An Object Oriented CAD Project Database

Hewijin Christine Jiau Kuo-Feng Ssu

Department of Electrical Engineering
National Cheng Kung University, Taiwan, R.O.C.
Email: {jiauhjc,ssu}@ee.ncku.edu.tw

ABSTRACT

Long-lived transaction always causes the bottle neck problem in different design work application domain. CAD VLSI design transactions are typically of long duration with frequently concurrent read-write and read-read accesses. A single design transaction can last for hours, days, even months before it completes. So a long duration transaction will cause long duration locks; we will have long blocking when we use a common concurrency control algorithm. Based on this analysis, we proposed a distributed and open control mechanism with multilevel transaction manager can be effectively used to drive the design process while not restricting how the designer interacts with the tools. As the design process becomes more automated, the designer spends less time directly manipulating the design data and more time managing CAD tools that manipulate the design data. Overall, design data manipulation shifts more towards the CAD tools and away from direct designer intervention.

1 INTRODUCTION

From a lot of books [1, 2] and papers [3, 4] which address the problems of the CAD VLSI design problem and suggest the goals of the design, we found the CAD VLSI design problem is to generate an acceptable design in limited time and with limited resources that conforms to the chip cavity size and pad location. Especially, as we know, VLSI design is an iterative process. So the CAD tools must deal with the entire life-cycle of a design. The detail steps are requirement, decomposition, specification, design implementation choice, analysis and verification, real implementation, testing, documentation, manufacturing, user feedback, evaluation. There can be plenty of different ways to do these steps. So there is no perfect answer for any design. Due to the changing nature of VLSI design, a design system will never be “finished". In order to keep up with the needs of the chip designers, the environment and the data representations must be kept flexible and extensible.

To make the CAD VLSI design task of filling the void on a VLSI chip manageable, an abstraction hierarchy is widely accepted. From the latest design information we have, the following list of design levels will be most reasonable: system, behavioral, register transfer level (RTL), logic and implementation as shown in Figure 1 [5]. There are a lot of facilities needed in the whole design process, like an extensive library of predefined parts, a chosen design style (step). In a simplistic view, the overall design task (problem) can be structured in a top-down manner into simpler subtasks with clearly defined functions. Then the design task is finished. But in practice, there will be designers who, familiar with the implementation technology, will explore other good solutions for generic functions in a given technology in a bottom-up fashion. So the complexity of VLSI chips requires the use of both top-down and bottom-up design approaches, which can meet in the middle and permit completion of the design. However, for a design framework system, it is unlikely that the design flow will happen in this way. Because the natural building blocks must first be defined or discovered, only then can the architecture be modified and partitioned by the synthesis tools properly. It is the key reason why it is so important to keep flexibility in the design framework, to add new technology and to integrate new available tools [6].

In the whole CAD design process, in different hierarchical levels, there are different types of tools to help the designer. Generally speaking, there are five different kinds of design tools: high-level decision tools, analysis tools, synthesis tools, optimization tools, checking and verification tools [1]. There is another important element in VLSI design which plays an important role in making design decisions: constraints. Constraints limit the search for a design by ruling out certain decision options at all steps of the design process. These constraint changes usually cause extensive back-tracking. Therefore, a design environment must provide convenient data structures for recording and propagating constraints. Because when more and more

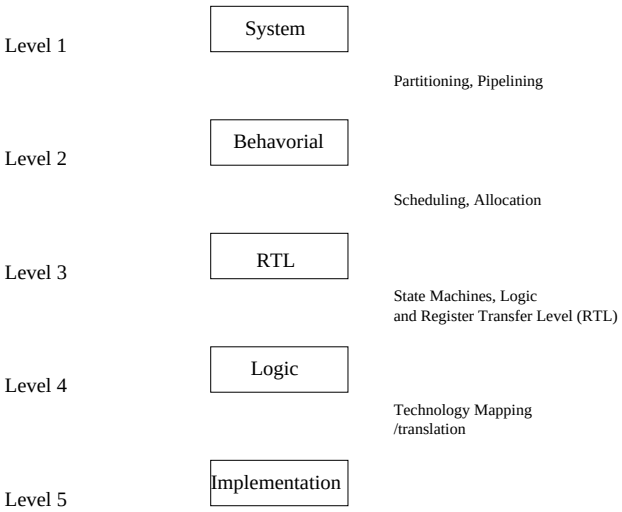

Figure 1. Design level hierarchies.

new technology comes to the market, the design should be modified and new technology fitted into the new design. Again in order to keep up with the needs of these expected changes, the CAD framework must be kept flexible and extensible. A modular set of tools coupled to an object-oriented, integrated database is a good solution to base a framework on [7, 8].

