

**EMBEDDING CLOCK BUFFER IP CORE
IN FPGA EMULATION**

By

JUN LI

**Bachelor of Science
Nanjing University
Nanjing, China
1991**

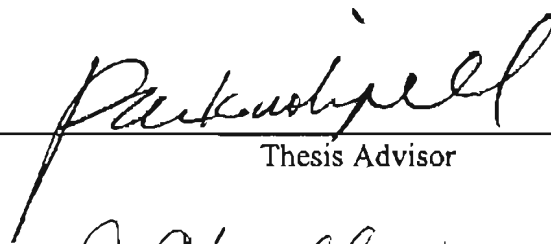
**Master of Science
Shanghai Institute of Biochemistry
Chinese Academy of Sciences
Shanghai, China
1994**

**Doctor of Philosophy
Oklahoma State University
Stillwater, Oklahoma
1999**

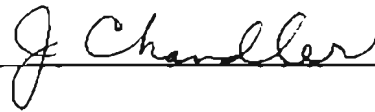
**Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
In partial fulfillment of
the requirement for
the Degree of
MASTER OF SCIENCE
December 2001**

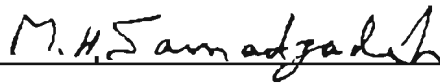
**EMBEDDING CLOCK BUFFER IP CORE
IN FPGA EMULATION**

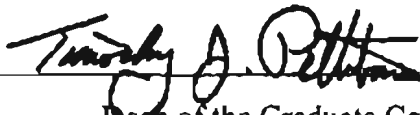
Thesis Approved:



Thesis Advisor







Dean of the Graduate College

ACKNOWLEDGMENTS

I'd like to express my sincere thanks to my major advisor, Dr. Nohpill Park. His careful reviewing and constructive advice are very important for the completion of the thesis. Dr. Park not only directed my research but also taught me to write engineering papers. I thank my committee members, Dr. John P. Chandler and Dr. Mansur H. Samadzadeh for their valuable instruction and helpful encouragement.

I shall appreciate Aptix Corporation and my colleagues for the good working atmosphere and helpful suggestions. Thanks go to my friends such as Min Cai and Xinyue Zhu, for their friendship help. In addition, they escorted me to hospital emergency room when I was preparing the proposal defense for this thesis.

Finally, I also want to thank my wife Xiahong Wang and my parents. Their encouragement keeps me away from laziness and depression. Their love makes me finish this thesis.

TABLE OF CONTENTS

Chapter	Page
I. Introduction	1
II. Literature Review	4
III. Preliminaries	10
IV. Netlist Object Model (NOM).....	15
V. NOM Controlled Embedding of Clock Buffer IP Core in FPGA Emulation	21
1. Design Analysis	21
2. Implementation	23
3. Verification and Emulation.....	29
4. Discussion	33
VI. Summary and Conclusions	35
References.....	37
Glossary	41

LIST OF TABLES

Table	Page
1 Emulation Efficiency -----	7
2 Output report of clock ports and their associated clock counts of design COUNTERBUFFERS -----	30
3 Resource utilization of design F1 in one FPGA -----	31
4 Resource utilization of design F2 in one FPGA -----	31

LIST OF FIGURES

Figure	Page
1 Typical ASIC Design Flow -----	5
2 Simplified FPGA Structure -----	8
3 Electronic Design Automation Object Model -----	11
4 MP-4 System Emulator -----	12
5 Reduced RC Model -----	14
6 NOM Architecture -----	16
7 A Simple Design Used To Illustrate The NOM Structure -----	18
8 The NOM Structure Of Example Design Fig. 5 -----	19
9 NOM Structure Of NO_BOUNCE Cell -----	20
10 Clock Types -----	22
11 Build Design Instance Tree -----	23
12 Build Clock Table -----	24
13 Integrated Clock Buffer BUFGP -----	26
14 Verification of insertion of clock buffer in test design by gatevision -----	32

Chapter I. Introduction

Application-specific integrated circuits (ASIC) are becoming more complicated, especially because of the advanced integrated circuit technologies that allow electronic system designers to put a whole system on a single ASIC chip, which previously required multiple integrated circuits such as digital logic, memory and mixed-signal digital/analog logics [6], so-called System-on-Chip (SoC). Since the size and volume of SoC are greatly reduced, SoC has the advantages of high performance and low power consumption. However, the high speed and high density of SoC challenge the ASIC verification methodology.

Traditionally, there are three types for ASIC verification, namely silicon prototyping, software simulation and breadboarding [15]. The drawback of silicon prototyping is that it is difficult to probe internal signal on chip for debugging. The long turn-around time makes silicon prototyping infeasible today. The system level simulation cannot allow enough time to ensure correct functionality because it takes even months to simulate a few seconds of real-time operation of SoC [29]. The ASIC breadboarding by using some low capacity programmable logic chips requires too long time to setup the board for large design. Hence the emerging of FPGA emulation technique is a key to resolve the SoC verification problem. This thesis will address a clock buffer insertion techniques in FPGA emulation.

Emulation speeds up verification dramatically by involving hardware system and field-programmable-gate-array (FPGA) to layout design blocks. "For the emulation to be effective, the emulated logic must be functionally equivalent to the logic running in

simulation and to the final chip" [23]. Each FPGA device running on an emulation board requires a wrapper file. The wrapper file contains two types of information. One type of information contains the necessary hardware-description-language (HDL) modules instantiated from the database. The second type of information contains any required interface logic, such as bi-directional drivers. If any devices other than FPGA are on the emulation board, HDL modules to define configuration of these devices must also be created. The connections between the FPGAs are defined in an emulation top-level file—a pure netlist with no additional logic [23]. The connections between FPGA and other device are defined in a pin map file [2]. Then, both the design files and the FPGA wrapper files are run through a standard synthesis process by the emulation software. This process targets on appropriate FPGA technology. The synthesis tools are provided by vendors. The typical synthesis time by using Synopsys' FPGA Compiler or FPGA Express is one to two hours per FPGA [24]. After synthesis, the emulation flow moves through several steps. The software works with the design's FPGA-specific netlist files, a top-level netlist defining the connections between the FPGA netlists, and a pinmap file describing the physical characteristics of the FPGAs and other devices in the setup. Beginning with a mapping process, the emulation software is used to check the input files for consistency, determine the clocking scheme, and fix the FPGA pin locations. Then constraint files for placing and routing the FPGAs are created. At this point, vendor-specific tools for the target FPGAs are utilized. If the vendor tools place all devices successfully, the final step is to produce a routing for the circuit board. These steps generate binary files for the FPGAs and the FPCB (Field Programmable Circuit Board). The binary files are then downloaded on the FPCB.

In each FPGA, there are I/O blocks and logic blocks. Each input and output signal needs corresponding buffers. For clock port, a clock buffer is required to enhance clock signal and reduce FPGA routing congestion, since the resource in each FPGA is limited. In the integrating process, verification engineer do not know the clock port and associated clock count in a custom design. Therefore, the clock port selection is based on information provided by design engineers and their experience and intuition instead of on the actual clock count. If the clock port selection is not consistent and results in different clock port selections each time, it will generate inconsistent emulation results. After the clock ports are selected, the clock buffer is integrated manually by the command, "insert pads" [30]. The manual integration of clock port is not efficient and error-prone. Therefore, in this thesis, we propose to use Netlist Object Module (NOM)-based control on the insertion of clock buffers into user design clock ports before a design is breadboarded onto an emulator.

Chapter II. Literature Review

ASIC application has been applied virtually to every type of chip that performs a dedicated task. Although its history is very short, ASIC captured 10% of the total semiconductor market by 1990 [20]. For some specific cases, ASIC is the only solution. For example, when small volume and power requirements are the major restrictions of using standard integrated circuits (IC) or any standard ICs cannot perform a unique function, ASIC becomes the only choice. ASIC can also increase system performance and throughput dramatically, and reduce system price if the volume of the production reaches break-even [20]. On the other hand, ASIC also has some disadvantages: designing and testing an ASIC are time-consuming. The ASIC verification phase of design consumes more than 60% of entire project [6], and design complexity results in an exponential rise in the verification time using traditional logic verification technique of simulation. SoC exacerbates this situation. The task of verifying the very complex SoC with simulation is very difficult if not impossible [6]. Therefore, the enhancement of logic verification of ASICs is required to facilitate and shorten designing and testing time. The ASIC design procedure is shown in Fig. 1.

Simulation means that an input pattern, when applied to an IC, operates upon and generates a special output pattern, which will be compared with the expected result [20, 18, 19]. Simulation can be performed at behavioral [21], structural, and hardware levels as well as any of their combinations. Logic simulation includes functional and timing simulation. Functional verification is the first necessary step after a design is completed. Functional simulation is used to verify the entire design behavior or functional

requirements. It is independent of timing delays. Functional simulations greatly simplify timing by assuming that all logic elements have a common delay of one time unit (unit delay) [28]. Since delays due to fanout loading and wiring capacitance are not comprehended, the simulations run faster, and the designer is able to obtain quick feedback of circuit functionality.

