT lambda (#PCDATA)>

```

Table 8: The full Document Type Definition (DTD) for technology files.

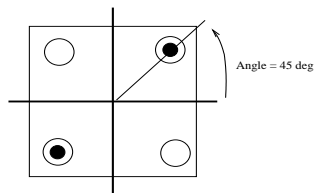


Figure 4: The “angle” in the technology file.

information for describing the technology of QCA cells, but if new or different information is needed it can be added with ease.

6 Tools

During the course of this project, only a few tools were needed to develop the three file types (architecture, technology, and cell library). The first tool used while developing the file types and DTDs is xmllint. This program can be used to check for the well-formedness and validity of XML files as was described in Sec. 3.3 and is extremely useful when using the XInclude standard. With xmllint, file input and output code could be tested prior to its integration with QCADesigner which greatly simplified the debugging process. File input refers to the process of parsing the three file types and converting the data into the proper form for QCADesigner. File output refers to the

processes of creating a qca or qcl file for the design based on whether the QCADesigner user asks for an architecture file or cell library file to be created.

The file input and output code for QCADesigner uses the libxml2 library [9]. Libxml2 is a C library capable of many different XML actions, but its use here is limited to file input and output. For those interested in how the file I/O is accomplished with this library, please refer to the source code of QCADesigner which is available at www.qcadesigner.ca or, if the XFFF is not included in the most recent version there, refer to www.nd.edu/~tdysart/xml/.

7 Integration with QCADesigner

The key aspect of this work is the integration of the XFFF with QCADesigner. This has been accomplished by modifying the internal data structures and file input/output functions of QCADesigner. The file input/output functions have been modified to use the qca, qcl, and tqc file types as opposed to the original file format used in QCADesigner. The internal data structures have been adjusted to use a cell that is “fixed” in terms of size for visualization purposes, but a technology file is required as input prior to any physical simulations being run. The digital simulation does not depend on the technology being used, but since it uses the old internal data structures, a technology file still must be specified before running this simulation.

One of the main features of these new file formats in QCADesigner is that when a block, or cell library, is imported into the design, that block is always treated as a single unit. Thus, one cannot move a cell from within the block without moving the whole block. Additionally, when the design is saved, the imported block will then be linked in to the top level design file as a cell library as opposed to individual cells which will reduce the time it takes to write the file and the space required for the qca file.

Another feature that has been implemented is that a design can be created and saved as a cell library (qcl file) as opposed to an architecture file. This enables a user to create a small block (i.e. inverter) and save it as a qcl file. Then the inverter can be imported and used in a NAND gate which can then also be saved as a qcl file. Lastly, a new design can be created which uses the NAND gate (imported), and can be saved as a cell library, or, if it is a final design, an architecture file. Thus, one of the unique features of the XFFF is that one can have a large (possibly infinite) number of cell library files that can be created and used provided a loop is not created. A loop in the above example would be using the NAND gate in the inverter which would cause an inverter \rightarrow NAND \rightarrow inverter loop which is non-sensical and cannot be parsed.

Lastly, a simple file converter (qca2xml) has been created which will convert files saved in the

old QCADesigner format into files using the XFFF. Given an old QCADesigner file as input an architecture or cell library file is output. The source code and other documentation for this program can be found at www.nd.edu/~tdysart/xml/.

8 Future Directions and Conclusion

One downside to the cell libraries is that they must be fixed in terms of size and number of cells. For example, one cannot have a wire in their cell library that has a varying number of cells. Thus, if one wants a five cell long wire, one has to create a five cell long wire in QCADesigner and save it as a cell library as opposed to having a generic wire qcl file that can be made any number of cells long. Removing this restriction is desired, but does not appear likely in the near future. As was discussed in Sec. 4.2 another small restriction is that the first cell of a cell library file must be at location 0,0. Without this restriction, the library will move as the architecture file using this cell library is opened and resaved. This restriction may be removed in the future as well. Other tasks to be undertaken include not having to specify a technology file before running a digital simulation and devising a method to avoid hand-writing technology files.

This work has taken the original QCADesigner file format and divided it into three different file types. The first, a technology file, contains the information needed to take a circuit design and scale it to the proper manufacturing method. The second, an architecture file, is the high level circuit design information while the third, an cell library file, enables the system to be designed in a hierarchical manner. These three types have separated the technology from the architecture and have enabled a significant reduction in file size and an increase in flexibility. Additionally, this work can be used as a guideline as to how other nano-electronic devices can develop a file format(s) to use from the start while developing CAD tools.

The reader is directed to www.qcadesigner.ca for the most recent version of QCADesigner and www.nd.edu/~tdysart/xml/ for the XFFF based version of QCADesigner along with all of the necessary DTDs and sample files of each type.

References

- [1] C. Mead and L. Conway, *Introduction to VLSI Systems*. Reading, MA: Addison-Wesley.
- [2] Extensible Markup Language (XML). [Http://www.w3.org/XML/](http://www.w3.org/XML/). [Online]. Available: <http://www.w3.org/XML/>

- [3] C. S. Lent and P. D. Tougaw, "A device architecture for computing with quantum dots," *Proceedings of the IEEE*, vol. 85, no. 4, pp. 541–557, Apr. 1997.
- [4] QCADesigner home page. [Http://www.qcadesigner.ca](http://www.qcadesigner.ca). [Online]. Available: <http://www.qcadesigner.ca>
- [5] K. Walus, T. J. Dysart, G. A. Jullien, and R. A. Budiman, "QCADesigner: A Rapid Design and Simulation Tool for Quantum-Dot Cellular Automata," *IEEE Trans. on Nanotechnology*, vol. 3, no. 1, pp. 26–31, Mar 2004.
- [6] L. M. Garshol, *Definitive XML Application Development*. Upper Saddle River, NJ: Prentice Hall, 2002.
- [7] S. S. Laurent, *XML: A Primer*. Foster City, CA: MIS:Press, 1998.
- [8] H. M. Dietel, P. J. Dietel, T. R. Nieto, T. M. Lin, and P. Sadhu, *XML: How to Program*. Upper Saddle River, NJ: Prentice Hall, 2001.
- [9] XML C Parser and toolkit of Gnome. [Http://www.xmlsoft.org](http://www.xmlsoft.org). [Online]. Available: <http://www.xmlsoft.org>
- [10] Xml inclusions (xinclude) version 1.0. [Http://www.w3.org/TR/xinclude/](http://www.w3.org/TR/xinclude/). [Online]. Available: <http://www.w3.org/TR/xinclude/>

A Old QCADesigner Input File – Single Cell