2 RELATED WORK

Many people have recognized the problems in integrating various existing CAD tools into a design environment. This integration is important for several reasons. The rapidly increasing transistor densities of IC's have drastically increased the complexities of individual chips so that the predominant life cycle cost for a chip is the design cost, rather than the manufacturing cost. By further automating the design process, the design time itself can be shortened. However, such automation requires the development of a design automation system in addition to computer-aided-design tools.

Traditionally, the designer manually executes each CAD tool individually and is responsible for managing the various design files used as inputs to these tools or created as outputs by the tools. Then several systems used different strategies to relieve the designer of such detail. ULYSSES [9] is the famous one that interprets descriptions of various design tasks, automatically invokes CAD tools to perform portions of these tasks, and manages the various files associated with each tool. Cadweld [10, 11] is presented as an extension of ULYSSES and forms the basis for a new way to model and implement a distributed control mechanism for a large population of heterogeneous CAD tools.

One problem in Cadweld is that it does not consider the hierarchical characteristic of CAD design flow. The tools can be organized according to the hierarchy to get better performance. Another problem we found in Cadweld is that there is no transaction manager to help to make the whole design framework work. Cadweld did not take advantage of the DBMS. Also as we know, only the teamwork of tools within CAD-toolboxes, controlled from a design flow manager and supported by a design database system, makes large designs possible. Siepmann provides another framework, the PLAYOUT design system [12, 13], which supports hierarchical decomposition, multiple representations, realization alternatives, versioning and concurrency. It includes a universal meta data schema, based on an object-oriented data model. The design work is done by toolboxes. This view motivates us to handle toolboxes instead of tools inside of our framework.

3 SYSTEM MODEL

Transaction plays the key role in both traditional and object-oriented world. The transaction is one of the main reason we think it will be very helpful to introduce databases into the CAD world. A so called long-lived transaction is a typical problem that exists in a CAD design system [14, 15]. A long-lived transaction is an active transaction which persists for hours, days or months, which is typical for a CAD design, especially as the design becomes larger and larger. In any CAD VLSI development, it is well-known that long-lived and nested transactions are two main concerns for the transaction management (TM) system in the framework. The support of early decisions and adjustments provided by the framework is important to help the whole design. In our framework, the use of hierarchal design and multilevel transactions can help to reach the design goal; of course, coordinating with the blackboard structure using the object specification [16].

Another key feature for the framework to provide is the design environment that the user will be involved in when the design work is in process. From the design engineer's perspective, the design environment consists of the set of design tools used during the design process and a design manager for managing these tools and their data. Because design tools and design management capabilities evolve over time, and because the set of design tools used within a design process change over time, one of the primary requirements on an environment is the rapid integration, extension, and modification of tools. So there is a need to develop an open and distributed control mechanism for the integrated design environment. The problem for the traditionally tightly integrated design environment is that new tools are difficult to add, old tools are difficult to delete. So the new approach should focus on handling the increasing number and complexity of CAD tools used during the design process and allowing easy integration at either control or data level.The communication between the toolboxes and the blackboard is performed according to the client-server

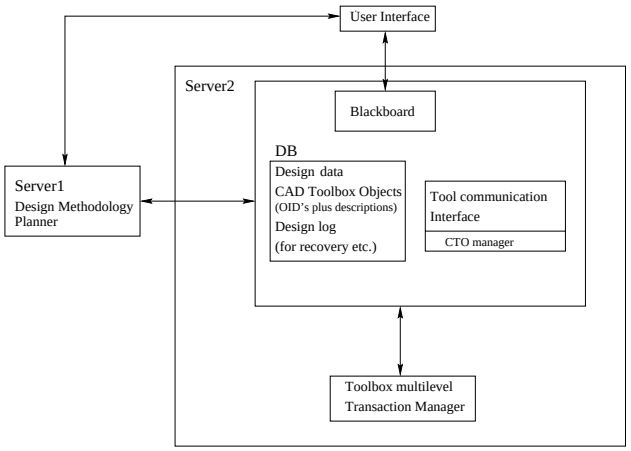

Figure 2. Elements of the framework.

