# *The Design of a Small Footprint Versa Writer Control Chip*

Mariya Kawamura, Tokai University, Japan
Shuhei Tomiyama, Tokai University, Japan
Naohiko Shimizu, Tokai University, Japan

The Asian Undergraduate Research Symposium 2016
Official Conference Proceedings

**Abstract**
High-level synthesis tools have been drawing attention because we can reduce design costs. However, it seems that we cannot consider circuit timing when we design circuits. In this paper, we present the design of a versa writer control chip with NSL (Next Synthesis Language). The versa writer consists of our control chip, 24 RGB LEDs, a clock generator, battery and an I2C serial EEPROM. We place the LEDs in-line to make the vertical line of the picture and characters. We utilize the human afterimage to make the images on the air. The inline LEDs will display the vertical line one by one for a short period. When users shake the versa writer, the afterimage of the inline LEDs makes the pictures. We store the data of the pictures in ROM in advance. We designed the control circuits of LED and ROM with NSL. We checked the operation of the versa writer on FPGA. We succeeded in displaying of the pictures. Our result indicates that we can develop a critical timing system with the high-level synthesis language. Then, we designed a layout of the circuit with an open source EDA tool Alliance. We fabricated a prototype chip from the layout.

Keywords: versa writer, CMOS ASIC, high level HDL design

## 1. Introduction

We can program logic circuits on the device FPGA (Field Programmable Gate Array). Hardware processing with FPGA is lower power consumption and faster processing speed than software processing with CPU. Thus FPGA has been used in many fields. As an example, Microsoft has used servers equipped with FPGA, and it has speeded up Bing search engine. In financial field, HFT (High-Frequency Trading) has worked on FPGA. Image processing and communication processing also have been speeded up like these. In the future, application of these to artificial intelligence is expected.

We generally used some RTL (Register Transfer Level) descriptions such as Verilog HDL and VHDL when we design circuits. However, there is a problem that RTL descriptions take a long time when we program complex processing. Therefore, high-level synthesis tools have been drawing attention because we can reduce design costs. We design circuits with high-level synthesis languages. The high-level synthesis tools convert the circuits into RTL descriptions. Many researchers develop a system with the tools. By way of example, as tools based on C, there have been CoDeveloper (Ohno, Nakahara, Izumi, & Lin, 2014; Kawai & Izumi, 2014), Bach system (Nagai, Kambe, & Fujita, 2014), Cyber Work Bench (Sugimoto, Miyajima, Kuhara, Mituishi, & Amano, 2014), Vivado (Georgopoulos et al., 2016; O'Loughlin, Coffey, Callaly, Lyons, & Morgar, 2014), and so on. As tools based on Java, there have been JavaRock (Miyoshi & Funada, 2011), MaxCompiler (Fukui & Fujita, 2011), and so on. In addition, as tools based on Python, there have been PolyPhony (Sinby Corporation, n.d.), and so on.

Circuit descriptions used the high-level synthesis languages are easy because the circuit descriptions have a higher level of abstraction than the RTL descriptions. However, in the case of using the languages, it seems that we cannot have considered circuit timing when we design circuits. Thus we developed a system used a high-level synthesis language NSL (Next Synthesis Language). Here we present that we can develop a critical timing system with the high-level synthesis language. To achieve this goal, we design a versa writer control circuit with NSL. As another problem, it seems that we cannot have designed chip layouts with these languages. Therefore, we designed a layout with an open source EDA (Electronic Design Automation) tool Alliance. Alliance generates chip layouts consistently after converting NSL into RTL descriptions. Here we present that the tool can generate chip layouts from the high-level synthesis language. To demonstrate this point, we fabricate a versa writer control prototype chip with a generated layout.

## 2. Methods

### 2.1. Specification of NSL

NSL is the hardware description language (HDL) developed by Overtone Corporation. Unlike existing HDL, behavior level descriptions are possible. It is relatively easy to learn NSL because the syntax of NSL is like C. It is possible to convert to Verilog, VHDL, and SystemC with the tool NSL Core. Thus we can perform logic synthesis and simulation with NSL.