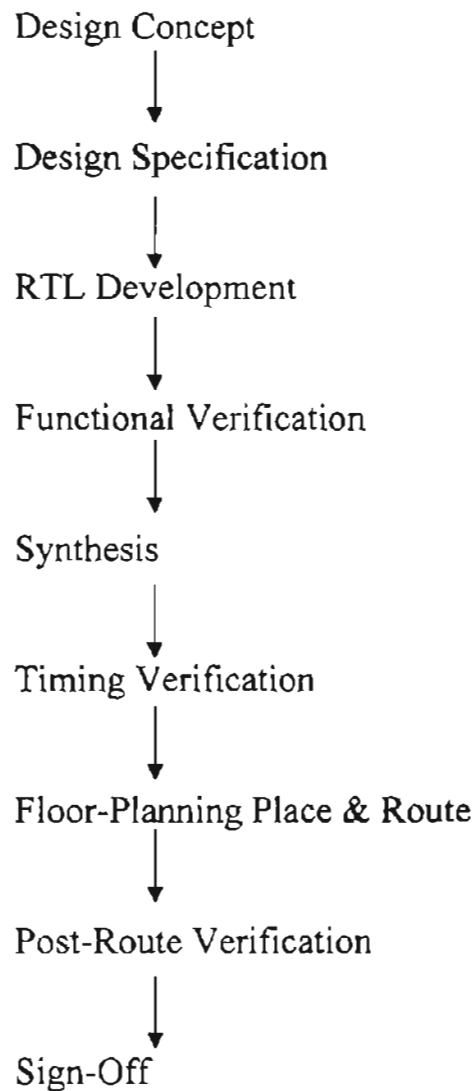


Fig 1. Typical ASIC Design Flow [16]

However, logic simulation has several shortcomings [20]. First, the quality of design verification is determined by the quality of the simulation vectors. The vectors must assure that the ASIC will operate properly in the system under all system conditions and modes of operation. Secondly, as designs have become more complex, the simulators simply run too slow to meaningfully verify complex designs at the detailed, gate-level of design description. Consequently, logic simulators have evolved to work on more abstract design descriptions. Finally, because breadboarding a complex ASIC design by simulation is not practical, the development of a comprehensive simulation places a tremendous burden on the designer. To resolve these problems, ASIC emulation was developed.

ASIC emulation has been becoming popular since late 1980s. Real world verifications can be done by using hardware to mimic custom logic design [26]. Companies that focus on ASIC emulation typically emulate the custom logic of an ASIC by mounting large numbers of FPGAs in a fixed array on printed circuit boards (PCB). However, generating such systems can require lengthy development periods because of the use of FPGA components. Fortunately, in the mid-1980s, reconfigurable prototyping technique was introduced by Aptix to overcome the difficulty-of-use, high cost and performance limitations of ASIC emulation, and to meet the increasing needs of system-on-chip design [3].

The ASIC emulator is a special type of hardware modeler [20]. It provides a means of automatically breadboarding a design on hardware. When using an ASIC emulator, the logic is partitioned and downloaded over an array of re-programmable logic devices. An automatic router, based on a matrix of switchable interconnects, completes

the connections between devices. The downloaded design is plugged into the target system board's ASIC socket and run. This arrangement offers a degree of a real time operation that would otherwise takes a lot of time to simulate. Based on the data provided by H. T. Verheyen., the emulation efficiency was listed in Table 1 [27].

A simplified FPGA structure is shown in Fig. 2. It contains many logic blocks, which are used to generate many ASICs. FFFA periphery is surrounded by many I/O blocks, which are used to provide channels to communicate to outside.

TABLE 1: Emulation efficiency [27]

Design Type	Application	Typical Design Size (Gates)	Typical FPGA utilization (Gates)	Typical #FPGAs per design	Typical Operating Frequency (MHz)
Central Processor	μ processor μ controller	1 million	2k	> 400	0. 1-2
Complex Datapath	ATM, LAN	250k	5k	30-40	5-15
Structured blocklevel	Multimedia	250k	10k	15-20	10-20
Pipelined SignalFlow	Telecom	150k	20k	5-10	>20

Table 1 shows that many mid-range ASICs with a design size between 100k to 300k gates can be emulated with less than 40 FPGAs: therefore, emulation allows significant speedup in development time, emulator configuration time, and incremental debugging time.

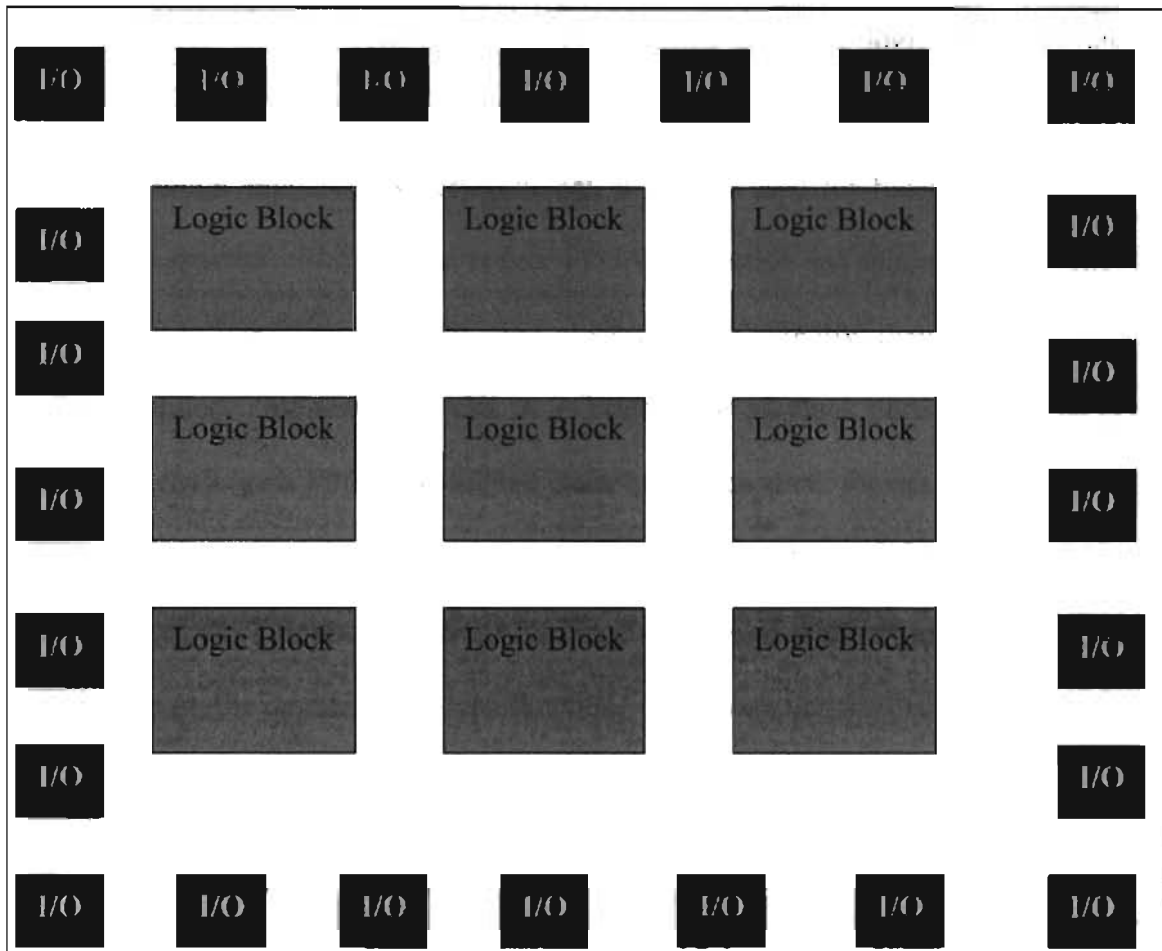


Fig. 2 Simplified FPGA structure

The FPGA emulator is named for equipment used to perform ASIC emulation. The emulator combines three basic techniques [2]: synthesis, high density FPGAs, and programmable interconnects. These technologies together enable the realization of rapid prototyping. Synthesis and compilation are used to quickly convert HDL to gate-level netlists, and C-codes to assembly codes for processors. FPGAs are then used to convert the gate-level netlists to a physical model of the netlist itself. The field programmable interconnects (FPICs) are used to realize a physical system that enables a combination of

FPGAs, micro-processors or micro-controllers, and discrete components such as random access memory (RAM), read-only memory (ROM), interface logic and even analog circuits, to be connected at a very high speed [2].

Although emulation is very efficient, it is only used for functional verification. To enhance external clock signals, reduce FPGA congestion and minimize clock skew, it is necessary to insert clock buffers after clock ports. Currently, there are two types of FPGA compiler. One always inserts BUFG buffer for all clock ports [25]. This is not efficient because each FPGA has limited clock buffer resource: for example, some Xilinx FPGA XC4000s only allows up to four clocks. The other type of FPGA compiler does nothing about clock ports and leaves it to users. If the user does not insert a clock buffer to a clock port, the emulation will malfunction. Therefore, verification engineers have to select the clock port manually and blindly, and insert clock buffers one by one. This process is time consuming and not accurate, and the emulation result may not be consistent because different clock ports are selected and different numbers of clock buffers are inserted. To overcome the shortcoming of current FPGA compilers, we'll use NOM to control the integration of clock buffers to proper positions on FPGA. With our program, we can automatically select clock ports, clock buffer type, the number of clock buffers and their integration.

Chapter III. Preliminaries

Electronic design automation (EDA) (Fig. 3) is a necessary tool for today's hardware design. Designs were generally written in hardware description language (HDL) that is more understandable. Then HDL codes are translated into netlists by software tools such as FC2 tools [25] and netlist-object model (NOM). The user-defined modules or predefined cells in HDL code are synthesized into gate-level [26] by vendors. The synthesized binary code can be downloaded on an emulator for logical verification.

In this work, we will use MP-4, an Aptix emulator. The structure of MP-4 is shown in Fig. 4. It is a fully functional hardware and software realization. The system emulator is realized as a very flexible architecture in which the user determines what FPGAs and what other components (RAM, ROM, and processor) to use. The emulator has built-in BUS, CLOCK and I/O resources to be utilized in the final design. MP-4 supports more than 3000 pins to be used for the design mapping, and supports more than 500 pins to be used for I/O. Realized frequencies vary from 15 MHz to 35 MHz. Gate-counts vary from 50k to 250k gates [2]. Aptix software tools provide automatic conversion of the computer-aided engineering (CAE)-domain description, which is generally in a hardware description language (Verilog, VHDL or EDIF) format, into physical domain.