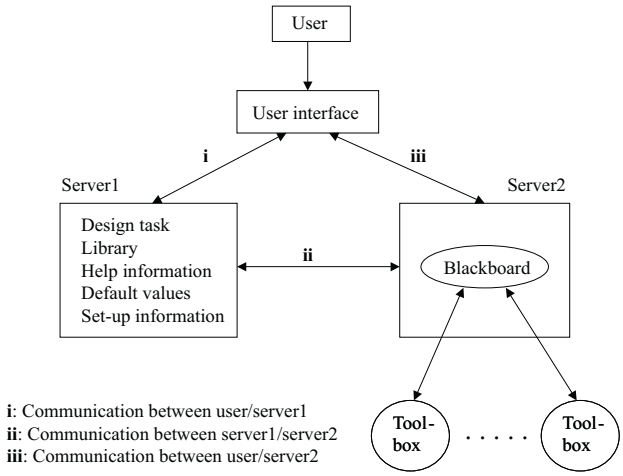

Figure 3. Communication between servers and user.

model, using the RPC (remote procedure call) mechanism, also between the tools and the toolbox.

Because of the complexity of real VLSI-designs and the NP-completeness of many design algorithms, a hierarchical design methodology is necessary. We will organize the tools into toolboxes and handle the toolbox as an object for the design work in the framework. The framework does not make any assumptions about the actual detailed design descriptions contained in those design objects (task, blackboard, tool and toolbox). A toolbox consists of a number of algorithms (tools) and a common internal data structure, thus the tools in a toolbox can communicate very fast, which is one of the basic reasons for structuring these tools into a toolbox. Handling the toolbox as an object inside of the framework not only can make a large design easier, it can also reduce the heavy traffic problem in communicating with the blackboard for every tool involved because communication through the blackboard only happens between toolboxes, not between every tool [17].

Our framework includes 2 servers. Figure 2 shows the high level architecture. Server1 contains the design methodology planner and the design tasks. A design task can be in any format constructed by a hardware description language, like VHDL [18, 19], VERILOG HDL [5, 20] or by CFG (control flow graph)[MPC90], DFG (data flow graph) [21] or CDFG (control data flow graph) [22] to represent the CAD VLSI design problem. The problem includes the preconditions and goal setting, etc. Inside server2 are the CAD project database, toolbox transaction manager, blackboard, design database, design log and the CAD toolbox object manager.

Server 1 contains the design methodology planner, design tasks and some default and help information for the tools and toolboxes. The planner will assist the designer with the selection of CAD toolboxes. It will provide the designer with information about toolbox usage, predictions about the efficiency/performance of toolboxes, the availability of toolboxes, toolbox capabilities, existence of toolbox communication problems, and a high-level mechanism to match the designer's needs with the abilities of the toolboxes. There might be a number of design tasks stored in server1. These design tasks can include successful previous design developments which have been logged, then stored back to server1 for the beginner to learn about how the framework works. A design task can also be a real design problem which is ready for the designer to solve or a specified design work in some legal format.

4 COMMUNICATION SYSTEM

Now let's discuss the communication between the main elements in the framework when the design work is in process. Figure 3 provides a clearer picture.

As we see in Figure 3, link i represents the communication between user and server1. Link ii represents the communication between server1 and server2. Link iii represents the communication between server2 and user. Now let's see, when the process begins, how communication works between those three.

We will analyze two different cases here. The first one will start by choosing an existing task in server1. The number in brackets indicates the communication link used.

- step A: User picks a design task from server1 (i). The design task is passed to server2 (ii).
- step B: Design starts, blackboard works with toolboxes and gets help or feedback from the user (iii).
- step C: During the design process, server1 is requested to provide some default values or database help (ii). (The database in server2 provides more local help

about tools or toolboxes. The database in server1 provides more global design help and some adjustment policy rules etc.)

- step D: If the design fails, do nothing. If the design succeeds, send the successful design sequence back to the database in server1 (ii). Also write results to the database of the results in server2 and report to the user (iii).

The other case is that the user inputs a design task for a special design.

- step A: User inputs the task specification by file or other method to server2 directly (iii).
- step B: Design starts, blackboard works with toolboxes and gets help or feedback from the user (iii).
- step C: During the design process, server1 is requested to provide some default values, adjustment or database help besides the database in server2 (ii).
- step D: Same as step D in previous case.

