C-based Synthesis Experiences with a Behavior Synthesizer, "Cyber"

Kazutoshi WAKABAYASHI
C&C Media Research Laboratories, NEC Corp.
4-1-1 Miyazaki Miyamae-ku Kawasaki 216-8555, Japan
wakaba@ccm.cl.nec.co.jp

Abstract

This paper presents C-based behavioral hardware design environment. Initially, we discuss motivations of the design environment and the merits and demerits of the C language as a behavioral hardware language. The configuration of the environment of which main component is a behavioral synthesizer "Cyber", is presented. Then, we explain various aspects of the C-based design method using some design experiences for both control dominated circuits and data intensive circuits. Lastly, we summarize current status of our C-based design and discuss problems to diffuse C-based design method widely.

1 Motivation

1.1 To raise level of abstraction

The importance of top-down verification has been emphasized. Hardware algorithm should be verified at behavioral level first, then cycle behavior should be verified at RT level. However, not so many designers describe behavioral description, but they start with "synthesizable" RT description. The main reason might be that they don't like to describe two different descriptions at behavior level and RT level, since describing their design in a formal language correctly(syntactically and semantically) itself, is time consuming task. By any way, description for simulation and synthesis must be the same. Consequently, a practical behavior synthesis system which allows designers to describe their design just at "synthesizable" behavior level as a final description, is most important to realize C-based algorithmic hardware design environment.

1.2 Why C-based synthesis?

Our behavior synthesis system, "**Cyber**" accepts behavior descriptions in either C or behavioral VHDL. We are promoting C-based design more than VHDL-based, since behavioral VHDL is not powerful enough to describe control dominated circuits of complex timing. In addition, C simulation is faster than VHDL simulation, and C model is convenient for HW/SW Co-simulation, also C development environment is of good quality, inexpensive and widely diffused.

Extensions and subset: However, C language requires additional semantics to describe hardware accurately. First, we extend C to support "bit-length" and "in/out port" declarations as inevitable nature for hardware. Secondly, we include "synchronization", "clocking(cycle boundary)", "concurrency" for control dominated area. Lastly, various data transfer types, such as register, terminal, latch, tri-state transfer, are introduced for lower level description of control dominated circuit. The second and third extensions are NOT necessary for many circuits. These extensions aims to make the language cover both behavioral and RT levels seamlessly, which enables designer to tune-up the C code as in detail as he like.

Also, to synthesize high quality hardware, pointers for dynamic data structure, dynamic allocations and recursions are deleted from "synthesizable" subset. Pointers which are located on linear memory space are very harmful to speed up the synthesized circuit. Run-time dynamic features are not realistic for hardware synthesis since all necessary components have to be fixed at synthesizing (compilation) time. We call the subset of C with above extensions as "Behavior Description Language (**BDL**)".

2 C-based hardware design environment

Our behavior synthesis system "Cyber" reads behavioral C(BDL) and VHDL, and generates "logic synthesizable" RT descriptions in C(BDL), Verilog, VHDL, NEC original RT language and internal format to our logic synthesizer (Fig.1). Designer can verify his hardware algorithm by C (or behavioral VHDL) simulation and verify cycle behavior at RTL simula-

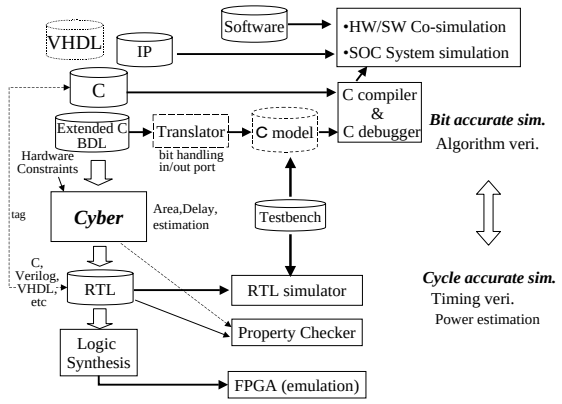

Figure 1: C-based Hardware Design Environment

tion. It is very important that algorithm and timing verification can be done separately. Currently, a designer has to find out both algorithm bugs and timing bugs by RT simulation. This is so confusing that RT simulation period becomes so long. Other merits of behavioral C simulation is as follows: C simulation is much faster (100x) than RT simulation, and a hardware C model can be easily simulated with software model then HW/SW co-simulation could be handled with usual C environment.

Cyber generates "bit and cycle accurate C code" at RT level, which is compilable with usual C compiler and which runs much faster than RT-VHDL or RT-Verilog simulator, therefore, designer don't have to know Verilog or VHDL. Currently, commercial logic synthesizer users have to go through RTL Verilog or VHDL.

In our synthesis environment, IP's can be at any level; behavior, RT, gate or transistor level. IP's lower than behavior level, are called as functions in behavior description, and synthesized as structural components by behavior synthesis. Many designer face the difficulties in using RT level IP's, but behavior level IP's are easier to use since they can be various number of cycle circuit and can be dissolved into a main process at behavior synthesis phase, where IP's and main process can share hardware resources. We believe behavioral IP's might be as useful as library functions in software.