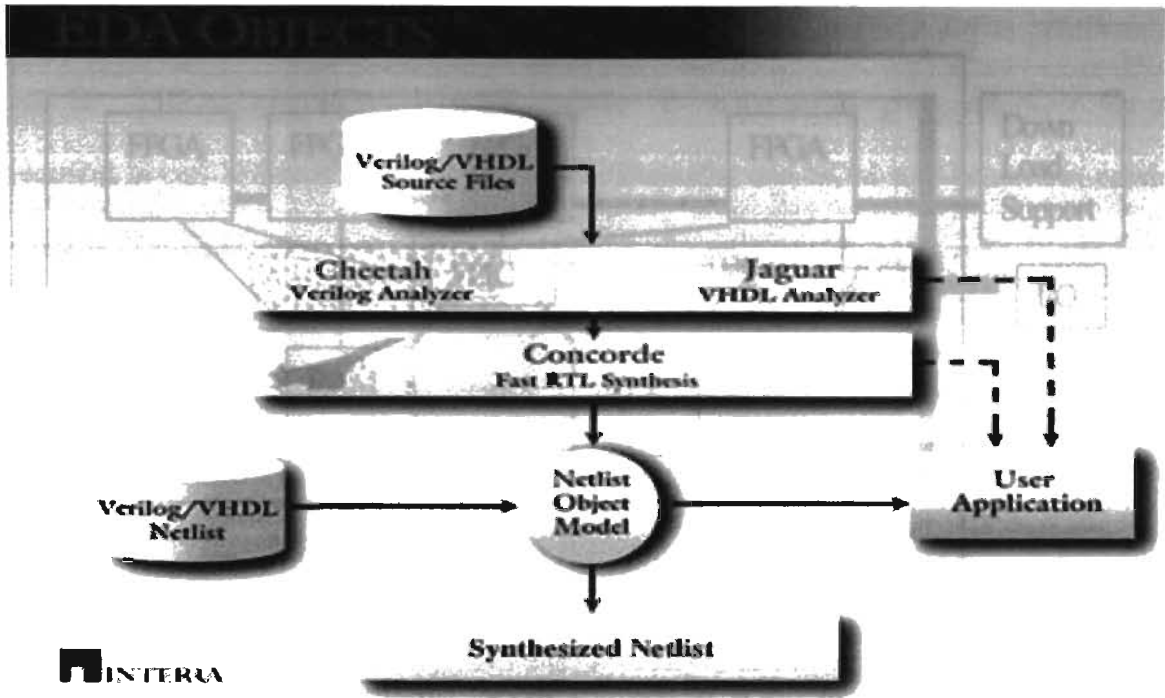


Fig. 3 Electronic Design Automation Object Model [10]

ASIC emulation sacrifices design timing accuracy because the reprogrammable logic devices have their own timing characteristic [20]. Its usefulness may therefore be limited to low-speed (less than 5MHz) or functional emulation and debugging only. Because of the timing inaccuracy and performance limitations, simulation and timing analysis are still required. In spite of its limitation, the use of emulation can be extremely valuable in resolving design errors, ambiguities, and specification miscommunications without having to wait for silicon fabrication and incurring the risk of iteration. The system also allows software incorporation to test their code as soon as the ASIC design can be downloaded. Therefore, it enables software-hardware co-design.

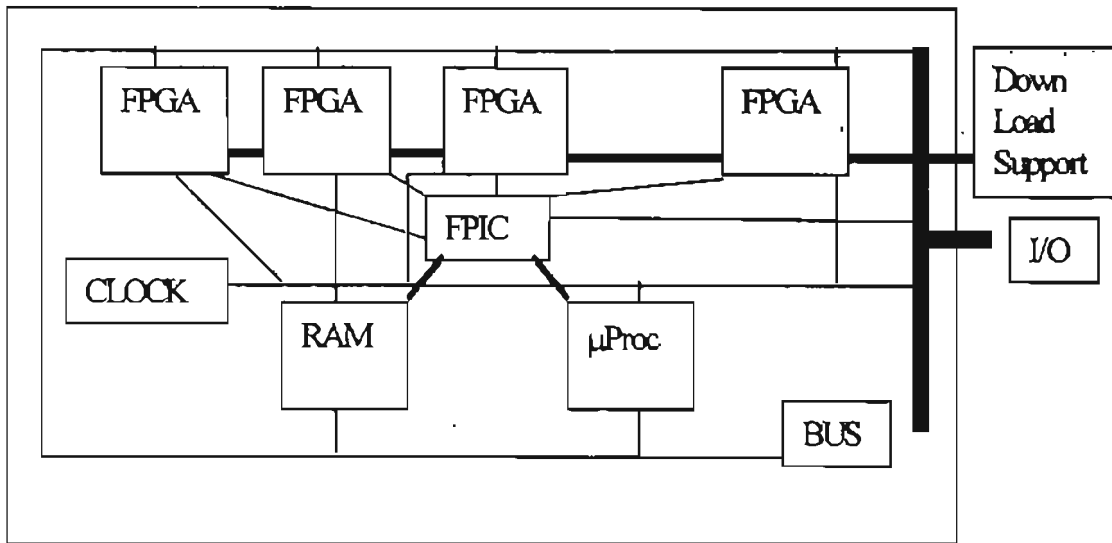


Fig. 4 MP-4 system emulator [2]

Netlist Object Module (NOM) is used in this work to integrate timing buffer into a design. NOM is a data structure and a language independent object model for multi-format netlist with application program interface (API) for representing connectivity information between design modules (cores). The connectivity information is stored in a hierarchical, language independent manner. The interface hides language specific details from application programs and provides a unified view of any netlist description through a common API. The API provides intuitive functions to access and modify the in-memory data structures. The NOM can be easily extended, allowing users to build new capabilities on top of the existing ones to meet their specific requirements from a netlist representation. NOM is used in a wide range of applications, including language translators, schematic generators, FPGA partitioning, and front-end to auto place-and-route tools.

The clock buffer IP core is a pre-defined cell by Xilinx [30]. It has interfaces I and O. "I" is an input port for inputting data. "O" is an output port. An EDIF file is required to call Xilinx libraries to generate buffer cells, and NOM will be used to create clock buffer instances.

Clock buffers are inserted into a clock net of custom design in FPGA emulation. The global primary clock buffer (BUFGP) can be used to reduce congestion in FPGA emulation. As clock pins have high fanout, if each clock pin of logic blocks has a separate net connected to the clock, some ASICs may not be placed and routed properly. In FPGA XC4000 [30], there are global buffer routing resources. Eight global nets run horizontally across the middle of the device. These nets can distribute signals to the configurable logic block (CLB), and reduce routing congestion, and thereby enable routing of some otherwise un-routable design.

Clock buffers can be used to enhance clock signals. An external clock signal may not reach CLB without a clock buffer, since the electric signal becomes weaker while being transported through a net.

A clock buffer reduces the clock skew by inserting some delay. "Clock skew is the difference in the arrival time of the clock pulse between a source and destination register (or other synchronous elements)" [13]. Global clock buffers are used for external clock signals. Some positive skew can be beneficial by decreasing the setup time. However, too much positive skew can create a race condition and negative skew will require long setup time. Clock buffers minimize clock skews by inserting delays through clock buffers to synchronize external clock signals.

Physically, clock buffers have one input and one output as interfaces. We notice that the input is clock signals, which are regular in pattern. The output of clock buffers is also regular with same pattern to that of the input, except that output has some delay.

It can be modeled by using reduced resistor-capacitor (RC) hardware model (Fig. 5) [16]. When an electric pulse arrives, the capacitors C are charged, and they will keep charged until the electric drops to 0 and then the capacitors are discharged, and generate an enhanced electric pulse output. Based on the value of capacitor C and resistor R, the value of the delay can be calculated. However, in FPGA device, it is impossible to determine the timing delay until the placement and routing of the design is completed [1]. As an I/O block in FPGA, clock buffer BUFGP cannot be separated from the FPGA. Since there is a great deal of flexibility and the routing resources are quite complex, Xilinx does not provide any simple timing models for FPGA family XC4000 [30]. The actual clock buffer model and its related parameters are unavailable. In this context, NOM is used to instantiate primary clock buffers and insert them to desired clock ports.

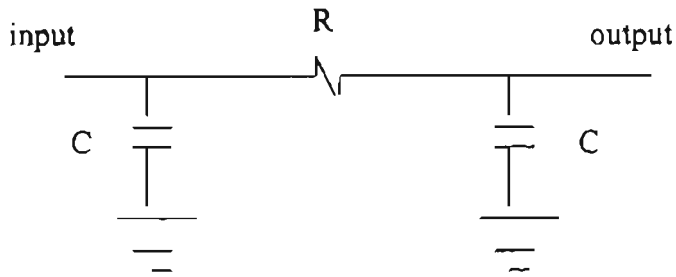


Fig. 5. Reduced RC model [16]

Chapter IV. Netlist Object Model (NOM)

NOM is composed of several hierarchical objects: root, library, cell, module, port, net, instance, terminals and assign (Fig 6).

Root is the base address of the whole data structure. The root may contain library lists, two kinds of module lists: top module list and module list. In each library, there are many cells. Each module is inherited from different cells. A library can be referenced from the NOM root. In a module, there is a net list, a port list, an instance list and an assign list. Net is communication pathway and it can be wired, wireless or other. Connected by a net, port is a transportation gate to control signal to or from a module. Ports control the data direction of input, output either uni- or bi-directional. The instance list lists up various instantiated modules. On each instance surface, there are terminals connected to outside net. An important concept is that the instances can be abstracted as modules, and then the terminals are converted to ports. This concept allows NOM netlist to be extendible.

NOM has the following advantages [9]:

- 1) It preserves electronic design hierarchy. Electronic components are designed hierarchically and the lower level in the hierarchy is generally designed first for modular design growth. The lower level modules with less complexity have a fewer number of gates and can be taken care by engineers. The upper level is built on top of the lower level designs.
- 2) NOM can read and write designs in different languages such as Verilog, VHDL and EDIF.