```

1  [VERSION]
2  qcadesigner_version=1.100000
3  [#VERSION]
4  [DESIGN_OPTIONS]
5  grid_spacing=1.000000e+01
6  [#DESIGN_OPTIONS]
7  [DESIGN_PROPERTIES]
8  total_number_of_cells=5
9  [#DESIGN_PROPERTIES]
10 [QCELL]
11 x=1.400000e+02
12 y=1.600000e+02
13 top_x=1.310000e+02
14 top_y=1.510000e+02
15 bot_x=1.490000e+02
16 bot_y=1.690000e+02
17 cell_width=1.800000e+01
18 cell_height=1.800000e+01
19 orientation=0
20 color=2

```

```

21 clock=0
22 is_input=1
23 is_output=0
24 is_fixed=0
25 label=a
26 number_of_dots=4
27 [QDOT]
28 x=1.445000e+02
29 y=1.555000e+02
30 diameter=5.000000e+00
31 charge=8.000000e-20
32 spin=0.000000e+00
33 potential=0.000000e+00
34 [#QDOT]
35 [QDOT]
36 x=1.445000e+02
37 y=1.645000e+02
38 diameter=5.000000e+00
39 charge=8.000000e-20
40 spin=0.000000e+00
41 potential=0.000000e+00
42 [#QDOT]
43 [QDOT]
44 x=1.355000e+02
45 y=1.645000e+02
46 diameter=5.000000e+00
47 charge=8.000000e-20
48 spin=0.000000e+00
49 potential=0.000000e+00
50 [#QDOT]
51 [QDOT]
52 x=1.355000e+02
53 y=1.555000e+02
54 diameter=5.000000e+00
55 charge=8.000000e-20
56 spin=0.000000e+00
57 potential=0.000000e+00
58 [#QDOT]
59 [#QCELL]

```

B Technology File

```

1 <?xml version="1.0" ?>
2 <!DOCTYPE TechFile SYSTEM "../tech.dtd">
3
4 <TechFile>
5 <CellInfo>
6 <height>5</height>
7 <width>5</width>
8 </CellInfo>
9 <DotInfo>
10 <diameter>1.000000e+01</diameter>
11 </DotInfo>

```