5 SERVER ANALYSIS

We look at the trade-offs between having two servers vs. one. The main disadvantage of two servers is the added communication between server1 and server2. But this communication load depends on whether the design process can rely on the CTO (CAD toolbox object) and blackboard to get most of the design work done, instead of getting a lot of help or information from server1. If little help from server1 is needed, the two-server system will probably have better performance than the one-server. There will not be a lot of traffic between the blackboard and CTOs. But if the design needs a lot of information stored in server1 to help with the design, one-server will have better performance than the two-server system. Communication is a main consideration when we try to organize the information about tools, toolboxes, and databases for server1 and server2, as well as the meta data for the blackboard. If we can get this information organized well, the two-server system will have the advantage of performance and flexibility [23].

Through the user interface, the designer can communicate with the blackboard and the toolbox transaction manager. The blackboard of server2 is the main part that communicates with the planner in server1, and it is in the blackboard that the CAD design task is stored, after a task is chosen from server1 to start the design process. Inside of server2, there is also the CAD toolbox object manager. Toolboxes are handled instead of tools during the design communication and interaction. No one can touch or use tools directly without going through the toolbox to which the tools belong. CTO contains detailed information about the CAD tools, represented by a new abstract type that is created in order for the existing CAD tools to interact with the blackboard. We also apply the contract-net mechanism to the blackboard model because we want to keep the same dynamically opportunistic control. The toolboxes will be treated as objects in their own right, capable of being modeled and manipulated flexibly by the CTO manager in server2.

6 CAD FRAMEWORK

Figure 2 shows the interrelationship among main elements in the server2. There will be various goals and intermediate results stored on the blackboard [6]. They will cause the execution of CAD toolboxes, the generation of new internal data and the modification of information that already exists on the blackboard. The designer can interact with the blackboard through the user interface when the designer wants to make decisions in cooperation with the blackboard for the usage of toolboxes. After a special toolbox is chosen by the initial agent, the blackboard sends the CTO of the chosen toolbox to the transaction manager to initiate processing. This kind of higher level transaction might be a nested transaction inside a single toolbox, but handling the inner nested transaction becomes the responsibility of the transaction manager. Handling the inner nested transaction this way will decrease the traffic between the blackboard and the transaction manager. The CAD project database refers to not only the central database facility, but also to all files and artifacts produced through the invocation of the CAD toolboxes. In the design log, we would like to store all the choices of toolboxes, no matter by the initial agent (toolbox), blackboard or the designer (through the user interface with the help of the blackboard).

7 TOOLBOX MULTILEVEL TRANSACTION MANAGER

Because of the characteristics of CAD VLSI design, the hierarchical design process should be applied to all types of chip designs. The traditional transaction will no longer fit the CAD transaction requirement. The definition of the traditional transaction manager can be found in [14].

The transaction manager here will handle design transactions. Similar to traditional transactions, the definition of a design transaction is “A sequence of database operations that maps a consistent version of a design into a new consistent version” [24]. Because of the hierarchical nature of VLSI design, we choose multilevel transactions to represent the design transactions in our framework. Multilevel transactions preserve two fundamental virtues of the classical transaction concept [25]:

1. They are based on a rigorous theoretical foundation that preserves the classical serializability theory as a special case.

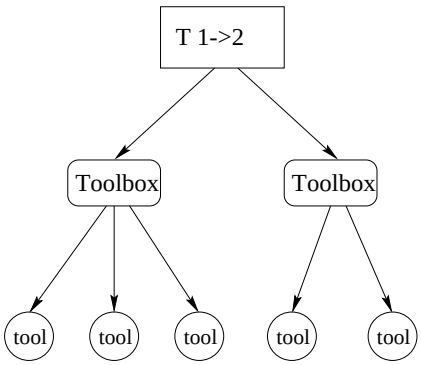

Figure 4. Transaction example.

2. They can reuse much of the well-proven implementation techniques that account for the high performance of current transaction processing systems.

So the transaction manager here will handle design transactions. In our framework, a transaction is invoked to take the design from level n to level n+1. Each transaction will be associated with several toolboxes. Every toolbox will contain several tools. After every tool in a toolbox finishes its job, the toolbox can commit. After every toolbox commits, the transaction can commit. So at every level a committed transaction has created a design accessible to other transactions, etc.. This follows the nested transaction model. Because the transaction manager works with the blackboard, it permits multiple individual queries to be executed as a single atomic operation. For example, to correctly implement a “reference" operation for the design objects (blackboard, toolboxes), multiple queries have to be performed atomically to check existence of those objects, check running design transactions on the object, and, if allowed and necessary, administer the new design transaction.

See Figure 4 for a transaction example. “T $1 \rightarrow 2$" indicates that this is a transaction taking the design from “level 1" to “level 2".