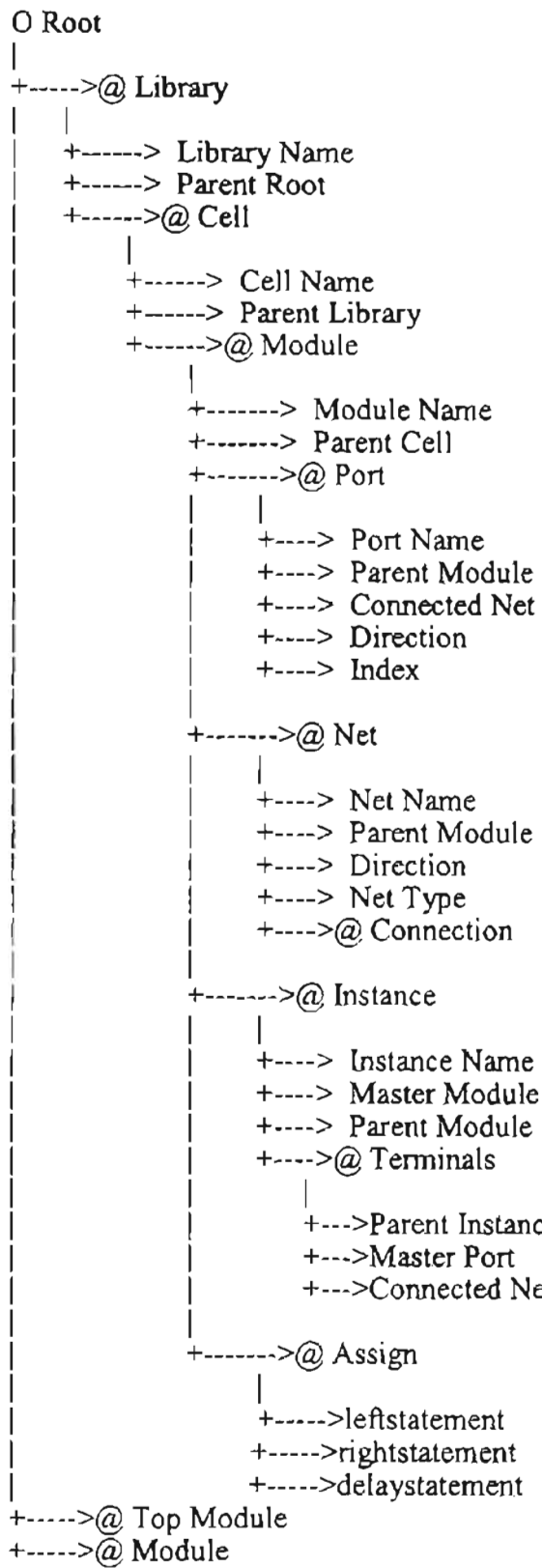


Fig. 6 NOM architecture [9] ('@' symbol represents a NOM list of the named entity)

- 3) Designs in different languages can be put together into NOM to build a language-independent data structure, and this structure can be encoded in its own format.
- 4) With a language-independent data structure, users can extend object model by adding, deleting or modifying the structure.

A simple design shown in Fig. 7 will be used to examine its NOM structure. In Fig. 7, the root is named nomRoot, the top module is FPGA_1 whose parent cell name is FPGA. Module FPGA_1 contains four instances: GND_1, LUT2_1, FD_1 and NB1, and their internal cells are GND, LUT2, FD and NO_BOUNCE, respectively. Suppose these modules are instantiated from library LIB, and the whole design will be populated on a dummy library 'WORK'. Fig. 7 can be represented by the following data structure in NOM (Fig. 8). The design of F1 contains the following lists:

Library List: WORK and LIB.

Cell List: GND, LUT2, FD, and NO_BOUNCE.

Module List: GND_1, LUT2_1, FD_1, and NB1.

Port List: I0, I1, I2, I3, O1, and O2.

Net list: N1, N2, N3, N4, N5, N6, N7 and N8.

Instance List: GND_1, LUT2_1, FD_1, and NB1.

Notice that NB1 contains one instance of FDE_1 inside. NO_BOUNCE can be defined in the same library as shown in Fig. 8 or in another library. The NOM structure of NB1 is shown in Fig. 9.

The cells GND, LUT2, FD and FDE are defined in library LIB and they are standard pre-defined libraries provided by ASIC vendors.

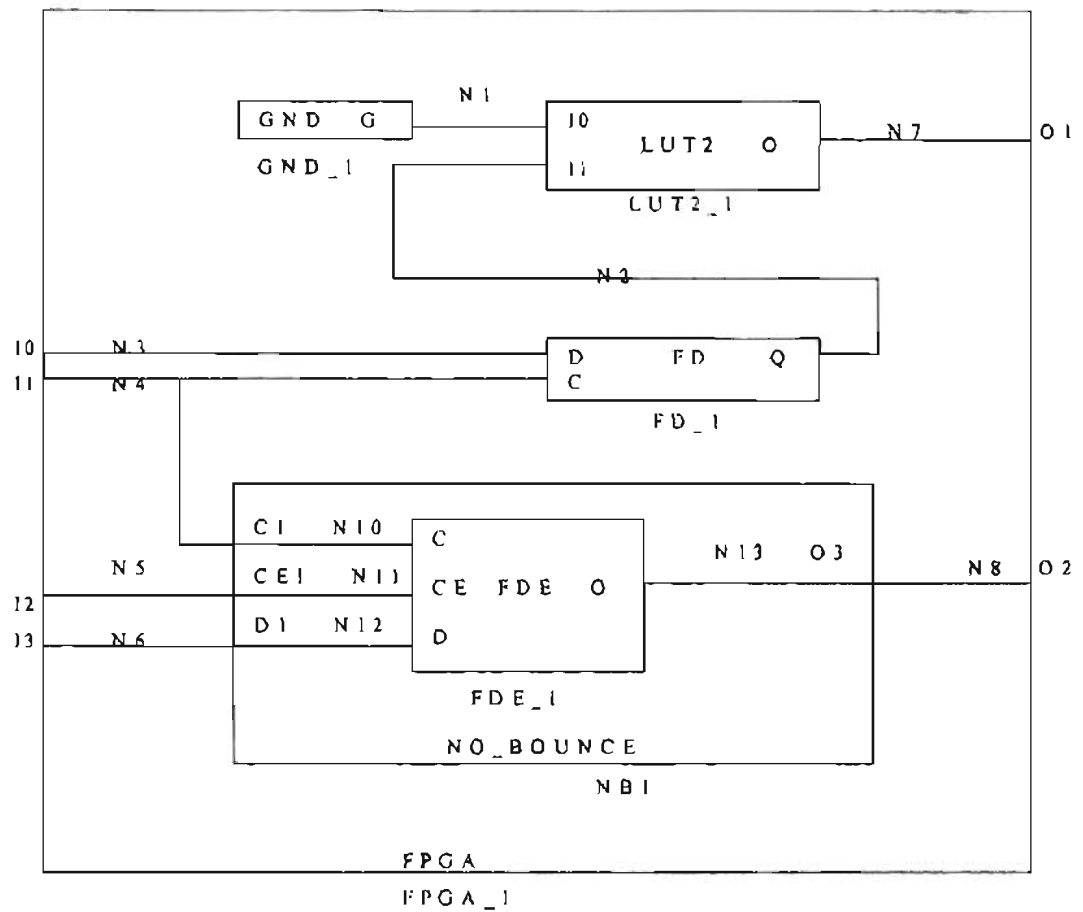


Fig. 7 A simple design used to illustrate the NOM structure

By definition, a clock net is a net that connects to C or G pins on the lowest module. C and G are just two special characters used in NOM. A port connected to a clock net is called a clock port. For example, in Fig. 7, net N10 is a clock net, because N10 is connected to C terminals of lowest module FDE. N4 is also a clock net that connects to C terminal of FD and to C1 of NO_BOUNCE, thus Port 11 is a clock port as well.

