Generation of Control Signals in High-Level Synthesis from SDL Specification

Sang-Hoon Kwak*, EuiSeok Kim*, Dong-Ik Lee*, Young-Seok Baek** and In-hak Park**

* Department of Information and Communications, Kwang-Ju Institute of Science and Technology

1 Oryong-Dong Puk-ku Kwang-Ju, South Korea

Phone: +82-62-970-2267, Fax: +82-62-970-2204

E-MAIL : shkwak@geguri.kjist.ac.kr, uskim@geguri.kjist.ac.kr, dilee@kjist.ac.kr

** IC Design Department, Electronics and Telecommunications Research Institute

161 Kajong-dong, Yusong-Gu, TAEJON, 305-350, KOREA

Abstract –This paper suggests a methodology in which control signals for high-level synthesis are generated from SDL specification. SDL is based on EFSM(Extended Finite State Machine) model. Data path and control part are partitioned into representing data operations in the form of scheduled data flow graph and process behavior of an SDL code in forms of an abstract FSM. Resource allocation is performed based on the suggested architecture model and local control signals to drive allocated functional blocks are incorporated into an abstract FSM extracted from an SDL process specification. Data path and global controller acquired through suggested methodology are combined into structural VHDL representation and correctness of behavior for final circuit is verified through waveform simulation.

I. INTRODUCTION

The number of transistors in a modern VLSI chip amounts to several millions and the system-on-a-chip trend has become more remarkable. The trends require higher-level specification and modeling scheme.

SDL is an international standard for a specification of communication systems, i.e. ITU-T Recommendation, Z.100. Due to the implementation independence properties, available tools for simulation and code generation, and wide prevalence as a system specification[1], SDL has been recently considered as an initial specification language for hardware/software co-design. Powerful tools and methodologies based on SDL have been developed for software part. However its hardware application is not the case.

On the other hand, HDLs such as VHDL and Verilog have become very popular and helpful for VLSI designers, especially in ASIC design, but its hardware oriented property and lack of abstraction capability make their use of system-level design difficult[2]. Use of SDL in hardware design makes it possible for hardware designers to give a system-level specification and verify the behavior of system at top level easily.

The objective of this work is to suggest a hardware synthesis framework in which an SDL specification of a VLSI system is automatically converted into gates and functional blocks. Previous work to use SDL specification in hardware design focuses on translating SDL code to equivalent VHDL code[3, 4]. Howerver in this paper an SDL specification is synthesized into gates and functional block-level circuit directly. Required functional blocks during synthesis are assumed to be pre-designed. Overview of the suggested methodology is depicted in Fig.1

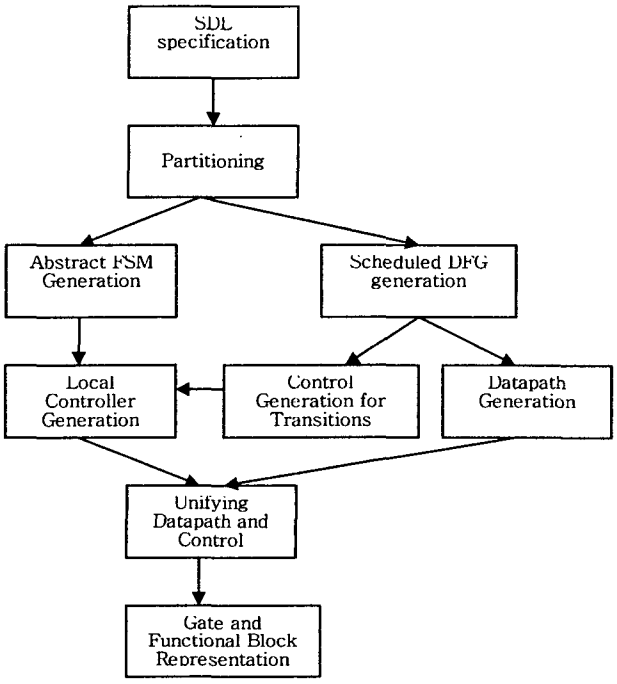

Fig.1 Overall Flow of the Methodology

This paper is organized as follows. Section II presents architecture model that the suggested methodology uses, and Section III explains how control signals form SDL process behavior are generated and they are combined with datapath control signals. In section IV, an example using suggested methodology is given, and conclusion is presented in section V.