```

12     <Other>
13         <cd_dist>17.67767</cd_dist>
14         <angle>45</angle>
15         <lambda>1.000000e+01</lambda>
16     </Other>
17 </TechFile>

```

C Technology DTD

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!-- QCADesigner XML Tech File DTD -->
3 <!-- DTD Version 1.0 -->
4 <!-- QCADesigner Home Page: http://www.qcadesigner.ca -->
5 <!-- Copyright (c) 2003 by Timothy Dysart (tdysart@nd.edu)-->
6 <!-- University of Notre Dame -->
7 <!-- Permission is granted to use, copy, distribute and/or modify this document,
8     provided that the copyright is maintained and all derivative works are
9     released into the public domain. -->
10
11 <!-- Purpose: Define physical parameters for a QCA system -->
12
13 <!ELEMENT TechFile (CellInfo, DotInfo, Other)>
14
15 <!-- Start Cell Info -->
16 <!ELEMENT CellInfo (height, width)>
17 <!ELEMENT height (#PCDATA)>
18 <!ELEMENT width (#PCDATA)>
19
20 <!-- Start Dot Info -->
21 <!ELEMENT DotInfo (diameter)>
22 <!ELEMENT diameter (#PCDATA)>
23
24 <!-- Start Other -->
25 <!ELEMENT Other (cd_dist, angle, lambda)>
26 <!ELEMENT cd_dist (#PCDATA)>
27 <!ELEMENT angle (#PCDATA)>
28 <!ELEMENT lambda (#PCDATA)>
29
30

```

D Architecture DTD

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!-- QCADesigner XML Architecture File DTD -->
3 <!-- DTD Version 1.0 -->
4 <!-- QCADesigner Home Page: http://www.qcadesigner.ca -->
5 <!-- Copyright (c) 2003 by Timothy Dysart (tdysart@nd.edu)-->
6 <!-- University of Notre Dame -->
7 <!-- Permission is granted to use, copy, distribute and/or modify this document,
8     provided that the copyright is maintained and all derivative works are
9     released into the public domain. -->
10

```

```

11 <!-- Purpose: Define a QCADesignerProjectFile which contains all high level
12      design information for a QCA circuit -->
13
14 <!ELEMENT QCADesignerProjectFile (INIT, (CELL_LIBS | QCELL)+ )>
15 <!ATTLIST QCADesignerProjectFile xml:base CDATA #IMPLIED>
16
17 <!-- Begin INIT Stuff -->
18 <!ELEMENT INIT (QCAD_Version, Design_Opts)>
19 <!ELEMENT QCAD_Version (#PCDATA)>
20 <!ELEMENT Design_Opts (grid_spacing)>
21 <!ELEMENT grid_spacing (#PCDATA)>
22 <!-- End INIT Stuff -->
23
24 <!-- CELL LIBS Linking -->
25 <!ELEMENT CELL_LIBS ((IncludedCells|xi:include)+)>
26 <!ATTLIST CELL_LIBS begin_x CDATA #REQUIRED>
27 <!ATTLIST CELL_LIBS begin_y CDATA #REQUIRED>
28 <!ATTLIST CELL_LIBS rotate CDATA #REQUIRED>
29 <!ELEMENT IncludedCells ((QCELL|CELL_LIBS)+)>
30 <!ATTLIST IncludedCells xml:base CDATA #IMPLIED>
31
32 <!ELEMENT xi:include (xi:fallback?)>
33 <!ATTLIST xi:include
34       xmlns:xi CDATA #FIXED "http://www.w3.org/2001/XInclude">
35 <!ATTLIST xi:include
36       xi:parse CDATA #FIXED "xml">
37 <!ATTLIST xi:include
38       xi:href CDATA #REQUIRED >
39 <!ELEMENT xi:fallback ANY>
40 <!ATTLIST xi:fallback
41       xmlns:xi CDATA #FIXED "http://www.w3.org/2001/XInclude" >
42 <!-- END CELL LIBS LINKING -->
43
44 <!-- Stat QCell Stuff -->
45 <!ELEMENT QCELL (Location, Initializers, Info)>
46 <!-- Start Location info. -->
47 <!ELEMENT Location (x_center, y_center)>
48 <!ELEMENT x_center (#PCDATA)>
49 <!ELEMENT y_center (#PCDATA)>
50 <!-- Start Initializers -->
51 <!-- iof: input, output, fixed, x = don't care -->
52 <!ELEMENT Initializers (orientation, color, clock, iof)>
53 <!ELEMENT orientation (#PCDATA)>
54 <!ELEMENT color (#PCDATA)>
55 <!ELEMENT clock (#PCDATA)>
56 <!ELEMENT iof (#PCDATA)>
57 <!ATTLIST iof polarize CDATA #REQUIRED>
58 <!-- Start Info Section -->
59 <!ELEMENT Info (label)>
60 <!ELEMENT label (#PCDATA)>
61 <!-- End QCell Stuff, End DTD -->

```

E Single Cell XML File