8 SIMULATION MODELS AND RESULTS

After we evaluated several possible simulation tools, we chose UltraSAN for our modeling simulation tool because it fits most of our requirements.

The one-server model was constructed as shown in Figure 5. Figure 6 shows the two-server model. Because the purpose of the simulation is to study the blackboard and other queue lengths to see if the two-server architecture solves the bottleneck problem, we only modeled the subtasks inside of the framework. UltraSAN is a very good tool for this kind of simulation, but it also has a limit on task size. When we ran the simulation model, we found that the task_size must be limited to 20 (i.e., 20 tasks are run in the model initially). Otherwise the number of generated states

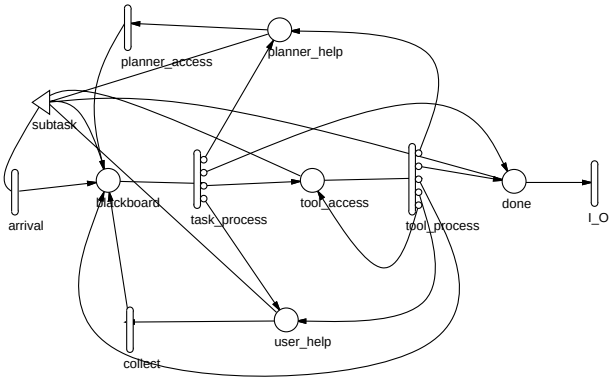

Figure 5. One-server model.

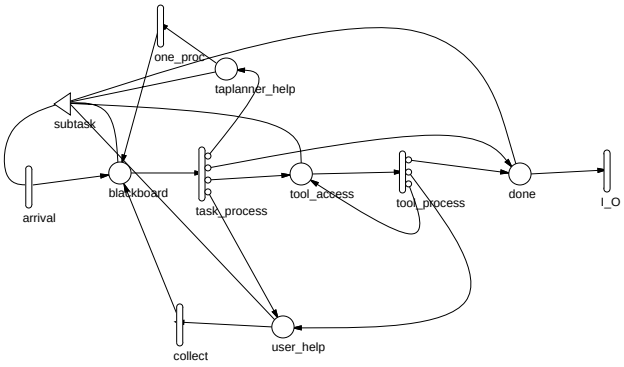

Figure 6. Two-server model.

exceeds the tool capacity. That’s why all of our results are with the same initial task size.

In the one-server model, we have five “places" which correspond to five different components of the modeled system. These components are the blackboard (the place represents a queue of subtasks waiting for blackboard posting and help), planner_help (the place represents a queue of subtasks waiting for planner help), user_help (the place represents a queue of subtasks waiting for user input help), tool_access (the place represents a queue of subtasks waiting to access the chosen toolbox) and done (the place represents a queue of subtasks waiting for I/O). In the one-server model, we also have six activities, which represent actions that take some specified amount of time to complete. Those activities are “timed type" in UltraSAN, which means they have durations which impact the performance of the modeled system. In Figure 5, arrival (the activity represents the time between arrivals of subtasks to the system), planner_access (the activity represents the time for the system to access the right planner for the subtask), collect (the activity represents the time for the system to collect the user input help for the subtask), task_process (the activity represents the time to process the subtask according to different cases), tool_process (the activity represents the time for tool

processing requested for the subtask) and I_O (the activity represents the time to send the subtask to I/O (the storage of results)) are timed activities. Each timed activity has an activity time distribution function associated with its duration. Among these timed activities, two of them contain case probabilities, represented graphically as circles on the right side of the activity, which model uncertainty associated with the completion of the activity. Each case stands for a possible outcome. In the one-server model, activity task_process has four cases. Each activity has a probability distribution, called the case distribution associated with its cases. With probability ta1_prob, case one is chosen, meaning the task needs planner_help to do further processing. With probability ta2_prob, case two is chosen, meaning the task cannot be done because of error or designer's choice in the controlled-automatic or manual design mode. With probability ta3_prob, case three is chosen, meaning the task finishes the task processing and is ready for the toolboxes to start the design work. With probability ta4_prob, case four is chosen, meaning the task needs user_input help for the task. The values we set for those probabilities are from the analysis of the design flow and the experience when we tried some of the CAD tools. That's why we set the probability for task to get the planner_help is 0.2, the probability for task to be dropped in the framework is 0.1, the probability for task finishing the processing is 0.5 and the probability for task to get user_help is 0.2. For those activities where no circles are shown on the right side, one case is assumed with a probability of one. The activity tool_process has five cases as shown in Figure 5. With probability to1_prob, case one is chosen, meaning the tool needs to get some information from the planner_help. With probability to2_prob, case two is chosen, meaning the tool finishes the design work and is ready for I/O. With probability to3_prob, case three is chosen, meaning the tool cannot get the assigned work done and needs to post the help on the blackboard. With probability to4_prob, case four is chosen, meaning the tool needs to get information or a decision from user_help. With probability to5_prob, case five is chosen, meaning the tool has to access another tool for the design work. The setting for those values are from the analysis and the experience.