The difficulties for C simulation is "bit accurate" simulation, which is very important especially for DSP area; e.g. graphical filters, encryptions. As stated in the previous section, BDL, an extended C language, has bit handling expressions such as bit-length declaration, bit extraction, concatenation. The BDL description is automatically transformed into bit accurate C code. The important thing is that designer describes only one description both for behavior simulation and behavior synthesis. All modifications, even if they derived from the found bugs or delay problems at RT simulation or gate simulation, should be made at behavioral description. For such sake, back annotation function from RT or gate level to BDL(C) is very important. In our environment, generated RT description has tags to corresponding behavior (e.g. line number). In addition, snap-shots(e.g. dump or print command) for some variables in behavior, are passed to synthesized RT description. The designated variables are dumped also at RT simulation. Mapping information from variables to registers, from arrays to memories, from operations to function units are also provided. It is also easy to find out corresponding behavior code from generated gate level circuits, even if they are synthesized by a commercial logic synthesis tool, since the generated micro architecture has some rules of structure and and regular naming rules.

Additional attractive merit of C-based synthesis is empowering property checking. Current property checker can verify only small circuit with a few hundred FF's, however, with information from behavioral synthesis system, it can verify much larger circuit.

Since Cyber has 10 years history, we received many requests from designers, and we implemented so many synthesis options to generate various types of circuits according to circuit constraints. Some of those synthesis features will be explained in the next section with design examples.

3 Issues through design experiences

In this section, we discuss several issues for C-based synthesis methodology with some successful design experiences.

3.1 NIC chip for PC cluster (Control dominated circuit)

Firmware or hardware for large embedded controller? We designed a network interface card (NIC) chip for Windows NT cluster, based on the "Virtual Interface Architecture" by C-based synthesis with Cyber. This chip has very complicated control, so initially, designers try to design a master controller with firmware for this chip since designing this controller at RT level is too complex. However, this NIC chip requires 1.25Gbps and this speed was a bit too fast for firmware control. Therefore, they decided to try C-based behavioral synthesis. The behavior description is 23,000 lines in BDL ("clocking" expressions are used), which consists of more than 20 parallel communicative processes. The synthesized circuit contains

more than 1,000 states in total. The largest controlling FSM has more than 300 states. This chip is so complicated that one of the designers said, "we couldn't design such complex NIC chip if we haven't used behavior synthesis". Also, he added, "behavior description for Cyber is much easier than firmware description, because we can use flexible control like multiple branches and we don't have to worry about data transfer problem between registers and busses which is essential for firmware design". The chip area was reduced by approximately 40% and performance was improved by around 30% from designers expectations. This example shows that many MPU(software) controlled chip can be replaced by faster and lower-power hardware since the hardware can be synthesized from the original C program for software control. This means behavior synthesis enlarges hardware domain for embedded controllers.

Controlling delay of FSM: The delay of controlling FSM is not negligible to that of datapath in control dominated circuits. Thus, minimizing FSM delays is a crucial synthesis task. State encoding techniques at logic synthesis is not powerful enough to control delays and area of large FSM, so we apply an unique behavioral partitioning technique which can partition a FSM into several FSMs of *any* number of states. This partitioning give us controllers of target performance.

3.2 DSP chip for CCD video camera (Data intensive circuit)

Design unit size for C-based synthesis: Real time color graphic processor for CCD video camera, consists of many modules, e.g. frame buffer read/write part, black level adjustor, matrix transformer, white balancing, high pass filter, gamma adjustor, etc. Those modules are appropriate size for RT-based design, or logic synthesis, and it is OK also for C-based behavior synthesis. However, they are too small to take full advantage of C-based synthesis. We synthesized this whole chip as one synthesis unit. The above modules are dissolved into one process.

Automatic pipelining: The whole behavior is synthesized as a sequential circuit of 10 cycles at first. Then, it is automatically transformed into a pipelined circuit which feeds data every cycle with 10 cycle latency. The behavior included conditionals of different number of operations, which is not easy for manual pipelining. Automatic pipelining of entire behavior is a very effective synthesis function for DSP area.

Fast simulation by rapid-prototyping: RT simulation for graphical data processing takes very long time, such as 1 day long for one frame. Then, as a fast cycle accurate verification tool, our environment provide FPGA emulator which is rapidly synthesized from the behavior. The turn-around-time from behavior to FPGA is not so short like software simulation, but not so long. For this case, 10 seconds for Cyber synthesis, 10 minutes for logic synthesis and 40 minutes for commercial FPGA layout and FPGA board configuration. Designer found out an "algorithm" bug which occurs only after one whole frame is processed. This kind of bugs are difficult to find by RT simulation.