```
1 <?xml version="1.0" standalone="no"?>
2 <!DOCTYPE QCADesignerProjectFile
3 PUBLIC " " "http://www.nd.edu/~tdysart/xml/arch.dtd">
4
5 <QCADesignerProjectFile xml:base = ".">
6   <INIT>
7     <QCAD_Version>1.100000</QCAD_Version>
8     <Design_Opts>
9       <grid_spacing>1.000000e+01</grid_spacing>
10    </Design_Opts>
11  </INIT>
12
13  <QCELL>
14    <Location>
15      <x_center>100</x_center>
16      <y_center>40</y_center>
17    </Location>
18    <Initializers>
19      <orientation>0</orientation>
20      <color>2</color>
21      <clock>0</clock>
22      <iof polarize = "0">x</iof>
23    </Initializers>
24    <Info>
25      <label>b</label>
26    </Info>
27  </QCELL>
28
29 </QCADesignerProjectFile>
```

F Cell Library DTD

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!-- QCADesigner XML IncludedCells DTD -->
3 <!-- DTD Version 1.0 -->
4 <!-- QCADesigner Home Page: http://www.qcadesigner.ca -->
5 <!-- Copyright (c) 2003 by Timothy Dysart (tdysart@nd.edu)-->
6 <!-- University of Notre Dame -->
7 <!-- Permission is granted to use, copy, distribute and/or modify this document,
8     provided that the copyright is maintained and all derivative works are
9     released into the public domain. -->
10
11 <!-- Purpose: Define a cell library -->
12
13 <!-- CELL LIBS Linking -->
14 <!ELEMENT IncludedCells ((QCELL|CELL_LIBS)+)>
15 <!ATTLIST IncludedCells xml:base CDATA #IMPLIED>
16 <!ELEMENT CELL_LIBS ((IncludedCells|xi:include)*)>
17 <!ATTLIST CELL_LIBS begin_x CDATA #REQUIRED>
18 <!ATTLIST CELL_LIBS begin_y CDATA #REQUIRED>
19 <!ATTLIST CELL_LIBS rotate CDATA #REQUIRED>
```

```

20
21 <!ELEMENT xi:include (xi:fallback?)>
22 <!ATTLIST xi:include
23     xmlns:xi CDATA #FIXED "http://www.w3.org/2001/XInclude"
24     xi:parse CDATA #FIXED "xml"
25     xi:href CDATA #REQUIRED >
26 <!ELEMENT xi:fallback ANY>
27 <!ATTLIST xi:fallback
28     xmlns:xi CDATA #FIXED "http://www.w3.org/2001/XInclude" >
29
30
31 <!ELEMENT QCELL (Location, Initializers, Info)>
32 <!-- Start Location info. -->
33 <!ELEMENT Location (x_center, y_center)>
34 <!ELEMENT x_center (#PCDATA)>
35 <!ELEMENT y_center (#PCDATA)>
36
37 <!-- Start Initializers -->
38 <!-- iof: input, output, fixed -->
39 <!ELEMENT Initializers (orientation, color, clock, iof)>
40 <!ELEMENT orientation (#PCDATA)>
41 <!ELEMENT color (#PCDATA)>
42 <!ELEMENT clock (#PCDATA)>
43 <!ELEMENT iof (#PCDATA)>
44 <!ATTLIST iof polarize CDATA #REQUIRED>
45
46 <!-- Start Info Section -->
47 <!ELEMENT Info (label)>
48 <!ELEMENT label (#PCDATA)>
49