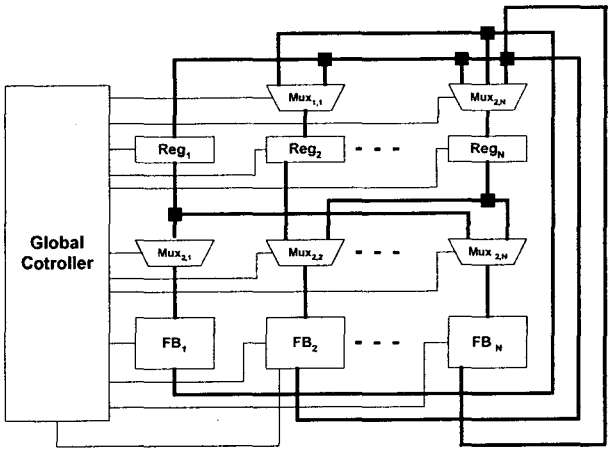

Fig.2 Architecture Model Used in the Suggested Methodology

II. ARCHITECTURE MODEL

SDL provides three reserved words, **system, block,** and **process** for a system and substructure of a system. System means a complete object that a designer tries to specify in real application, block and process are used to describe behavior of parts of a system. Communications between processes and blocks are performed by exchanging signals that can carry data variables conceptually. So we present an architecture model to cover behavior of terminal process and communication of SDL objects (processes, blocks).

A process that corresponds to a terminal component in a hierarchical SDL structure is implemented based on the following architecture. Functional blocks and registers required in a process are allocated. A multiplexer is connected to each input/output port of functional blocks or registers. In the case of using some resources with different input sources, multiplexers permit to select one input according to control signals [5, 6]. Architecture model used in this methodology is presented in Fig.2.

Resource allocation procedure is performed over the suggested architecture model, and resource sharing is accomplished by the multiplexers. Control signals that drive functional blocks and multiplexers are generated by local controllers for a process, and all the control signals are sampled at the rising edge of clock. Global structure of an SDL system, where SDL system implies a system specified by SDL, including multiple processes directly corresponds to hierarchical structure of implemented hardware. All the SDL objects like processes and blocks are mapped to corresponding hardware modules, and all communication paths like signal routes and channels are mapped to data buses and control wires.

A signal object of SDL which is an event to system affects the system behavior as an input or an output. This event is discrete object conceptually, but hardware signal has 1, 0 value physically. Therefore we need to interpret signal object in implementation of hardware from another

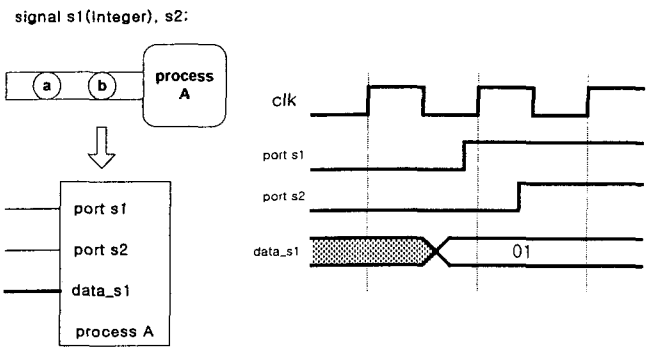

Fig.3 Port Organization for Input Signals and Corresponding Waveforms

point of view.

Since the methodology suggested in this paper assumes that the specification is implemented in a clock driven synchronous circuit, a state transition of a process and data transfer to a register are performed at the rising edge of global clock. Arrival of a signal object to a process are modeled in such a way that for, an incoming SDL signal, hardware signal has 1 value at the designated port. Input and output ports are assigned for distinct SDL input and output signals. Output signal transmitted from one SDL process to another one should be stable at least for a rising edge of the clock so that later one can accept the signal. For data values carried with a signal, extra ports are to be assigned, and the data values are assumed to be stable before the signal input comes into a process. Fig.3 represents the organization of hardware block corresponding to the situation that signals s1 and s2 enter into a process named process A. The signal S1 carries an integer type of data variable.

III. SYNTHESIS FROM SDL SPECIFICATION

SDL specification is partitioned into data operation part and control part and each part is represented by a scheduled data flow graph, in short an SDFG, and an abstract FSM, respectively. Abstract FSM is defined to describe only the behavior of FSM of an SDL process. An SDFG is constructed for one transition of original SDL process and is assumed to be already available for the synthesis procedure in this paper.