An advantage of our emulation on our C-based environment is that designer can describe debugging functions (e.g. observation of state register from IO pins, check illegal conditions and output alarm to IO's) in a behavior (by #ifdef in C) and synthesize the debugging circuit onto the FPGA of which only IO pins can be probed. Final circuit without debugging circuit can be synthesized just by changing compile option.

This C-based synthesis environment enable us to designed this chip just with one man-month, while it took 10 man-month by RT-based design.

3.3 v42bis data compression

Existent C program into hardware: The C codes for the above two examples are newly described as a hardware behavior, but this example is initially described as a C application program and then used as a hardware description. The modifications to the existent software is proceeded as follows: original C description feeds entire ASCII file from the standard input, then generates compressed file to the standard output. Designer changed the program flow such that one ASCII character in a FIFO is fed and compressed in sequence. Then, he add some descriptions such as handshaking with FIFO, initial request, flush request, etc, to satisfy the hardware IO specifications. Lastly, he modified integer variable into smaller bit length(1,8 or 12bit) variable to get a smaller circuit. The modified C(BDL) can be compiled by usual C compiler and also synthesized in a minutes by Cyber. Though it is said that v42bis data compression algorithm is not suitable for hardware since the algorithm is sequential, this V42bis data compression chip is synthesized and successfully verified just with one week.

3.4 SDH AU pointer controller

Application specific synthesis: This example is the first commercial chip designed by Cyber in 1994.

In SDH network, many channels data are transmitted by time-multiplexed way. Though the processing of one channel is complicated, time-multiplexed behavior for many channels makes design more difficult, even if it is described at behavior level. Time-multiplexing is a typical problem for transmission equipments, so we provided domain specific synthesis function. This function generates circuit for time-multiplexed data automatically from the behavior for one channel.

The synthesized circuit and manual design has very different architecture, even though their IO behaver is equal. C-based synthesis gave us not only 1/10 design period, but less area and better performance chip than manual design. This example shows us that application specific synthesis could be a strong design leverage.

3.5 DES Encryption

Processor vs virtual computing: The feature of encryption algorithm is long bit-length data(64bit, 128bit) operation and many bit-extraction and -insertion operations, which is realized by "shift" and "mask" operations in C language, These data type and operations are not performed efficiently by general purpose processor. We experimented to compare the performance of synthesized FPGA and a Pentium processor for a DES encryption program. The result is remarkable. Even though the clock frequency of FPGA is less than 1/10 of Pentium, encryption speed of FPGA is 800 times faster than Pentium.[1] This means the encryption code on Pentium requires 8,000 times more cycles than synthesized FPGA dedicated to encryption. This performance of the FPGA synthesized from C program, gives us a dream of "virtual (reconfigurable) computing" concept for embedded system and also for general computing system.

4 Summary

As discussed above, C-based synthesis has already reached at practical phase. Actually, it leverages hardware design and enlarges hardware domain for embedded systems. This design methodology is suitable for HW/SW co-design or System-On-Chip design. Behavioral IP's, and reuse of behavior description would be a key to cope with huge SOC chips.

However, there are many barriers to introduce C-based behavioral hardware design methodology to real design teams. Design teams for large equipments are hierarchically constructed such as system design group, RT design, circuit design and layout design. Many system designers use human language as a specification language, and unfortunately, not many of them know C language. Currently, system designer only have to pass their specification to RT designers, and RT designer have to verify algorithm and cycle behavior formally by RT simulation. However, C-based design methodology requires system designers to formalize their design and to simulate it. They are not familiar to CAD tools and sometime don't like to use them. Secondly, not many RT designers know C language, either. Since they are accustomed to concurrent language like RT-VHDL, RT-Verilog, they say they feel sequential language like C as somewhat strange. They also claim that they feel anxiety to design circuit without images of structural RT components. Lastly, behavioral synthesis tool must become more optimizing power than C compiler since requirements for size and speed for hardware is more strict than software. Current behavior synthesis is not mature enough to generate excellent circuits without tune-up the behavioral description. Manuals telling what C description produces good hardware must be provided. We are implementing new synthesis functions every week, which enables us to synthesize better quality circuits. However, too many synthesis options confuses users. Automatic selection of these synthesis options is important, though it is very difficult since good circuit is varying according to the constraints.

On the other hand, good news is that C-based hardware synthesis conducts software designers to come into hardware fields. Embedded system divisions with large software group and small hardware group, didn't try to design chips by themselves instead of software on processor, since they believe chip design costs huge man-power. However, they began to think of synthesizing chips from their C programs by tuning it for hardware by the proposed environment. We found that good software designer describe good C code for hardware by our training experiences. Finally, we believe that hardware design at 2000's will be dominated by C-based behavioral design instead of RT-based design.

[1] We measured the Pentium speed on Windows95 by echoing characters on the display. Therefore, Pentium speed includes file IO's. However, if you encrypt your data on your windows machine, this is the encryption speed you can get. In addition, Cyber, of course, is able to generates various speed circuit. We picked a moderate sized circuit.