```

G Inverter (Cell Library)

```

1 <?xml version="1.0" ?>
2 <!DOCTYPE IncludedCells PUBLIC
3 " " "http://www.nd.edu/~tdysart/xml/include.dtd">
4 <!-- INVERTER -->
5 <IncludedCells>
6
7 <QCELL>
8 <Location>
9 <x_center>0</x_center>
10 <y_center>0</y_center>
11 </Location>
12 <Initializers>
13 <orientation>0</orientation>
14 <color>2</color>
15 <clock>0</clock>
16 <iof polarize = "0">x</iof>
17 </Initializers>
18 <Info>
19 <label>NO NAME</label>
20 </Info>

```

```

21 </QCELL>
22
23 <QCELL>
24   <Location>
25     <x_center>20</x_center>
26     <y_center>0</y_center>
27   </Location>
28   <Initializers>
29     <orientation>0</orientation>
30     <color>0</color>
31     <clock>0</clock>
32     <iof polarize = "0">x</iof>
33   </Initializers>
34   <Info>
35     <label>NO NAME</label>
36   </Info>
37 </QCELL>
38
39 <QCELL>
40   <Location>
41     <x_center>20</x_center>
42     <y_center>-20</y_center>
43   </Location>
44   <Initializers>
45     <orientation>0</orientation>
46     <color>3</color>
47     <clock>0</clock>
48     <iof polarize = "0">x</iof>
49   </Initializers>
50   <Info>
51     <label>NO NAME</label>
52   </Info>
53 </QCELL>
54
55 <QCELL>
56   <Location>
57     <x_center>40</x_center>
58     <y_center>-20</y_center>
59   </Location>
60   <Initializers>
61     <orientation>0</orientation>
62     <color>3</color>
63     <clock>0</clock>
64     <iof polarize = "0">x</iof>
65   </Initializers>
66   <Info>
67     <label>NO NAME</label>
68   </Info>
69 </QCELL>
70
71 <QCELL>
72   <Location>
73     <x_center>60</x_center>
74     <y_center>-20</y_center>

```

```

75     </Location>
76     <Initializers>
77         <orientation>0</orientation>
78         <color>3</color>
79         <clock>0</clock>
80         <iof polarize = "0">x</iof>
81     </Initializers>
82     <Info>
83         <label>NO NAME</label>
84     </Info>
85 </QCELL>
86
87 <QCELL>
88     <Location>
89         <x_center>20</x_center>
90         <y_center>20</y_center>
91     </Location>
92     <Initializers>
93         <orientation>0</orientation>
94         <color>3</color>
95         <clock>0</clock>
96         <iof polarize = "0">x</iof>
97     </Initializers>
98     <Info>
99         <label>NO NAME</label>
100    </Info>
101 </QCELL>
102
103 <QCELL>
104     <Location>
105         <x_center>40</x_center>
106         <y_center>20</y_center>
107     </Location>
108     <Initializers>
109         <orientation>0</orientation>
110         <color>3</color>
111         <clock>0</clock>
112         <iof polarize = "0">x</iof>
113     </Initializers>
114     <Info>
115         <label>NO NAME</label>
116     </Info>
117 </QCELL>
118
119 <QCELL>
120     <Location>
121         <x_center>60</x_center>
122         <y_center>20</y_center>
123     </Location>
124     <Initializers>
125         <orientation>0</orientation>
126         <color>3</color>
127         <clock>0</clock>
128         <iof polarize = "0">x</iof>

```

```

129     </Initializers>
130     <Info>
131         <label>NO NAME</label>
132     </Info>
133 </QCELL>
134
135 <QCELL>
136     <Location>
137         <x_center>80</x_center>
138         <y_center>0</y_center>
139     </Location>
140     <Initializers>
141         <orientation>0</orientation>
142         <color>3</color>
143         <clock>0</clock>
144         <iof polarize = "0">x</iof>
145     </Initializers>
146     <Info>
147         <label>NO NAME</label>
148     </Info>
149 </QCELL>
150
151 </IncludedCells>

```

H NAND Gate (Cell Library)

```

1 <?xml version="1.0" ?>
2 <!DOCTYPE IncludedCells PUBLIC
3 " " "http://www.nd.edu/~tdysart/xml/include.dtd">
4
5 <IncludedCells>
6
7     <CELL_LIBS begin_x="0" begin_y="0" rotate="0">
8         <xi:include xmlns:xi="http://www.w3.org/2001/XInclude"
9             xi:href = "../Inverter.xml" />
10     </CELL_LIBS>
11
12     <QCELL>
13         <Location>
14             <x_center>100</x_center>
15             <y_center>0</y_center>
16         </Location>
17         <Initializers>
18             <orientation>0</orientation>
19             <color>2</color>
20             <clock>0</clock>
21             <iof polarize = "0">x</iof>
22         </Initializers>
23         <Info>
24             <label>NO NAME</label>
25         </Info>
26     </QCELL>
27