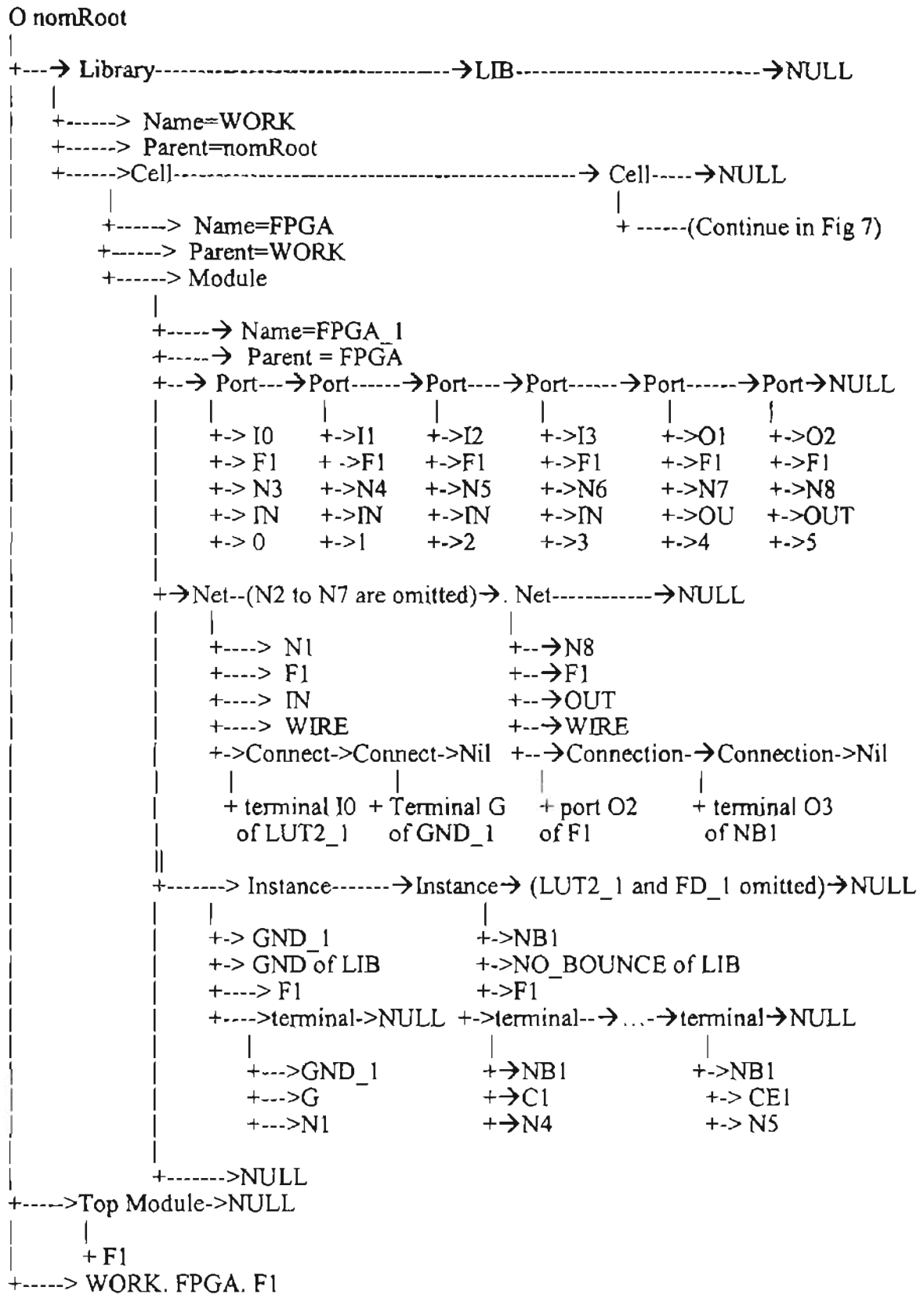


Fig. 8 The NOM structure of example design Fig. 5

(Continue from cell NO_BOUNCE in Fig 8)

```

|
+-----> Name = NO_BOUNCE
+-----> Parent = WORK
+-----> Module
|
+-----> Name = NB1
+-----> Parent = NO_BOUNCE
+-----> Port----->Port----->Port----->Port ----->NULL
|           |           |           |
+> C1      +> CE1      +> D1      +> O
+> NB1      +> NB1      +> NB1      +> NB1
+> N10      +> N11      +> N12      +> N13
+> IN       +> IN       +> IN       +> OUT
+> 0        +> 1        +> 2        +> 3
|
+-----> Net----->Net----->Net----->Net ----->NULL
|           |           |           |
+> N10      (omitted)  +> N12      (omitted)
+> NB1
+> IN
+> wired
+> connect->Connect->Nil +> connect->connect->NULL
|           |           |           |
+> C1      +> C        +> D1      +> D
of NB1     of FDE_1   of NB1     of FDE
|
+-----> Instance -----> NULL
|
+--> FDE_1
+--> FDE of LIB
+--> NO_BOUNCE
+--> terminal--->terminal-->terminal-->terminal->NULL
|           |           |           |
+>FDE_1    +>FDE_1    +>FDE_1    +>FDE_1
+>C        +>CE       +>D        +>O
+>N10      +>N11      +>N12      +>N13
+----->NULL

```

Fig. 9 NOM structure of NO_BOUNCE cell

Chapter V. NOM Controlled Embedding of Clock Buffer IP Core in FPGA Emulation

1. Design Analysis

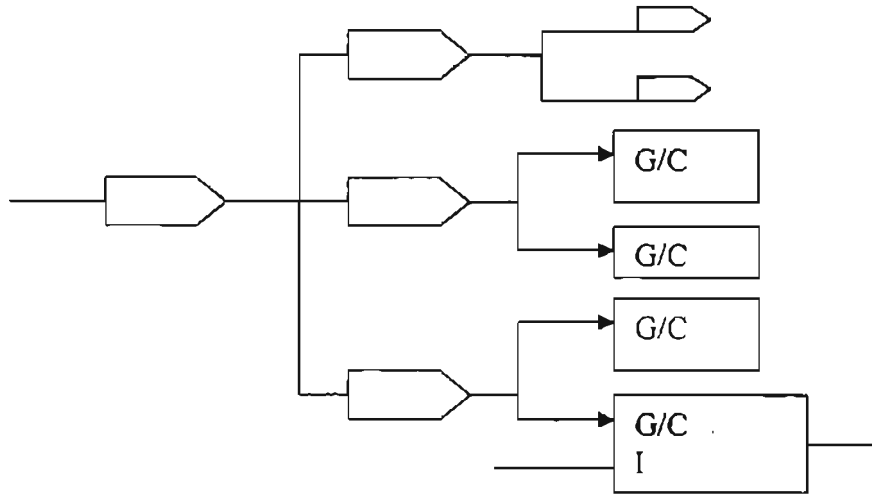
There are many EDA tools based on netlist object models. In this thesis, a commercial NOM tool developed by Interra Inc. [9] is used.

In a design, the number of clocks may vary. However, FPGA on emulator has only a limited number of clock buffers provided. If a design has more clocks than allowed, which port should be selected to insert a clock buffer is based on their connections. The complexity of connections is measured by clock count.

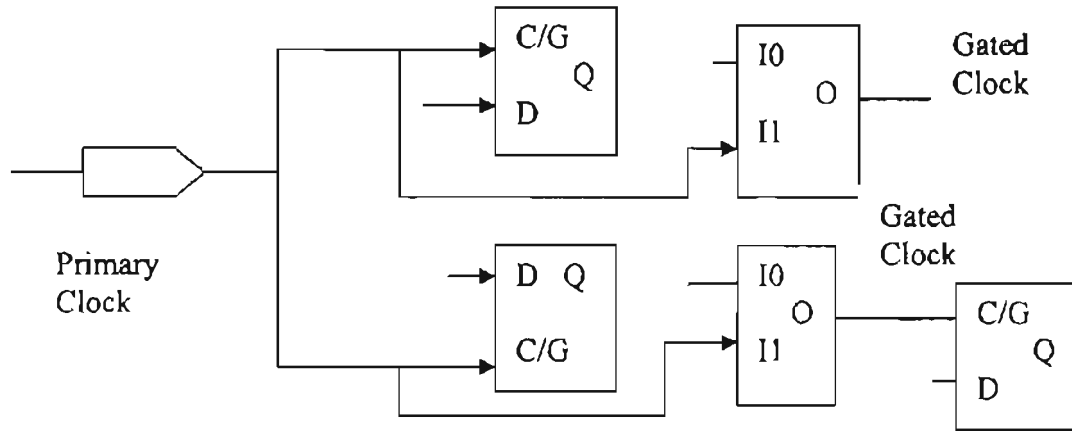
In order to get the right clock ports, the clock net must be obtained first. By definition, clock net is a net connected to C or G pins in the primitive module. The clock net connected port is called clock port. When the clock nets are found, they are traced up to the top-level port. If it reaches the top-level port (e.g. in Fig 7, they are I0, I1, I2, I3, O1, O2), the clock count associated with that port is increased by 1. Otherwise, a warning or error message is issued to indicate that the clock is a gated clock (or internal clock). For example, the clock count of clock port I1 is 2 in Fig. 7.

There are four possible types of clocks as shown in Fig. 10. Some are permitted, and some are forbidden. One common rule is that to distribute clock signals, only one path is allowed from the reference clock to storage elements. The storage elements can be buffers, inverters, etc.

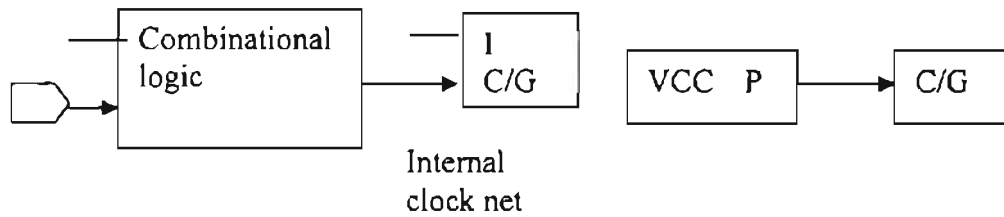
Case A: Distributed Clock Structure [5]



Case B: Gated Clock [5]



Case C: Internal clock



Case D: Multiple driver clock

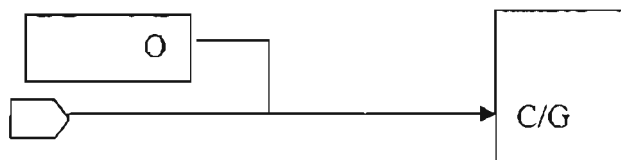


Fig. 10 Clock types

In addition, gated clocks are permitted only if they do not harm other element signals [16]. That is to say, the internal clock net is allowed conditionally, but warning messages must be issued. For example, a divided clock is a kind of allowed internal clock [16]. Multiple drive clock nets are not allowed. In Fig 7, empty arrows represent input clock ports. Empty squares represent instances. Letters I, O, C, G, D, Q represent terminals.

2. Implementation

Based on the analysis, a flowchart to get a correct clock port is shown in Fig 11 and Fig. 12.