## 2.2. Specification of the prototype chip

Fig. 1 shows the specification of our prototype chip. In the case of CMOS process of our chip, the minimum gate length is 2μm as well as the chip has a layer of polysilicon gates and two layers of aluminum wiring. The chip size is 3.2mm square, but we cannot design any circuits in the area with Input/Output pins and so on. Thus we can design layouts in only 2.5mm square area. The layout area cannot contain large circuits because the chip size is small and a lot of fabricating is by hand.

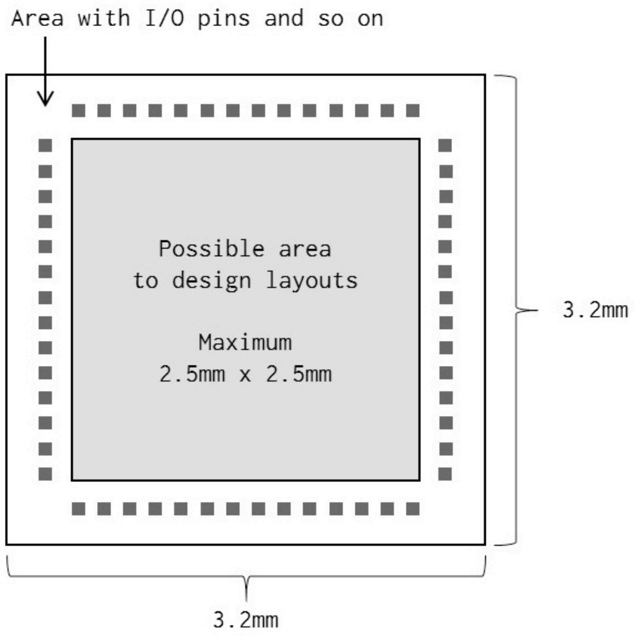

Figure 1 The specification of our prototype chip

## 2.3. Specification of a versa writer

Fig. 2 shows the specification of a versa writer. The versa writer is a device shaped like a rod. Some LEDs are lined up on the versa writer. We used 24 RGB LEDs. When we shake the versa writer in the dark, an afterimage makes pictures and characters. The afterimage is like a pixel art. We stored data of the pictures and characters in an I2C serial EEPROM in advance.

We explain an operating principle of the versa writer. First, once a versa writer controller transfers the first line of data, the LEDs turned on as the first data. Then, the controller transfers the second line of data once we shake the versa writer only a little. After transferring the second data, an afterimage leaves the first data, and the LEDs turned on as the second data. The afterimage makes pictures and characters by repeating this processing for a short period.

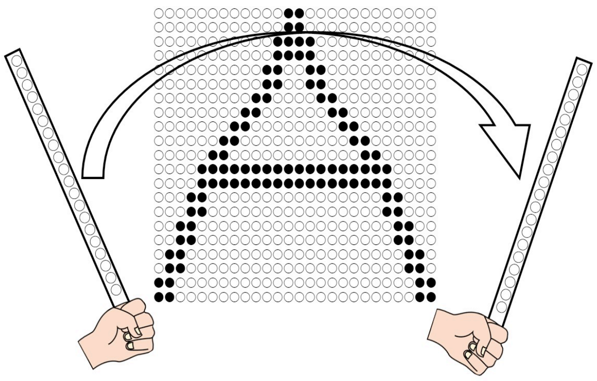
Figure 2 The specification of a versa writer

## 2.4. Structure of versa writer

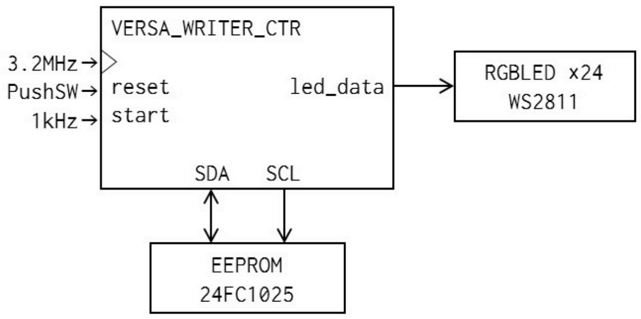

Figure 3 The structure of the versa writer

Fig. 3 shows the structure of the versa writer. The versa writer control circuit transfers data to RGB LEDs. We store the data in an EEPROM. The circuit changes a lighting pattern of LEDs whenever 1kHz start signal rises. We use RGB LEDs WS2811 and an EEPROM 24FC1025. Both LED and ROM operate at 400kHz. We have to reduce the circuit scale because we fabricate a versa writer control prototype chip. Thus we have to reduce the number of registers in the circuit. We do not have to use any buffers because LEDs and ROM operate at the same timing.