```

```

28 <QCELL>
29   <Location>
30     <x_center>120</x_center>
31     <y_center>0</y_center>
32   </Location>
33   <Initializers>
34     <orientation>0</orientation>
35     <color>0</color>
36     <clock>0</clock>
37     <iof polarize = "0">x</iof>
38   </Initializers>
39   <Info>
40     <label>NO NAME</label>
41   </Info>
42 </QCELL>
43
44 <QCELL>
45   <Location>
46     <x_center>120</x_center>
47     <y_center>20</y_center>
48   </Location>
49   <Initializers>
50     <orientation>0</orientation>
51     <color>3</color>
52     <clock>0</clock>
53     <iof polarize = "0">x</iof>
54   </Initializers>
55   <Info>
56     <label>NO NAME</label>
57   </Info>
58 </QCELL>
59
60 <QCELL>
61   <Location>
62     <x_center>120</x_center>
63     <y_center>40</y_center>
64   </Location>
65   <Initializers>
66     <orientation>0</orientation>
67     <color>3</color>
68     <clock>0</clock>
69     <iof polarize = "0">x</iof>
70   </Initializers>
71   <Info>
72     <label>NO NAME</label>
73   </Info>
74 </QCELL>
75
76 <QCELL>
77   <Location>
78     <x_center>120</x_center>
79     <y_center>60</y_center>
80   </Location>
81   <Initializers>

```

```

82         <orientation>0</orientation>
83         <color>3</color>
84         <clock>0</clock>
85         <iof polarize = "0">x</iof>
86     </Initializers>
87     <Info>
88         <label>NO NAME</label>
89     </Info>
90 </QCELL>
91
92 <CELL_LIBS begin_x="0" begin_y="80" rotate="0">
93     <xi:include xmlns:xi="http://www.w3.org/2001/XInclude"
94         xi:href = "Inverter.xml" />
95 </CELL_LIBS>
96
97 <QCELL>
98     <Location>
99         <x_center>100</x_center>
100        <y_center>80</y_center>
101    </Location>
102    <Initializers>
103        <orientation>0</orientation>
104        <color>3</color>
105        <clock>0</clock>
106        <iof polarize = "0">x</iof>
107    </Initializers>
108    <Info>
109        <label>NO NAME</label>
110    </Info>
111 </QCELL>
112
113 <QCELL>
114     <Location>
115         <x_center>120</x_center>
116         <y_center>80</y_center>
117     </Location>
118     <Initializers>
119         <orientation>0</orientation>
120         <color>3</color>
121         <clock>0</clock>
122         <iof polarize = "0">x</iof>
123     </Initializers>
124     <Info>
125         <label>NO NAME</label>
126     </Info>
127 </QCELL>
128
129 <QCELL>
130     <Location>
131         <x_center>100</x_center>
132         <y_center>40</y_center>
133     </Location>
134     <Initializers>
135         <orientation>0</orientation>

```

```

136         <color>3</color>
137         <clock>0</clock>
138         <iof polarize = "1">fixed</iof>
139     </Initializers>
140     <Info>
141         <label>NO NAME</label>
142     </Info>
143 </QCELL>
144
145 <QCELL>
146     <Location>
147         <x_center>140</x_center>
148         <y_center>40</y_center>
149     </Location>
150     <Initializers>
151         <orientation>0</orientation>
152         <color>3</color>
153         <clock>1</clock>
154         <iof polarize = "0">x</iof>
155     </Initializers>
156     <Info>
157         <label>NO NAME</label>
158     </Info>
159 </QCELL>
160 <QCELL>
161     <Location>
162         <x_center>160</x_center>
163         <y_center>40</y_center>
164     </Location>
165     <Initializers>
166         <orientation>0</orientation>
167         <color>3</color>
168         <clock>1</clock>
169         <iof polarize = "0">x</iof>
170     </Initializers>
171     <Info>
172         <label>NO NAME</label>
173     </Info>
174 </QCELL>
175
176 </IncludedCells>

```