The model also has one input gate (subtask) which controls the enabling of activities and defines the marking changes that will occur when an activity completes. On the other side of the triangle indicating the input gate are a set of arcs to the places upon which the gate depends, also called input places. In the one-server model, blackboard, planner_help, tool_access, user_help and done are the input places.

In Figure 6 is the model for the two-server structure. As we can see, this model also has five places, blackboard, taplanner_help, user_help, tool_access and done. Also there are six timed activities, arrival (same as one-server model), one_proc (the activity represents the time for the system to process the plan from server1 for the subtask), collect (same as one-server model), task_process (the activity represents the time to process a subtask according to different cases, which are different from those of the one-server model on account of the need to access server1), tool_process (the activity represents the time for tool processing) and I_O (same as one-server model). The two major differences between one-server and two-server models are the activities, the task_process and the tool_process. Because of the different architecture, the timed activity tool_process has only three cases instead of the five in the one-server model. After the designer chooses the design task in server1 or inputs the design task to the server1, some related information in server1 will attach with the task when the design task is posted on the blackboard in the server2. So there is no need for planner_help at this level. The task_process activity has four cases as in the one-server model, but for case one, the system will go to server1 to get task planner help instead of searching inside of the system as in the one-server model. Of course, the probability settings are different for the one-server and two-server models because of their different architectures. For task_process, with probability ta1_prob, case one is chosen, meaning the task needs the planner_help to do further processing. With probability ta2_prob, case two is chosen, meaning the task cannot be done because of error or designer's choice in the controlled-automatic or manual design mode. With probability ta3_prob, case three is chosen, meaning the task finishes the regular processing and is ready for the toolboxes to start the design work. With probability ta4_prob, case four is chosen, meaning the task needs user_input help for the task.The activity tool_process has three cases as shown in Figure 6. With probability to1_prob, case one is chosen, meaning the tool finishes the design work and is ready for I/O. With probability to2_prob, case two is chosen, meaning the tool needs to get information or a decision from user_help. With probability to3_prob, case three is chosen, meaning the tool has to access another tool. In the two-server model, planner_help will be attached to the task after the task is chosen in server1, so there is no need to get planner_help during the tool processing. Also the tool will not go directly to the blackboard for the posting help because of the server structure. That's why the cases in the two-server model for the tool_process are three.

The main purpose for this simulation work is to compare the queue lengths of the blackboard in the one-server and two-server models. So we studied different situations using different input and process rates for the activities and compared the blackboard and other queue lengths. The first part of our results is based on a high input rate with fixed task_size, task_process probabilities and tool_process probabilities. Then we analyze the performance by test-

	blackboard_len	tool_len	planner_len	user_len	done_len
task_rate=2 tool_rate=30 planner_rate=2	Mean 19.37 Variance 0.7849	0.050 0.0525	0.3725 0.5113	0.1217 0.1366	0.0670 0.07157
task_rate=2 tool_rate=30 planner_rate=20	Mean 19.72 Variance 0.3022	0.050 0.0525	0.0279 0.02867	0.1217 0.1366	0.0670 0.07157
task_rate=2 tool_rate=2 planner_rate=10	Mean 17.26 Variance 8.623	2.480 8.345	0.0573 0.0606	0.1217 0.1365	0.0670 0.07153
task_rate=30 tool_rate=2 planner_rate=10	Mean 0.1029 Variance 0.01135	19.521 0.5379	0.08225 0.08901	0.1792 0.21137	0.09649 0.1058
task_rate=20 tool_rate=20 planner_rate=10	Mean 6.0269 Variance 22.936	1.7176 4.4032	0.9323 1.7689	9.9358 26.433	1.260 2.7613
task_rate=5 tool_rate=10 planner_rate=10	Mean 18.695 Variance 1.8119	0.5555 0.8641	0.1570 0.1816	0.3725 0.51134	0.1864 0.2212

arr_rate is the value set for arrival activity
task_size is the value set for the task size
task_rate is the value set for task_process activity
tool_rate is the value set for tool_process activity
planner_rate is the valus set for planner_access activity