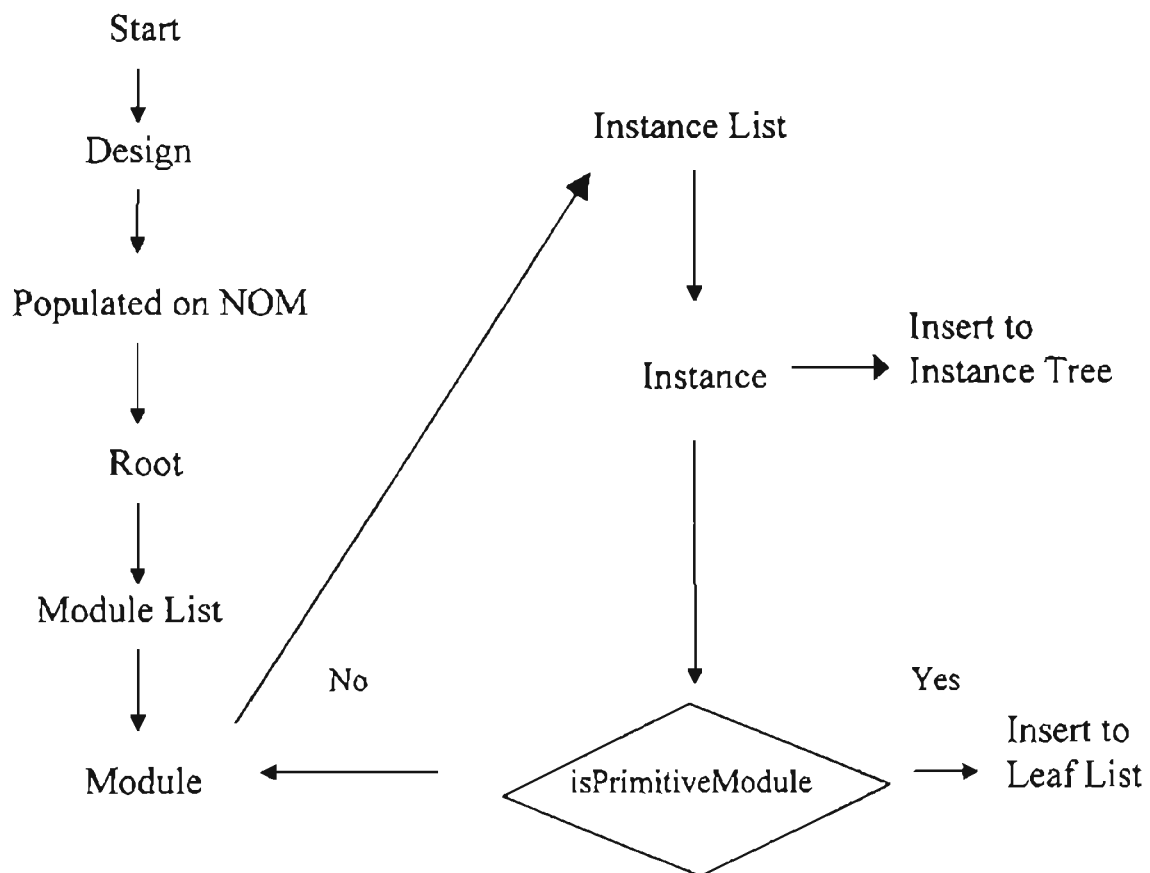


Fig. 11. Build a design instance tree

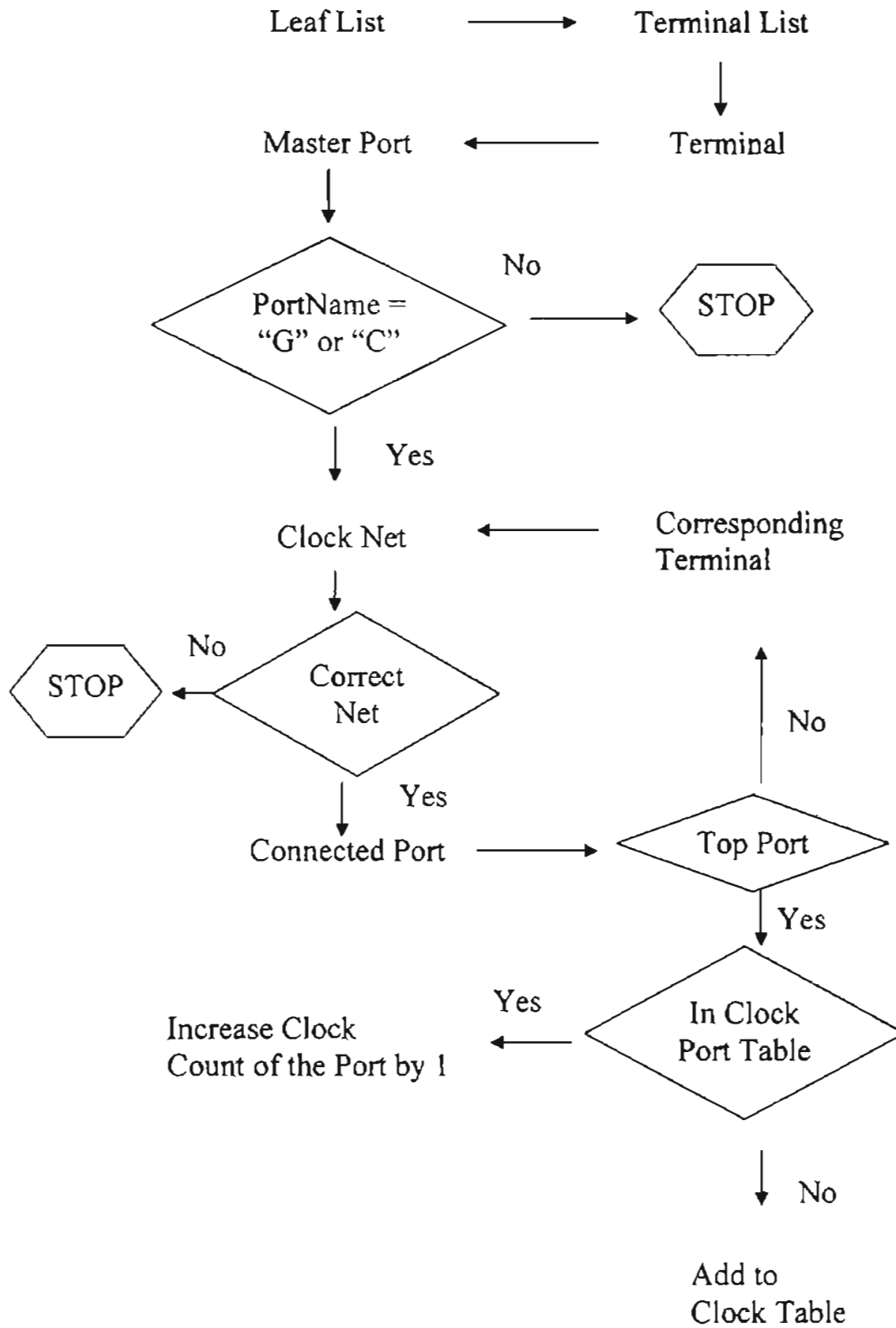


Fig. 12 Build clock table

Once the addresses of clock ports and clock counts are obtained, the clock port with the highest clock count will be selected. An EDIF file containing declaration of a clock buffer cell is input to the NOM, and the corresponding instance is created and inserted into the net after the clock port (Fig. 13). Note that the controlled insertion of clock buffers will avoid random insertion and the number of clock buffers to be inserted depends on the FPGA type.

In this work, the clock buffer IP core provided by Xilinx [31] is instantiated and inserted to the clock net. It is called BUFGP. BUFGP is a primary global clock buffer IP.

Three types of structures are defined in this work. Struct `instTree` is used to recode instance structures of hardware design. Struct `leafList` is used to recode all the primitive cell addresses in a design. Struct `clockTable` is used to recode all clock port address and their clock counts. Their definitions are:

```
typedef struct instTree {
    nomNode node;
    struct instTree *parent;
    struct instTree *next;
    struct instTree *child;
} InstTree;

typedef struct leafLinkedList {
    InstTree *node;
    struct leafLinkedList *next;
} LeafList;
```