A local controller for a process is made by incorporating control signals into an abstract FSM. New states are assigned between two states of an abstract FSM according to the number of time steps of an SDFG. The control signals appear as outputs of an abstract FSM, and include selection signals for multiplexers, and enabling signals for registers. Resource allocation procedure generates datapath from an SDFG based on the suggested architecture model , and local control signals to control functional blocks in

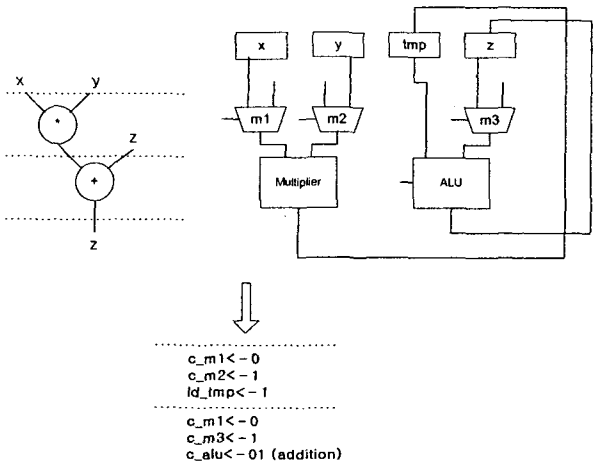

Fig.4 Control Signal Generation from Scheduled DFG

datapath are implicitly decided according to the allocated datapath.

Algorithm 1 describes how control signals for functional blocks and MUXs are generated. In the algorithm, TR is a set of transitions to be able to occur in an SDL process, $SDFG_{tr}$ is a scheduled data flow graph for data operations to occur during a state transition, and op_i is a node in an SDFG which means an operation to be performed, and OP(j) is the set of all nodes at time step j of an SDFG. reg_{target} is a register to store result of operation performed by a functional block in a clock cycle, which allows resource sharing. Operation code means the control code required when a functional block can perform multiple operations according to control code as most ALU do or when extra control code is required. Fig.4 depicts an example which shows how control signals to functional blocks, multiplexers, and registers are generated.

Algorithm 1 - Control Signal Generation for Data-Path
for all *tr* $\in$ *TR*
  if ($SDFG_{tr}$) $\neq \varnothing$ *then*
  for *j* := *1* ***to*** *the number of last step in* $SDFG_{tr}$
    for all $op_i \in OP(j)$
      Find functional block FB_k *allocated for* op_i
      Specify operation code of FB_k *for* op_i
      Find input source of FB_k *for* op_i *and Specify correct MUX control code*
      Find reg_{target} *and specify enable signals for* reg_{target}
      Specify MUX control code for reg_{target} *so that* FB_k*'s output is selected*
    end for
  end for
 end if
end for

After resource allocation procedure, control and an abstract FSM acquired from control part are combined to generate local control signals. Fig.5 shows how the composition is performed. Fig.5 (a) depicts a graphical

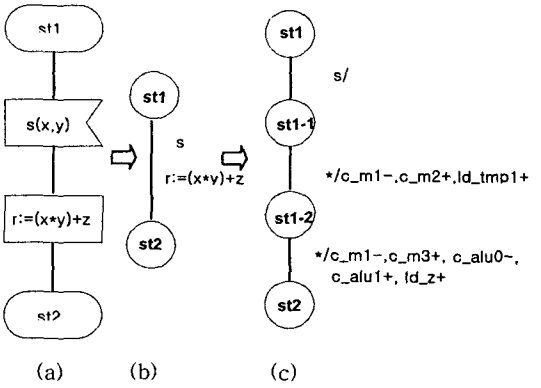

Fig.5 Composition of Abstract FSM and Control Signals

SDL representation of a transition of a process, c_m1, c_m2 and c_m3 represent control signals for multiplexers, c_alu0 and c_alu1 are control signals for ALU, and ld_z, ld_tmp are enabling signals for registers.

Final circuits are obtained by port mapping local controller's output to each functional block's control input and are simulated by a VHDL simulator. The following algorithm describes a composition algorithm in which control signals are incorporated into an abstract FSM. A local FSM which is generated by inserting control signals into an abstract FSM implements a local controller for a process.