Figure 7. Simulation results for one-server model with high arrival rate. Table entries are queue lengths with arr_rate=50, task_size=20.

	blackboard_len	tool_len	planner_len	user_len	done_len
task_rate=2 tool_rate=30 one_rate=2	Mean 19.37 Variance 0.7487	0.050 0.0525	0.250 0.3125	0.2411 0.2992	0.067 0.0715
task_rate=2 tool_rate=30 one_rate=20	Mean 19.60 Variance 0.457	0.050 0.0525	0.0204 0.0208	0.2411 0.2992	0.067 0.0715
task_rate=2 tool_rate=2 one_rate=10	Mean 17.15 Variance 8.754	2.479 8.333	0.0416 0.04338	0.241 0.299	0.0670 0.0715
task_rate=30 tool_rate=2 one_rate=10	Mean 0.1029 Variance 0.1135	19.349 0.8136	0.05932 0.06284	0.3736 0.5132	0.0964 0.1058
task_rate=20 tool_rate=20 one_rate=10	Mean 1.060 Variance 2.184	0.5813 0.9193	0.2592 0.3264	17.55 4.210	0.7782 0.7069
task_rate=5 tool_rate=10 one_rate=10	Mean 18.17 Variance 3.078	0.555 0.8641	0.111 0.1234	0.9444 1.836	0.1864 0.2212

arr_rate is the value set for arrival activity
task_size is the value set for task size
task_rate is the value set for task_process activity
tool_rate is the value set for tool_process activity
one_rate is the value set for one_proc activity

Figure 8. Simulation results for two-server model with high arrival rate. Table entries are queue lengths with arr_rate=50, task_size=20.

ing different (high/low) values of task_rate, tool_rate and planner_rate. Figure 7 gives results for the one-server model. Figure 8 shows comparable results for the two-server model.

From Figure 7 and Figure 8 we get the following conclusions:

- Task_rate plays a key role in the blackboard queue length.
- When the task_rate is either very high or very low, blackboard queue lengths for one-server and two-server behave almost the same.
    1. When the task_rate is very high, one-server and two-server have no obvious accumulation in the blackboard queue.
    2. When the task_rate is very low, one-server and two-server models both develop a blackboard queue bottleneck.
- But when the task_rate is medium, the two-server model has smaller queue length than the one-server model. We believe this should be the most common case for CAD design. So the two-server model wins when the situation is normal.

To test and support our conclusion that the two-server model is much better for the normal case, we fix the arrival rate to normal (because the task_size is 20, when arr_rate=20, the input rate matches the size) and run some more simulations. Figure 9 shows results for the one-server model and Figure 10 shows results for the two-server structure.

From these simulation results, we see that the two-server model can really solve the bottleneck problem that arises in the blackboard of the one-server model. Also, the two-server model can usually provide faster service for task planner help, but when we set one_rate for the one_proc activity in the two-server model, we still set it the same as the planner_rate in the one-server model to avoid taking too much advantage of this processing difference.

9 CONCLUSIONS

Because the reason for developing the two-server model instead of the one-server model is the bottleneck problem in the blackboard, as we mentioned earlier, we did simulate the alternatives to show that the two-server model does get that problem solved. We also compare the performance in different kind of situations, as different input rate, different input task size, different tool process rate and the task to subtask re-submitting, etc. We study the queue lengths of blackboard components in different situations to evaluate our two-server model vs the one-server model. When we constructed the model, we found that it is not necessary to simulate the whole system if we only

	blackboard_len	tool_len	planner_len	user_len	done_len
task_rate=10 tool_rate=10 planner_rate=20	Mean 15.342 Variance 11.863	2.459 8.046	0.37194 0.5099	1.1826 2.5644	0.4575 0.6662
task_rate=5 tool_rate=10 planner_rate=10	Mean 18.643 Variance 1.8709	0.555 0.8641	0.1570 0.1816	0.3725 0.5113	0.1864 0.2212

Figure 9. Simulation results for one-server model with normal arrival rate. Table entries are queue lengths with arr_rate=20, task_size=20.

	blackboard_len	tool_len	planner_len	user_len	done_len
task_rate=10 tool_rate=10 one_rate=20	Mean 9.410 Variance 29.96	2.088 5.977	0.1056 0.1167	7.790 28.89	0.4284 0.6091
task_rate=5 tool_rate=10 one_rate=10	Mean 18.11 Variance 3.137	0.555 0.8641	0.1111 0.1234	0.9444 1.8361	0.18644 0.2212

Figure 10. Simulation results for two-server model with normal arrival rate. Table entries are queue lengths with arr_rate=20, task_size=20.

want to compare the performance of the blackboard, so we removed the multiple-user input part and only model the subtasks which are processed in the framework. We think this simplification is reasonable for the purpose of the simulation.

