# A DISTRIBUTED AND OBJECT-ORIENTED FRAMEWORK FOR VLSI PHYSICAL DESIGN AUTOMATION

*Fong-Ming Shyu* and *Sao-Jie Chen*

Department of Electrical Engineering
and Graduate Institute of Electronic Engineering
National Taiwan University
Taipei, Taiwan, R.O.C.
(E-mail: csj@cc.ee.ntu.edu.tw)

## ABSTRACT

*In this paper, we propose a framework for VLSI physical design automation. This framework is object-oriented and built in a distributed environment. The VLSI physical designers can use our framework to develop their partitioning, placement, and routing algorithms through Internet. We model our framework using Unified Modeling Language and develop a prototype example with PDML (Physical Design Markup Language), a customized XML.*

## 1. INTRODUCTION

Software application framework helps software developers to construct their prototype systems. In this paper, we propose a Distributed and Object-Oriented Framework for VLSI (DOOF/VLSI) to help physical designers to develop algorithms for partitioning, placement, and routing. Unlike the hardware components, these components are distributed on several servers and organized into packages, libraries, or repositories; thus they are easier to reuse or customize. Internet technologies have also been imported to achieve a distributed framework of components.

XML is an acronym of eXtensible Markup Language [1], which is a kind of SGML (Standard Generalized Markup Language). XML, in general, is a universal data format description used to transfer data between nodes on Internet. This kind of markup language can also be customized for dedicated purpose. For example, Gellersen and Gaedke [2] proposed a WebComposition Component Model for the web application development. In this paper, we propose a distributed framework and a customized markup language, called Physical Design Markup Language (PDML), for physical design automation. In the following, we discuss the framework for VLSI Design, show how to use UML to model the object model of a partitioning problem for our framework, and how to describe with PDML the partitioning input to the framework.

* This work was supported by the National Science Council, ROC, under Grant #NSC-89-2215-E002-045.

## 2. OBJECT-ORIENTED FRAMEWORK IN VLSI PHYSICAL DESIGN

VLSI design cycle [3] can be classified into the following design phases: (1) System specification, (2) Functional design, (3) Logic design, (4) Circuit design, (5) Physical design, (6) Fabrication, and (7) Packaging.

There have been many studies on applying object-oriented framework for Computer-Aided Design (CAD) tools to minimize the iterations of design cycle thus reduce the time-to-market. For example in the circuit design phase, Gupta et al. [4] developed a CAD framework application system, called Cbase, where he applied the OODBMS to analyze and store components for circuit design. Distributed and web-based IC design frameworks have been discussed in [5-6]. Hsiung et al. [7] successfully applied Object-Oriented technologies in the synthesis of multiprocessor system.

In fact, many other similar studies have been proposed in recent years. But it seems that there are few studies on the VLSI physical design using object-oriented techniques. Therefore, the valuable experiments performed so far inspire us to apply object-oriented technologies in physical design. We explore every step of the physical design phase and try to develop a framework for them. Since the internet technologies are popular all over the world today, we try to develop a collaborative system and build a distributed framework running on internet.

## 3. UNIFIED MODELING LANGUAGE

The Unified Modeling Language, UML, developed by the Rational Software Corporation [8], is an integration of the Booch [9], Rumbaugh [10], and Jacobson [11] methods. We first use UML to describe our DOOF/VLSI framework as follows.

UML supports the following graphical diagrams: (1) Use case diagrams, (2) Class diagrams, (3) Behavior diagrams, and (4) Implementation diagrams. Figure 1 shows the class diagram of packages used in DOOF/VLSI from where we can see how UML models a framework.



0-7803-6685-9/01/$10.00©2001 IEEE

After modeling our framework using UML graphical notations, we develop a prototype of the framework with Java, an object-oriented programming language. Then, for a specific physical design problem, e.g., partitioning, we have first to identify the partitioning object and classify it into class hierarchies. The UML model of the partitioning object is also shown in Figure 1 .