## 2.5. RGBLED control circuit

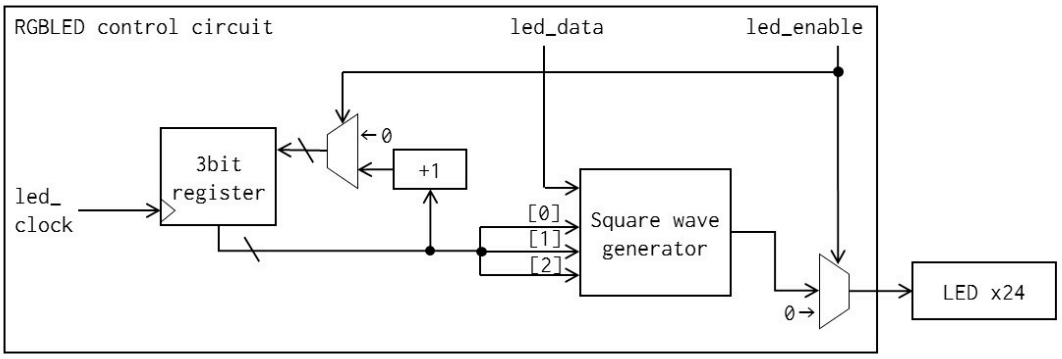

Figure 4 LED control circuit

A LED control circuit is described in Fig. 4. Once a ROM circuit transfers data as led_data, the circuit outputs a square wave. The wave corresponds to the data. In the case of the LED, a wave of 50% duty cycle is 1, and one of 12.5% duty cycle is 0. We generate the wave with a counter. The counter synchronize with a 3.2MHz clock because transferring a wave has to synchronize with a 400kHz clock. Data outputted to the LED is 24bit. The data contains 8bits of red data, 8bits of green data and 8bits of blue data. A LED is turned on or off once the circuit transfers 24bits of data to the LED. As an example, the LED is turned on in red once the circuit transfers 0xFF0000. In addition, we can connect more than one LED in the way of cascade connection. The circuit can output each data to all LEDs because the LEDs have a microcomputer. The circuit has to output a reset signal for over 50μs after the circuit outputs as many data of 24bits as the number of the LEDs connected. The circuit decides whether it outputs the wave or the reset signal. The circuit transfers the wave when led_enable is 1, and it transfers the reset signal when led_enable is 0.

## 2.6. EEPROM control circuit

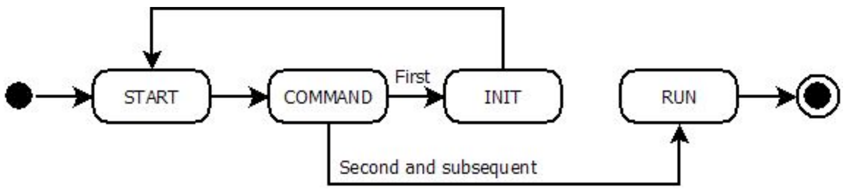

Figure 5 A state transition diagram of a ROM control circuit

Fig. 5 shows a state transition diagram of a ROM control circuit. There are SDA (Serial DAta) signal and SCL (Serial CLock) in the ROM. We use SDA signal for sending commands, and we use it for sending and receiving data. The ROM sends and receives a bit of a command or a data at a time since the ROM uses serial communication. The circuit outputs 400kHz clock to SCL signal.