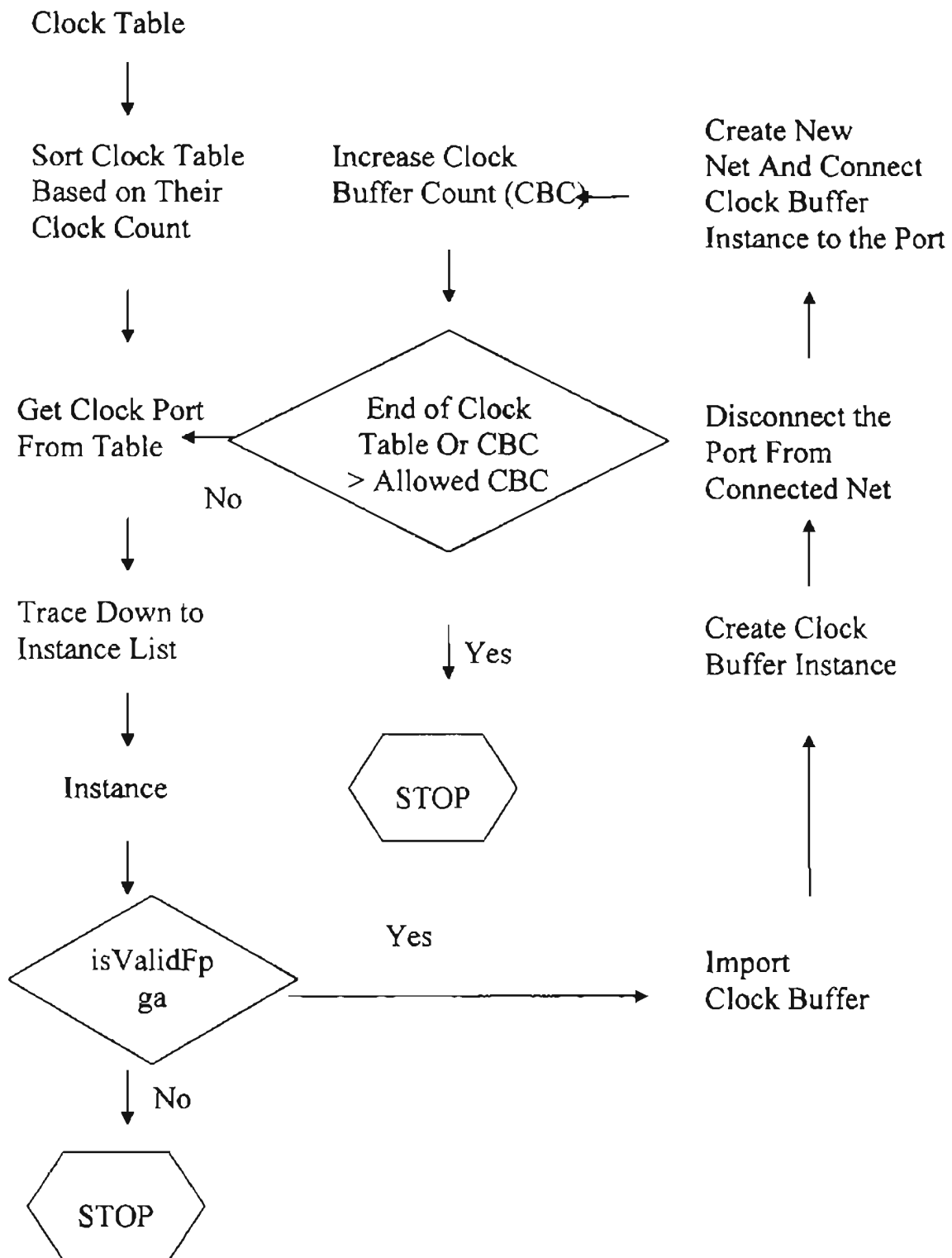


Fig. 13 Instantiate a clock buffer and integrate the clock buffer beyond FPGA boundary


```

typedef struct clockTable {
    nomNode node;

    int count;

    struct clockTable *next;
} ClockTable;

```

Using these data structures, an object ClockPad is implemented. It has nine data and eight public operations. The detailed declaration of object ClockPad is listed as follows.

```

class ClockPad : public DealWithIONClock
{
private:
    InstTree *TREE;

    LeafList *leafList;

    ClockTable *CLKT;

    nomNode root;

    nomNode module; //top module

    int allowClockNum;

    char *outputFileName;

    FILE *fout;

    char *top;

    static int allowClockCount;

```

```

private:

    int buildInstTree ();

    void buildRoot();

    void buildLeafListRoot ();

    void buildRootOfCLKT ();

    int buildTop (nomNode inst);

    int insertLeafList (InstTree *node);

    int buildTree (nomNode inst, InstTree *pNode);

    void getLowestMod();

    void traceUp (nomNode inst, nomNode clockNet, InstTree* temp, nomNode
*topPort );

    int testForMultiDriveClockNet (nomIter list);

    nomNode findCorrespondingPort (nomIter list);

    void updateClockTable (nomNode topPort);

    void printRoutine (nomNode port, nomNode inst, nomNode clockNet, InstTree
*tree);

    void printMiddlePartOfHierInstTree (InstTree *tree);

    void sortingTable();

    int insertBUFGP();

    int insertingBUFGP(ClockTable *CLKT, nomNode module);

    void releaseSpace(ClockTable *tbl);

    int setUpNewClockTable (nomNode port, ClockTable *Tbl);

```

```

protected:

    void freeCLKTSpace();

    void freeInstTree();

    void freeLeafList();

public:

    ClockPad(nomNode root, int totalClockNum, char *outputFileName, char
*topModName);

    ~ClockPad();

    int run_ClockPad();

    void printInstTree ();

    void printLeafList ();

    void printTable ();

    LeafList *getLeafList();

};

```

3. Verification and Emulation

After the programs are implemented, a test design is input into this program. The test design is designed to count numbers from 1 increasing by 1. Thirty-two bits are used to store the number. If the number reaches top and another one is added, overflow will occur. The test design is partitioned into two FPGAs, and its top module was called COUNTERBUFFERS. The output of this program is the design with clock buffers

inserted, which was ready for emulation test written in EDIF. The output by the program is shown on Table 2.

Table 2: The output report of the clock ports and their associated clock counts of the design COUNTERBUFFERS

Clock Port Name	Clock Count
SysClk	3350
Tclk	1108

Two commercial tools are used to check the output EDIF file. One is called gatevision [7]. It displays the logic design graphics by parsing the EDIF file. The second tool is called xflow [31]. It checks syntax errors in EDIF files. Therefore, xflow is used to test whether some syntax error was introduced when a clock buffer was inserted. In addition, xflow packed the design into FPGA.

The schematic shown in Fig. 14 is obtained from gatevision. The figure on the top in Fig. 14 is the whole design view. The figure on bottom shows the detail of clock ports and indicates that clock ports have been selected correctly. Two clock ports sysClk and tclk are selected which is consistent to data in Table 3. From the figure, it is shown that under top module COUNTERBUFFERS, there are two FPGA modules, F1 and F2. Within each FPGA boundary, two clock buffers are instantiated (the instance names are BUF_{GP}_sysClk and BUF_{GP}_tclk). This indicates that clock ports are selected correctly and clock buffers are inserted as expected.

Results from the gatevision indicates that BUF_{GP}s have been embedded into the design. However, it does not necessarily mean that the corresponding nets and instances

have been connected and function properly. So, the output EDIF files are passed through Xilinx [31] placement & routing (P&R) tool named xflow. This tool translates EDIF file contents into FPGA components, and maps different instances onto corresponding intellectual properties (IPs) located in FPGA. The design without BUFGP gives the following error messages:

“ERROR 98: Pack: 198- NCD was not produced. All logic was removed from design. This is usually due to having no input PAD connections in the design

This error message means that xflow fails to P & R the design if the program fails to insert BUFGP into the clock port and leave I/O pad empty.

On the other hand, the output EDIF from the test design is passed through xflow successfully. The log files from xflow are shown in Table 3 and Table 4.

Table 3 Resource utilization of design F1 in one FPGA :

Number of External GCLKIOBs	2 out of 4	50%
Number of External IOBs	121 out of 404	29%
Number of SLICES	582 out of 12288	4%
Number of GCLKs	2 out of 4	50%
Number of TBUFs	64 out of 12544	1%

Table 4 Resource utilization of design F2 in one FPGA

Number of External GCLKIOBs	2 out of 4	50%
Number of External IOBs	105 out of 404	29%
Number of SLICES	1395 out of 12288	4%
Number of GCLKs	2 out of 4	50%
Number of TBUFs	64 out of 12544	1%

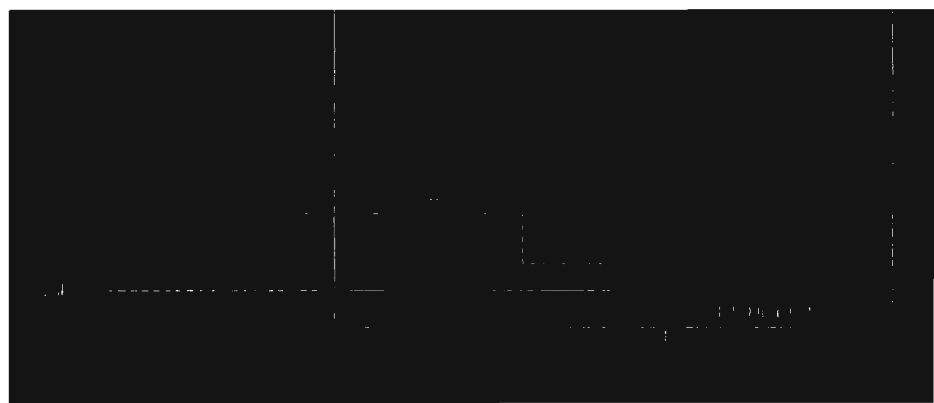


Fig. 14 The verification of insertion of clock buffers in the test design by gatevision

The logic design is placed and routed into the FPGA successfully. The time of placer completion of FPGA F1 was 16 minutes 14 seconds. The time of placer completion of FPGA F2 was 18 minutes 53 seconds. According the resource utilization tables (Table 4 and Table 5), number of GCLKs is 2 out of 4. This means that in each FPGA XC4000, there are four global clock buffers, and two global clock buffers are used respectively. These results were consistent with the report in Table 2 that shows two clock ports (SysClk and Tclk) are selected and one clock buffers is embedded after each in one FPGA, that is to say, two clock buffers in each FPGA.

The buffer-counter design with clock buffer inserted was compiled and downloaded onto Aptix MP-4 emulator as in Fig 4 and tested. The output is connected to a hardware equipment named logic analyzer. The logic analyzer shows that the number increased continuously by one from 1 to $2^{32} - 1$ and then overflow to -2^{32} . Then the display was a loop from -2^{32} to $2^{32} - 1$. The result was the same as expected.

Therefore, the program has selected the clock port correctly and the clock buffer IP cores were embedded into design correctly.

4. Discussion

Clock buffer is an important global resource IP core in FPGA emulation. It is used to reduce FPGA congestion, enhance clock signal, and minimize clock skew. Each clock buffer connected net in FPGA must be connected to a global net on the emulator, and a common top clock net connected to different FPGAs must share the same global buses on the emulator. Since a large design is generally partitioned into multiple FPGAs,

independent clock buffer selections among the multiple FPGAs will cause problems in breadboarding a design on an emulator, such that total required global net buses may exceed the number of global nets on the emulator, and that the global nets selected on one FPGA with a clock buffer are not selected on other FPGA involved in the same design may cause bus errors, and clock skews. The proposed method resolves this problem by controlling the selection of clock ports and the insertion of clock buffers globally across the multiple FPGAs under an emulation. Then the total number of required global net buses will never exceed the number of global nets on the emulator, and keep the clock ports on different FPGAs connected to the same top clock nets to have the same configuration.

This method selects the clock ports based on their fanouts, and checks the clock connections, which will enhance the efficiency and accuracy of the emulation process by providing automated selection of clock ports and insertion of clock buffers. In the process of the selection and insertion, the design logic is kept intact.