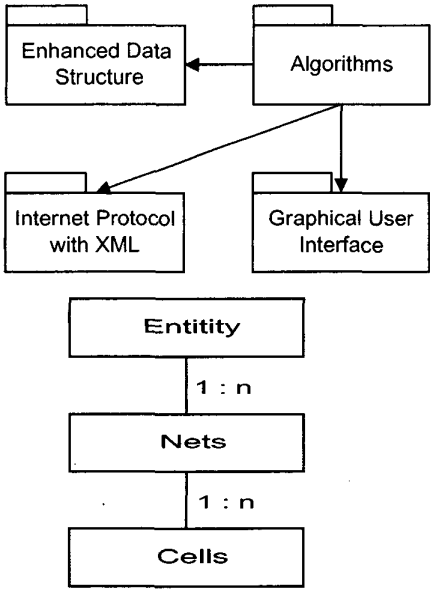


**Figure 1**. The framework packages and partitioning object model

## 4. DISTRIBUTED OBJECT FRAMEWORK

The distributed object technology is mature and has been used in Internet all over the world. Incorporate this technology into our framework will achieve a distributed environment for physical design.

For example in the partitioning step, we can separate a whole partitioning problem into several sub-problems. Each of these sub-problems can be distributed into another server side for its solution. Figure 2 depicts the configuration of a distributed framework environment.

Another distributed mechanism used in DOOF/VLSI is to store the sub-problem components in a server and to distribute these sub-components to the requiring clients. Physical designers developing their algorithms in client sides can make use of these sub-components. In this way, designers can achieve the development of algorithms in a collaborated environment. Figure 3 shows this concept.

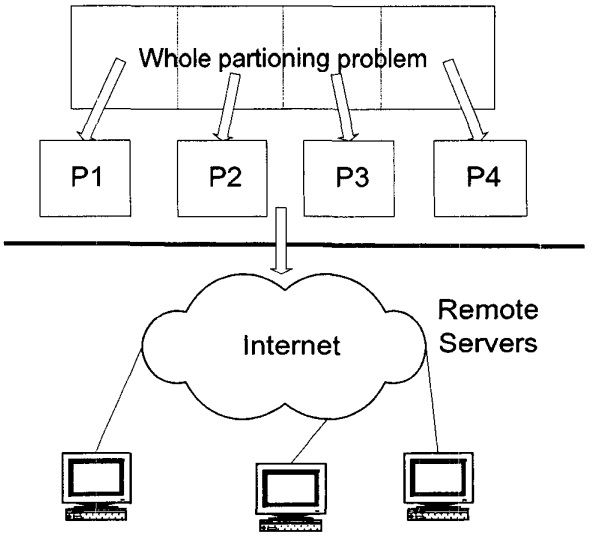


**Figure 2**. Distributed framework configuration

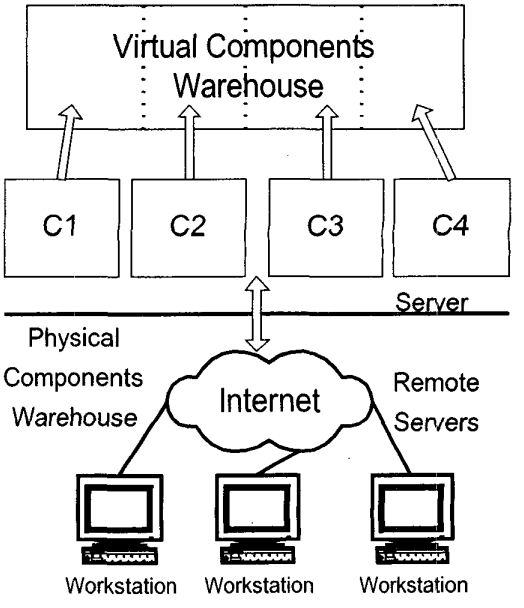


**Figure 3**. Collaborated development environment

In Figure 3, we observe that the physical design components stored in a server can also be distributed to some remote servers. This virtual components warehouse can decrease the load of a single server. Thus, we can use this environment to support the collaborated development of algorithms.

In fact, many computer languages support distributed components, e.g., Java beans RMI and Microsoft DCOM. Assume that we have already built some components on certain distributed nodes over internet, for example, the K-L [12] and F-M [13] algorithms for VLSI partitioning. How can we use these components professionally? Can these components communicate with the same protocol? XML provides a solution. We can define a DTD (Data Type Definition) for the object model of each component and place it on the internet. Then, the distributed nodes can retrieve and use these XML-based component descriptions by following what described in DTD. Moreover, since these XML-based descriptions containing real data can be placed on any internet node, any application system can read these XML-based descriptions, parse them by the DTD files, and then solve the modeled (here partitioning) problem. Figure 4 shows the implementation configuration of our XML-based framework.