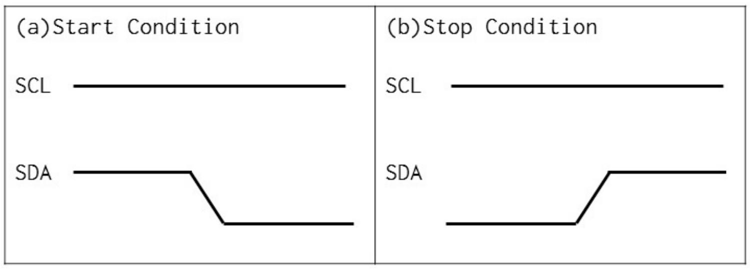

Figure 6 A wave of start/stop condition

The circuit sends a start data to ROM in the START state. The way of sending the start data is that SDA signal is changed to LOW from HIGH while SCL signal is HIGH as indicated in Fig. 6(a). The circuit sends a stop data to ROM in finishing processing. The way of sending the stop data is that SDA signal is changed to HIGH from LOW while SCL signal is HIGH as indicated in Fig. 6(b).

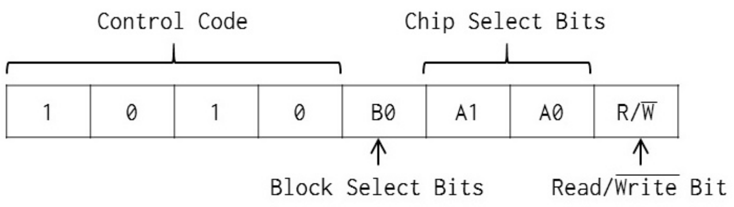

Figure 7 The format of the control byte

During the COMMAND state, the circuit sends information about doing now. The circuit uses a control byte for sending information. Fig. 7 shows the format of the control byte. First, the circuit sends 4bits of the control code ‘1010’, then selects a block of ROM. We select 0 when we want to use the former half of blocks, and we select 1 when we want to use the latter one. If we connect more than one ROM, we specify next 2bits of an identifier to select a ROM. Finally, we select whether reading or writing. In first COMMAND state, we select writing to specify a reading address. In second and subsequent, we select reading to read data. The circuit receives ACK (Acknowledgement) signal from ROM after sending the control byte.

The state of the circuit transitions to the INIT state from first COMMAND state. During the INIT state, we specify an initial address of reading data. The circuit sends 8bits of a lower address after sending 8bits of an upper address. The circuit receives ACK signal after each sending the address.

The state of the circuit transitions to the RUN state from second and subsequent COMMAND state. During the RUN state, the circuit receives 8bits of data in order from the initial address. The circuit sends ACK signal after receiving 8bits of the data. The circuit finishes the processing after receiving as many data as the number of LEDs connected. After the lapse of 1ms, the circuit begins the processing with the START state and receives the following data. By repeating the state transition, the circuit can change data for sending to LEDs at the same time as we shake the versa writer only a little.