Chapter VI. Summary and Conclusions

Simulation and emulation have been two common methods of verifying microelectronics design. As today's designs are becoming more and more complex with higher density, traditional simulation techniques cannot verify the whole design to meet the time-to-market. Hence, emulation technique is emerging in today's design verification. Field-programmable-gate-array (FPGA) technology is one of the keys to realize the emulation. Since today's design is becoming larger and larger while the capacity of FPGA is limited, it is necessary to partition a design across a number of FPGAs. However, the capacity of emulation is limited because the number of FPGA on an emulator is limited. Therefore, the design should manage to make full use of the FPGA resources. Clock buffer is one of the important IP cores in FPGA emulation and each FPGA has a certain number of global clock buffers. If they are not used properly, external clock signals may not be able to reach their destinations properly and the high clock fanout may cause problems during placement & routing, which will reduce the FPGA capacity due to congestion. Also, since a whole design is partitioned and breadboarded into several FPGAs, the embedding process of the global clock buffers must be consistent across the whole design on an emulator, otherwise it may cause clock skews and emulation malfunctioning.

Finding right clock ports and inserting clock buffers to them are critical issues in FPGAs emulation. This work establishes an electronic-design-automation (EDA) solution. Netlist Object Model (NOM) can be used to control the insertion of clock buffer cores. NOM is a language-independent data structure. After parsing a design into NOM, an

instance tree is built to trace instances, their parents, children, siblings and their connection nets. Those nets connected to clock nets are traced back hierarchally along the instance tree all the way up to the top level. If they reach a top port, the port is identified as a clock port, otherwise as gated-clock. By using this method, the connection complex of clock ports is obtained (quantified by clock count). Then, based on different FPGA types, a certain number of clock ports with the highest top clock counts are selected and global clock buffers are instantiated and inserted to them. A design, which is composed of 32-bit counters and buffers, has been used to evaluate this work in terms of speed and accuracy. After clock buffers are embedded in the test design, the gate-level netlist has been parsed by gatevision, the results show that clock ports have been selected correctly, and clock buffers have been inserted properly (Table 2, Fig. 14). Then the gate-level design is parsed by xflow [31]. The result shows that the design has been placed and routed successfully on the FPGAs (Table 3 and Table 4). Finally, the design is downloaded onto an emulator, and the logic analyzer has verified the correct design and implementation by using the proposed method.

The problem of selecting clock port and inserting global clock buffer has been resolved successfully in this thesis. According to different clock types in Fig. 10, gate-clock transformation is an issue for future work on the base of this work.

References

1. "FPGA Data book and Design Guide". *Actel Corp.* California. 2000.
2. "MP4 System Explorer User's Manual". *Aptix Corp.* California. 1998.
3. "Reconfigurable Prototyping offers Increased Speed, Flexibility and Debugging".
Aptix Corp. Willess System Design/Systemchips Supplement. Aptix Corp. August 1998.
4. Benini L. , De Micheli G. , Macii A. , Macii E. , Poncino M. and Scarsi R., "Glitch Power Minimization by Gate Freezing". *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition.* pp163-167. IEEE Computer Society. 1999.
5. Bobba S. and Hajj I. N., "Maximum Current Estimation in Programmable Logic Arrays". *Proceedings of the Great Lakes Symposium on VLSI '98.* pp301-306. IEEE Computer Society. 1998.
6. Chang C. M., "SOC becomes Household Name". *Electronic News.* 46(43): 64. 2000.
7. Concept Engineering Corp. Connecticut. 2001.
8. Gonzalez-Torres J. , Mateos P. A. , Hernandez, J. M., "Full custom chip set for high speed serial communications up to 2. 48 Gbit/s". *Proceedings of the 1997 European Design and Test Conference (ED&TC '97).* pp614. 1997. IEEE Computer Society.
9. 'NOM API Programmers' Reference, Version 2. 6". *Interra Inc.* San Jose, California. 1995.

10. "http://www.interraeda.com". *Interra, Inc.* Access date: 02-27-01. Create Date: 2000.
11. Kaplan G., "Programmable logic devices developments in industrial electronics during 1992". *SPEC 93*. pp58-60. 1993.
12. Khare J. , Heineken H. T. and d'Abreu M., "Cost Trade-Offs in System On Chip Designs". *Proceedings of the 13th International Conference on VLSI Design*. pp178-184. IEEE Computer Society. 2000.
13. Knipping U., "3. 1i Constraints: Understanding Timing and Placement Constrains". Personal Communication. 1999.
14. Leeser M. , Meleis W. M. , Vai M. M. , Chiricescu S. , Xu W. and Zavracky P., "Rothko: A Three Dimensional FPGA". *IEEE Design & Test of Computers*, 15(1), pp 16-23. 1998.
15. Lo W, Choy C, and Chan C., "Hardware Emulation Board Based on FPGAs And Programmable Interconnections". *Proceedings of 15th International Workshop on Rapid System Prototyping*. pp126-130. IEEE Computer Society. 1994.
16. Nekoogar F., "Timing Verification of Application-Specific Integrated Circuits". Upper Saddle River, NJ: Prentice Hall PTR. 1999.
17. Pomeranz I. and Reddy S. M., "On Finding Functionally Identical and Functionally Opposite Lines in Combinational Logic Circuits". *Proceedings of the Ninth International Conference on VLSI Design: VLSI in Mobile Communication (VLSI '96)*. pp254-259. IEEE Computer Society. 1996.
18. Savino-Vázquez N. and Puigjaner R., "A UML-Based Method to Specify the Structural Component of Simulation-Based Queuing Network Performance Models".

- Proceedings of the Thirty-Second Annual Simulation Symposium*. pp71-78. IEEE Computer Society. 1999.
19. Scholl C. , Drechsler R. , and Becker B., “Functional Simulation using Binary Decision Diagrams”. *Proceedings of the 1997 International Conference on Computer-Aided Design (ICCAD '97)*. pp8-12. IEEE Computer Society. 1997.
 20. Schroeter J. and Cliffs. E., “Surviving the ASIC experience”. New Jersey: Prentice Hall. 1992
 21. Shoji M. , Hirose F. , Shimogori S. , Kowatari S. , and Nagai H., . “Acceleration of Behavioral Simulation on Simulation Specific Machines”. *Proceedings Of The 1997 European Design And Test Conference (ED&TC '97)*. pp373-377. IEEE Computer Society. 1997.
 22. Shriver E. Hall, D. Nassif, N. , Rahman, N. , Rethman, N. , Watt, G. and Farrell, J., “Timing verification of the 21264: A 600MHz Full-custom microprocessor”. *Proceedings Of The International Conference On Computer Design*. pp96-103. IEEE Computer Society. 1998.
 23. Singletary A., “Emulation technology for ASIC core verification: emulation strategies vary across three ASIC core”. *Integrated System Design (www.isdmag.com)*, July, 2000. www.us.design-reuse.com/NEWS/papers.html. Access date: 2-28-01. Create date: 07-01-00.
 24. . “FPGA Compiler II User Manual”. *Synopsys Inc.*, San Jose, California. 2000.
 25. “Synthesizing your design”. *Xilinx/Synopsys interface guide*. *Synopsys Inc.*, California. 2000.

26. Tessier R., "Incremental compilation for logic emulation". *Proceedings Of The Tenth IEEE International Workshop On Rapid System Prototyping*. pp236-241. IEEE Computer Society. 1999.
27. Verheyen H. T., "Accelerating Hardware/software co-design through system emulation". *Proceedings of 1996 High-performance System Design Conference*: pp1-18. IEEE Computer Society. 1996.
28. Vuletic M. , Davidovic G. and Muilutinovic V., "Suboptimal Detection of Telemetry Signals: Functional Simulation and VLSI Implementation". *Proceedings of the Sixth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*. pp289-294. IEEE Computer Society. 1998.
29. Walters S., "Reprogrammable Hardware Emulation Automates System-Level ASIC Validation". *WESCON/90 Conference Record*. pp1-5. Nov. 1990.
30. "Libraries Guider", *Xilinx Corp*. California. 2001
31. *Xilinx Corp*. California. 2001

Glossary

ASIC: Application Specific Integrated Circuits.

CLB: Configurable Logic Block.

EDA: Electronic Design Automation.

EDIF: electronic data interchange format.

FPGA: Field Programmable Gate Array.

FPIC: Field Programmable Interconnect.

FPCB: Field Programmable Circuit Board.

FPL: Field Programmable Logic.

HDL: Hardware Descriptive Language.

IC: Integrated Circuits.

IP: Intellectual Property.

NOM: Netlist Object Module.

PCB: Printed Circuit Board.

PLA: Programmable Logic Array.

PLD: Programmable Logic Device.

RAM: Random Access Memory.

ROM: Read Only Memory.

RTL: Register Transfer Level.

SoC: System on Chip.

VITA

Jun Li

Candidate for the Degree of
Master of Science

Thesis: EMBEDDING CLOCK BUFFER IP CORE IN FPGA EMULATION

Major Field: Computer Science

Biographical:

Personal Data: Born in Binhai, Jiangsu Province, China, on August 4, 1968.

Education: Received Bachelor of Science in Biochemistry from Nanjing University, China in 1991. Graduate and received Master of Science in Molecular Biology from Shanghai Institute of Biochemistry, Academia Sinica, Shanghai, China in 1994. Received Doctor of Philosophy in Veterinary Biomedical Science from Oklahoma State University, Oklahoma, USA in 1999. Completed the requirements for the Master of Science with a major in Computer Science at Oklahoma State University in October, 2001.

Experience: Worked as a researcher in National Center for Gene Research, and Shanghai Institute of Plant Physiology, Academia Sinica from July, 1994 to July, 1996. Then worked as research associate in Oklahoma State University until May, 2000. From then on, employed by Aptix Corp.