In Figure 4, components are placed in different nodes on internet, e.g., the DTD files and XML descriptions are placed in different remote nodes. We can create a web server to handle user requirements and provide services to solve the problem. This kind of service is called "Application Service Provider". In next section, we show an example for the DTD and XML-base description of a partitioning problem. Then, a Java version of the K-L algorithm (pre-stored in a remote server node) can be used to solve this partition problem through our framework.

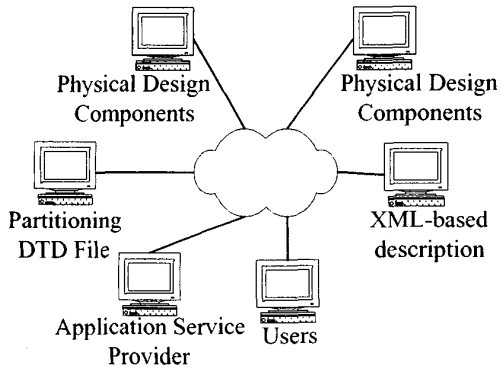


**Figure 4**. Distributed framework configuration

## 5. AN EXAMPLE OF SOLVING PARTITIONING WITH XML

According to the partitioning object model Figure 1, the DTD file of a partitioning problem can be defined as follows:

```
<?xml version='1.0' encoding='us-ascii'?>
<!--
    DTD for a "Partitioning Problem in VLSI Physical Design ".
-->
<!ELEMENT NET (CELL+)>
<!ELEMENT CELL (#PCDATA)*>
```

This partitioning DTD is very simple. It shows that we have many NETs and each NET connects to CELL(s). The #PCDATA is the data format tagged by <CELL> in XML description. It can be any data type, likes numeric or alpha-numeric. Then, a detailed XML-based description file for the to-be-partitioned circuit in Figure 5 is listed as follows:

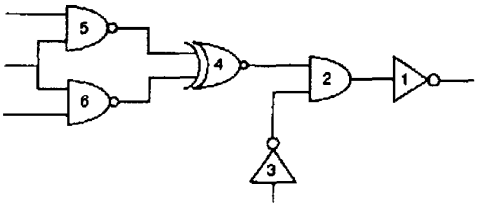


**Figure 5**. The entity of a circuit for partitioning [3]

```
<?xml version='1.0' encoding='us-ascii'?>
<!--  A SAMPLE of Physical Design Partitioning  -->
<!DOCTYPE PHDPartition SYSTEM "partition.dtd">
<PHDPartition>
  <K-L Aogirithm>
    Remote Server A
  </K-L Aogirithm>
  <NET>
    <CELL>C2</CELL>
    <CELL>C4</CELL>
  </NET>
  <NET>
    <CELL>C2</CELL>
    <CELL>C3</CELL>
  </NET>
  <NET>
    <CELL>C1</CELL>
    <CELL>C2</CELL>
  </NET>
  <NET>
    <CELL>C4</CELL>
    <CELL>C5</CELL>
  </NET>
  <NET>
    <CELL>C4</CELL>
    <CELL>C6</CELL>
  </NET>
  <NET>
    <CELL>C5</CELL>
    <CELL>C6</CELL>
```

```
    </NET>
</PHDPartition>
```

The line <!DOCTYPE PHDPartition SYSTEM "partition.dtd"> describes that the document type definition is in "partition.dtd". This "partition.dtd" file can be placed on a remote node over internet by setting a URL (Universal Resource Location) for this file. We can also see the above description looks like an HTML (Hyper-Text Markup Language) description. The difference is in the tag "<some tag name>". The tag in our description is customized and it must follow the definition in the DTD file. So the remote node can understand our XML description from the DTD file. In implementation, a Java application program has been developed to parse the XML description and then a Java version of the K-L 2-way partitioning component, pre-stored in a remote server (Server A), is called to partition this circuit. Figure 6 presents the result after partitioning, which left window frame shows the document object model and which right frame shows the input data parsed by the XML parser and the partitioning result.