## 2.7. Flow of designing layouts

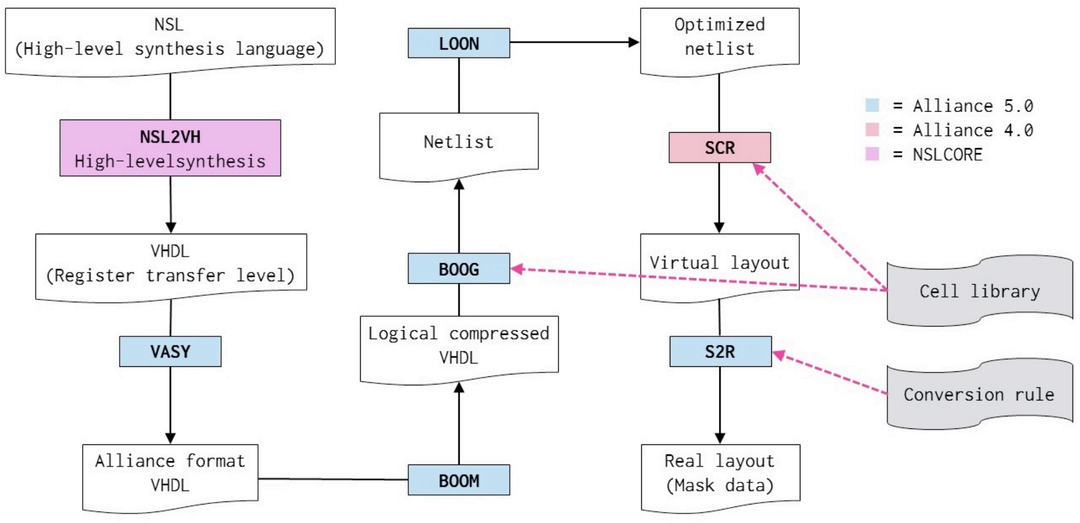

Figure 8 The flow of designing layouts

Fig. 8 shows the flow of designing layouts. We use the open source EDA tool Alliance for designing layouts. A high-level synthesis tool NSL2VH convert NSL descriptions circuit into VHDL descriptions because Alliance uses VHDL. VASY convert into Alliance format VHDL descriptions. BOOM performs logical compression, and BOOG generates a netlist. The netlist corresponds to a cell library. The cell library contains cell patterns of various circuits, for example, an inverter, an AND circuit, an OR circuit, a flip-flop circuit and so on. The netlist is descriptions how to connect these cell patterns. Then LOON optimizes the netlist. SCL generates a virtual layout, and S2R generate a real layout according to a conversion rule.

## 3. Results

First, we designed a layout of our circuit. We fabricated a prototype chip of only the LED control circuit because the layout area could not contain the whole of the versa writer control circuit. Fig. 9 shows the chip layout. We implemented the ROM control circuit on MAXV, and we connected the circuit to the LED control chip.

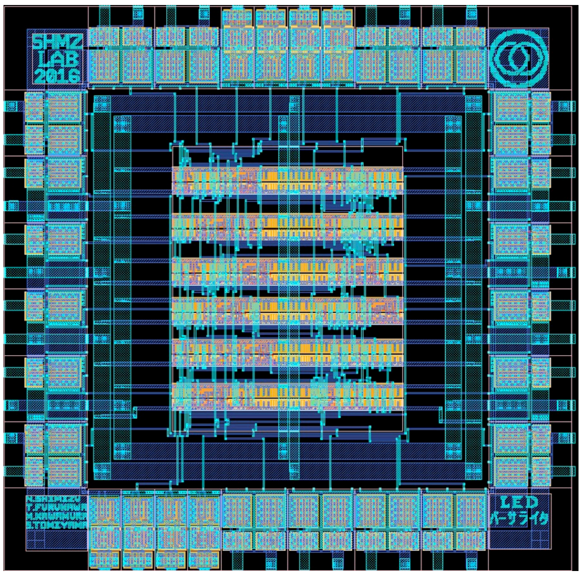

Figure 9 The chip layout of the LED control circuit

We implemented the LED control circuit on DE0-CV, and we tested the circuit before fabricating the prototype chip. We performed logic synthesis with Quartus II after converting NSL into Verilog. Fig. 10 shows the test result. It can be seen that the circuit output a square wave of 12.5% duty cycle when SDA signal is 0. In addition, it can be seen that the circuit output one of 50% duty cycle when SDA signal is 1. We connected the circuit to MAXV, and we checked the operation of the versa writer. Fig. 11 shows the operation result. An afterimage makes characters in the dark. Then we fabricated the prototype chip, but we are checking the operation of the chip.

Accordingly, it is possible to say that we can develop critical timing systems with NSL. We could also design the layout consistently. However, it remains an open question whether the method is effective or not because we cannot have tested the prototype chip yet.