Algorithm 2 - Composition of Control Signals and Abstract FSM
while *(non-terminal state in an SDL process)* ***do***
  when *meet a state in abstract FSM, assign a new state in the local FSM*
  Give an input signal in an abstract FSM as local FSM's input
  if *SDFG for a current state transition exists* ***then***
    while *(time step remains in an SDFG)* ***do***
      For a time step in an SDFG assign a new state in a local FSM
      Specify control signals for data modules driven in current step as local FSM's output signal
      Specify input signal as don't care input
    end while
  end if
  Specify abstract FSM's outputs as local FSM's outputs
end while

IV. BARCODE READER EXAMPLE

A controller for a barcode reader is synthesized by the proposed methodology. This example is taken from 1995 High-Level Design Repository[8]. The algorithm used by the reader is composed of reading in bits from the optical scanner and recording the width of black and white striptes. The VHDL model of the reader is also available in the repository.

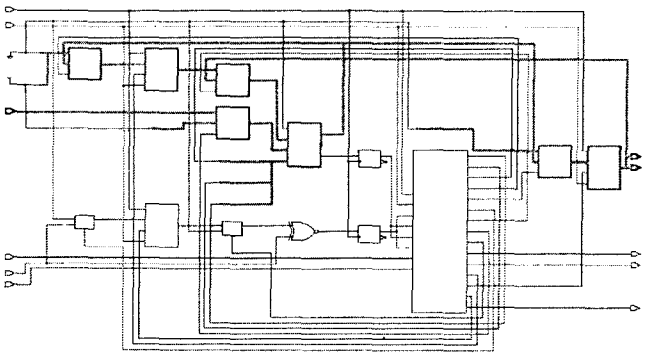

Fig.6 Block Diagram of Barcode Reader

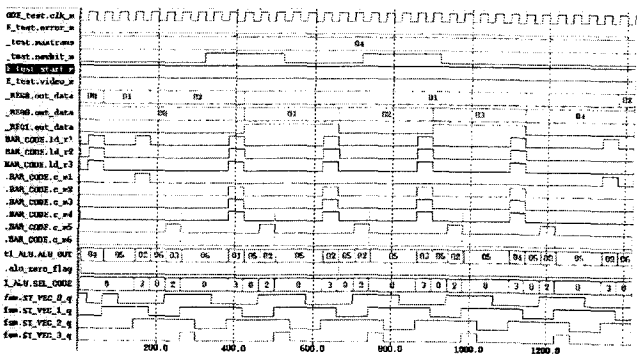

Fig.7 Simulation Result of the Barcode Reader

Initial specification for the controller is given in SDL source code. Its behavior is verified at system-level using an SDL simulator. Fig.6 and Fig7 represent block diagram synthesized by the suggested methodology and waveform that shows the behavior of the barcode reader, respectively.

V. CONCLUSION

We suggest a hardware synthesis framework by which an SDL specification is synthesized into hardware without being converted into HDL. Further in the course of synthesis, system level specification is checked using a SDL simulator. Thus the advantages of early prototyping and the implementation independent property of the languages are certified. It is shown by experimental results that the proposed methodology is comparable to conventional VHDL based design in terms of size of circuit.

With the proposed methodology we reduce the gap between specification and hardware implementation. This work presents the methodology using system level specification with regard to high-level synthesis. However there is much room for further study such as optimization of synthesized circuits in terms of area and/or performance and architecture. And the applicability to hardware/software co-design remains for further research.

Acknowledgement
This work is supported in part by Ministry of Education through the BK 21 Information Technology program.

REFERENCES

[1] J. Ellsberger, D. Hogrefe, and A. Sarma, "SDL : Formal Object-oriented Language for Communicating Systems", Prentice-Hall, 1997

[2] S. Narayan, and D. Gajski, Features Supporting System-Level Specification in HDLs, In Proceedings of EURO-DAC, pages 540-545, 1993

[3] O. Pullkkinen, and K. Kronlõf, "Integration of SDL and VHDL for High-Level Digital Design", In Proceedings of DAC, pages 624-629, , 1992

[4] I.S. Bonatti, and R.J.O. Figueiredo, "Stoht – An SDL-to-Hardware Translator", In Proceedings of ASP-DAC, pages 33-36, 1995

[5] G. De Micheli, "Synthesis and Optimization of Digital Circuit", McGraw, Inc., 1994

[6] D. Gajski, N.D. Dutt, A. Wu, and S. Lin, "High-Level Synthesis : Introduction to Chip and System Design", Kluwer Academic Publishers, 1992

[7] ITU-T Recommendation "Z.100; CCITT Specification and Description Language (SDL) ", ITU-T, 1993

[8] P.R. Panda, N.D. Dutt, "1995 High Level Synthesis Design Repository", In Proceedings of the Eighth International Symposium on System Synthesis, 1995