Similarly, we can modify the XML description to use another partitioning (e.g., the F-M) algorithm stored in another remote server (Server B) as follows.

```
<?xml version='1.0' encoding='us-ascii'?>
<!--  A SAMPLE of Physical Design Partitioning  -->
<!DOCTYPE PHDPartition SYSTEM "partition.dtd">
<PHDPartition>
    <F-M Algorithm>
        Remote Server B
    </F-M Algorithm>
    <NET>
        <CELL>C2</CELL>
        ...
```

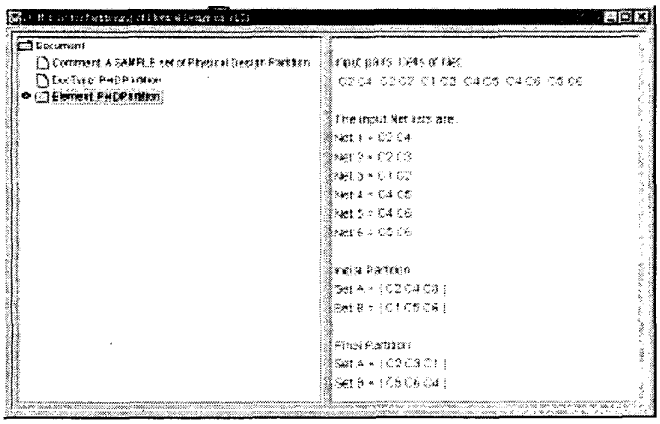

**Figure 6**. The result of partitioning automation

## 6. CONCLUSION AND FUTURE WORK

In this article, we presented a framework of DOOF/VLSI to help VLSI physical designers developing their algorithms and components library. Designers can use DOOF/VLSI either to develop new algorithms, or to reuse conventional algorithms for physical design automation easily.

With the example in this paper, we show how to apply XML for partitioning automation through our DOOF/VLSI distributed object-oriented framework. This customized XML can be easily expanded into a complete PDML (Physical Design Markup Language) to handle more complex descriptions for other physical design automation problems.

## 7. REFERENCES


[1] http://www.w3.org/XML

[2] H-W. Gellersen and M. Gaedke, "Object-Oriented Web Application Development," *IEEE Internet Computing*, pp. 60-68, Jan-Feb 1999.

[3] N. Sherwani, *Algorithms for VLSI Physical Design Automation*, 3rd Ed., ©1999, Kluwer Academic Publishers.

[4] R. Gupta, W. H. Cheng, I. Hardibag, and M. A. Breuer, "An Object-Oriented VLSI CAD Framework: A Case Study in Rapid Prototyping," *IEEE Computer*, 22(5), pp. 28-37, May 1989.

[5] L. Geppert, "IC Design on the World Wide Web," *IEEE Spectrum*, pp. 45-50, June 1998.

[6] D. Saha and A. P. Chandrakasan, "Web-based Distributed VLSI Design," in *Proc. IEEE Internaltional Confercence on VLSI Design*, Jan. 1998, pp.449-454.

[7] P. A. Hsiung, S. J. Chen, T. C. Hu, and S. C. Wang, "PSM: An Object-Oriented Synthesis Approach to Multiprocessor System Design," *IEEE Trans. on VLSI Systems*, Vol. 4, No. 1, pp. 83-97, 1996.

[8] Rational Software Corporation, *Unified Modeling Language Version 1.0*, Jan. 1997.

[9] G. Booch, *Object-Oriented Analysis and Design with Applications*, 2nd Ed. Benjamin Cummings Pub., 1997.

[10] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, *Object-Oriented Modeling and Design*, Prentice-Hall, 1991.

[11] I. Jacobson, M. Christerson, P. Johnson, and G. Overgaard, *Object-Oriented Software Engineering*, Addison-Wesley Publishing, 1992.

[12] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell System Technical Journal*, vol. 49, no. 2, pp. 291-307, Feb. 1970.

[13] C. M. Fiduccia and R. M. Mattheyses, "A linear-time heuristic for improving network partition," in *Proc. ACM/IEEE Design Automation Conf.*, 1982, pp. 175-181.