In this paper, we gave a brief introduction to a CAD design framework that contains two servers. The main element in server1 is the design methodology planner which can provide help with a design in process. In server2, the main elements can be the blackboard, CAD project database, CTO manager, multilevel transaction manager, which are responsible for the design. We can apply this kind of similar framework model in any design work application domain to take all the advantage which we specify in this paper.

REFERENCES

[1] W. Fichtner and M. Morf, *VLSI CAD Tools and Applications*. Kluwer Academic Publishers, 1987.

[2] D. D. Hill and D. R. Coelho, *Multi-level Simulation for VLSI Design*. Kluwer Academic Publishers, 1987.

[3] J. Granacki et al., "The ADAM Advanced Design Automation System: Overview, Planner, and Natural Language Interface," *Proceedings of 22th DAC*, 1985.

[4] B. Mitchang, "Towards a Unified View of Design Data and Knowledge Representation," *Proceedings of 2nd International Conference on Expert Database Systems*, pp. 33–49, 1988.

[5] E. Sternheim, R. Singh, R. Madhavan, and Y. Trivedi, *Digital Design and Synthesis with Verilog HDL*. Kluwer Academic Publishers, 1994.

[6] S. R. Schach, *Classical and Object-oriented Software Engineering*. Mc Graw-Hill, 1999.

[7] R. J. Muller, *Database Design for Smarties*. Morgan Kaufmann, 1999.

[8] E. G. Mallach, *Decision Support and Data Warehouse Systems*. Mc Graw-Hill, 2000.

[9] M. Bushnell and S. W. Director, "VLSI CAD Tool Integration Using the ULYSSES Environment," *Proceedings of 23th DAC*, pp. 55–61, 1986.

[10] J. Daniell and S. W. Director, "An Object Oriented Approach to CAD Tool Control Within a Design Framework," *Proceedings of 26th DAC*, pp. 197–202, 1989.

[11] J. Daniell and S. W. Director, "An Object Oriented Approach to CAD Tool Control," *IEEE Transactions on Computer-Aided Design*, pp. 698–713, 1991.

[12] E. Siepmann, "A Data Management Interface as Part of the Framework of an Integrated VLSI-Design System," *Proceedings of ICCAD*, pp. 284–287, 1989.

[13] E. Siepmann and G. Zimmermann, "An Object-Oriented Datamodel for the VLSI Design System PLAYOUT," *Proceedings of 26th DAC*, pp. 814–817, 1989.

[14] J. Gray and A. Reuter, *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann, 1993.

[15] R. Ramakrishnan and J. Gehrke, *Database Management Systems*. Mc Graw-Hill, 2000.

[16] H. C. Jiau, K. F. Ssu, J.-M. Lin, and Y.-P. Ko, "Reusing CAD Tools in Object Oriented Based Framework," *Proceedings of 24th COMPSAC*, pp. xxx–xxx, Oct. 1999.

[17] R. J. Muller, *Productive Objects*. Morgan Kaufmann, 1998.

[18] *Standard VHDL Language Reference Manual*. IEEE, Mar. 1988.

[19] R. Camposano, L. F. Saunders, and R. M. Tabet, "High-level Synthesis from VHDL," *IEEE Design and Test of Computers*, Mar. 1991.

[20] D. E. Thomas and P. Moorby, *The Verilog Hardware Description Language*. Kluwer Academic Publishers, 1991.

[21] S. Note, F. Catthoor, G. Goossens, and H. De Man, "Computer Hardware Selection and Pipelining in High-performance Data-path Design," *Proceedings of International Conference on Computer Design*, Oct. 1990.

[22] J. S. Lis and D. D. Gajski, "Synthesis from VHD," *Proceedings of International Conference on Computer Design*, pp. 378–381, 1988.

[23] B. R. Preiss, *Data Structures and Algorithms*. Wiley, 1999.

[24] R. H. Katz, *Information Management for Engineering Design*. Spring-Verlag, 1985.

[25] A. Elmagarmid, ed., *Database Transaction Models for Advanced Applications*. Morgan Kaufmann, 1993.