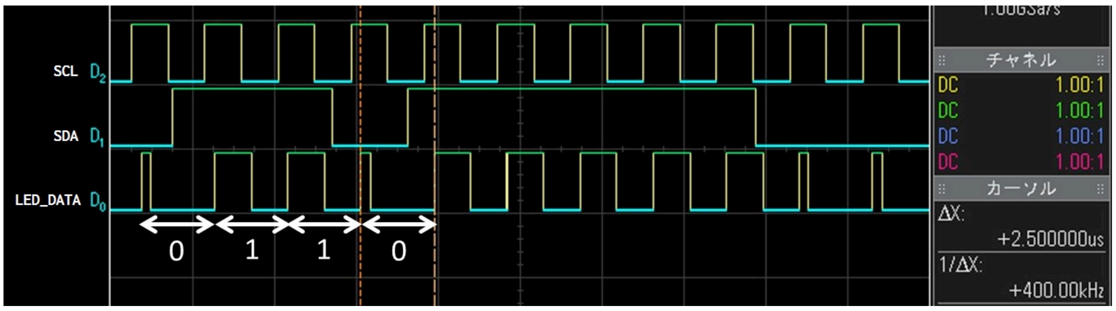

Figure 10 The test result of the versa writer control circuit

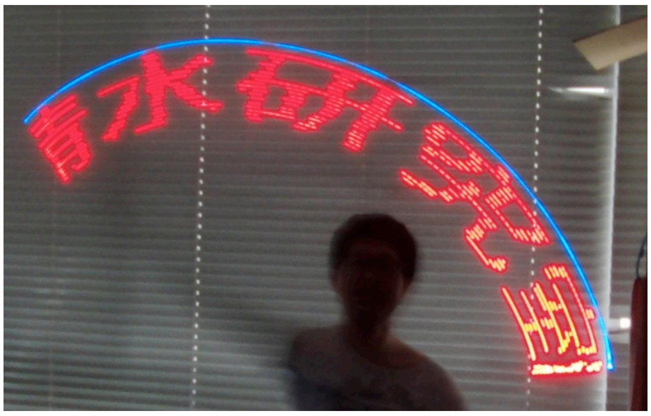

Figure 11 The operation result of the versa writer

## 4. Discussion and Conclusion

Our goal in this paper has been to present that we can develop a critical timing system with the high-level synthesis language. In addition, another goal has been to present that we can also generate chip layouts from the high-level synthesis language. By designing the versa writer control circuit, we presented that we can develop the critical timing system with the language NSL. The considering timing of circuits has received little attention in previous studies. However, as a result of our development, it has come to light that we can develop not only arithmetic processing but also such a system with the high-level synthesis language. About the result of designing the layout, we cannot have tested the prototype chip yet. In the future, we need to test the chip because we need to establish the usefulness of the language NSL and the tool Alliance. Then, we will aim to fabricate the one-chip from the whole of the versa writer control circuit. It remains a challenge for future research to design layouts consistently from modeling languages.

## References

Fukui, A., & Fujita, M. (2011). Acceleration of Smith-Waterman Algorithm on FPGAs. *IEICE Technical Report*, *111*(31), 67-72.

Georgopoulos, K., Chrysos, G., Malakonakis, P., Nikitakis, A., Tampouratzis, N., Dollas, A., … Pnevmatikatos, D. (2016). An evaluation of vivado HLS for efficient system design. Proceedings from 58th International Symposium ELMAR-2016.

Kawai, R., & Izumi, T. (2014). A Design of Blokus Player Algorithm with Impulse High-Level Synthesis Tools. *IEICE Technical Report*, *114*(75), 43-47.

Miyoshi, T., & Funada, S. (2011). A Study of Employing Java as A High-level Synthesis Language for FPGA. *IEICE Technical Report*, *111*(328), 3-8.

Nagai, S., Kambe, T., & Fujita, G. (2014). A Hardware Implementation of Motion Estimation Technology Using High Level synthesis. *IEICE Technical Report*, *113*(454), 73-77.

Ohno, M., Nakahara, Y., Izumi, T., & Lin, M. (2014). A Trial Hardware Design of a Recursive Function to Solve "OX game" by Code-Modification and High-Level Synthesis. *IEICE Technical Report*, *114*(223), 87-92.

O'Loughlin, D., Coffey, A., Callaly, F., Lyons, D., & Morgar, F. (2014). Xilinx

Vivado High Level Synthesis: Case studies. Proceedings from ISSC 2014/CIICT2014.

Sinby Corporation. (n.d.). Introduction of PolyPhony. Retrieved from http://www.sinby.com/PolyPhony/

Sugimoto, N., Miyajima, K., Kuhara, T., Mituishi, T., & Amano, H. (2014). Artificial Intelligence of Blokus Duo on FPGA Using Cyber Work Bench. *IEICE Technical Report*, *113*(416), 13-18.