LightClockV2 - A Motivation for Teaching Scalable Digital Hardware Design

Michael Rathmair, Friedrich Bauer, Marcus Meisel
TU Wien, Institute of Computer Technology
1040 Vienna, Gusshausstrasse 27-29/384, Austria
[michael.rathmair|friedrich.bauer|marcus.meisel]@tuwien.ac.at

Abstract—Learning embedded design can be achieved with simple standard applications and microcontrollers but to spark interest in the topic, to pursue digital hardware design, motivation of students is a key factor demanding new approaches. One of which is presented in this paper as the LightClockV2, a fancy LED-based visualization gadget, built from 60 RGB color LEDs mounted on a circular PCB board to generate radial light beams. For focusing beams, we used laser pointer lenses which are mounted and adjusted in 3D printed holders. A main challenge of the project was the generation of 180 individually controllable pulse width modulation (PWM) signals. Characteristics as a 100 kHz PWM frequency, expandability for other visualization applications and a full single chip solution tighten the project requirements. In an implementation perspective a FPGA based solution efficiently provides an appropriate computational unit (finite state machine) generating light patterns, and 180 PWM peripheral modules. However, the FPGA implementation is a powerful and high-performance realization of this application and faces significant advantages in contrast to a classical microcontroller-based solution.

I. INTRODUCTION AND MOTIVATION

Today's highly innovated technologies in electronic devices and integrated circuits enable besides their usage for professional (industrial) applications the design of smart and cool looking gadgets for our daily life. The decrease of structural sizes enables the integration of higher functional density within a single silicon device [1]. Board level components like external communication peripherals, co-processors, analog signal processing, etc. are integrated to a so-called System on Chip (SoC). Hence, for a wide range of applications today single chip implementations are available. Further advantages such as higher operating speeds of the full system, robustness against electromagnetic influences, etc. arise based on this design methodology.

For this work, we focus on field programmable gate array (FPGA) platforms which are a specific realization of so-called programmable logic devices (PLD). FPGA's include a high number of equal logical standard-cells which are interconnected in a way to realize the indented hardware design. Hence, functionalities of the final application are directly mapped to a specific hardware structure. This mapping process, translating an abstract hardware model to a structural representation, is done by a synthesis tool. In contrast to a microcontroller where an application is realized by sequential software instructions executed on a static hardware architecture, an FPGA allows “real” hardware design. However, FPGA platforms are widely used in industrial designs where high speed, cost-efficient hardware prototyping, test and verification, etc. characteristics play an essential role.

This is where teaching activities come into play. In previous semesters, students typically learn the design of embedded software on microcontroller platforms. If they are suddenly “forced” to solve a standard application on a hardware platform such as an FPGA, the meaningfulness is often questioned. In principle, they are right that the task can often be realized with sufficient performance in a microcontroller. Thus, we were looking for an application which is “not so easy to implement” on a standard microcontroller due to limited hardware resources [2]. Based on a previous bachelor project we planned to implement the second version of our “light clock”. The device uses LEDs mounted on a circular disc to project a light beam on the surface (wall) behind. Three of this focused beams are used to display the hands of a clock showing the current time while all other LEDs can be used to visualize some flashing light effects. The first version (light clock v1) includes 120 just white LEDs which can be dimmed in a multiplexed way. So for version two we absolutely want full-color RGB LEDs and smooth intensity control. At a more precise technical view, we require a high number of pulse width modulation (PWM) units for the control of color mixing and single intensities. Hence, this application fits perfect to our intention of using an FPGA, because we design our own PWM peripheral and place it before each output pin. This should show the students that such an approach to hardware design easily scales in size.

II. EVALUATION OF SYSTEM REQUIREMENTS

Based on the idea of the light clock V2 application described in the introduction we identify the following technical and functional requirements:

High number of PWM channels:
For full RGB operation and a minute wise clock face division, we need 180 LED control channels. For smooth and efficient intensity control each LED is dimmed by a PWM module individually. Due to the danger of supplying the LEDs in an over-current way when multiplexed (as discussed in the next section) we intend 180 real non-multiplexed PWM controllers operating independently from each other.

Clear and focused light beams
For the projection of a light beam as required for the hands of the clock, LEDs have to meet hard optical requirements.
The cores of beams should not overlap for each LED segment.

**Single chip, robust and cheap solution:**
We aim to use the implemented board level hardware (mounted LEDs and drivers) also for other visualization applications. Thus, the full design should fit within a single chip solution. Communication with external components (e.g., microcontrollers, CPLDs, etc.) would increase the complexity of the chip solution. Communication with external components (e.g., applications) thus the full design should fit within a single chip solution. Communication with external components (e.g., applications) would increase the complexity of the chip solution. Communication with external components (e.g., applications) would increase the complexity of the chip solution.

**III. Discussion of Possible Solutions**

A single microcontroller solution for the generation of 180 PWM signals is quite hard (and also not planned for the realization of this project). Standard state of the art microcontrollers (as used for teaching and hobby applications) have just a very low number of independent PWM units (3-10). This is not sufficient for the proposed application. A further way to increase the number of PWM channels is so-called software PWM. In this mode, the timing for switching an output on or off is solved in application software. A variable is used for counting up, and corresponding compare variables toggle the output ports. Due to the increased computational effort on this algorithm, we had the experience that low speed (approximately around 20 MHz) microcontrollers as are just able to create about 20 PWM signals. From that on the full processing power is just used for the PWM algorithm. Hence, such a solution will, however, require to a cascade of numerous microcontrollers in a chain. But as discussed earlier in the requirements section the goal is a single chip solution.

A further approach could be to multiplex PWM signals. However, if you generate a multiplexed signal for each RGB LED the total illumination power of the LEDs will be just one third due to the “shared” signal characteristic. A possible solution to overcome this drawback is to over-current the LEDs and hence boost their illumination. This mechanism has the significant disadvantage that if a single PWM signal however stuck at one, the LED will be destroyed.

Using shift registers could be a reasonable solution. But if we require high PWM frequencies (100 kHz) the serial reload rate for all 180 shift registers is dramatically high. An attractive approach is using the WS2812 LED controller chip. Three internal 8-bit registers of the chip are loaded serially. After each load cycle, the register values are interpreted as duty cycles for a three channel PWM signal [3].

The last solution we discussed is to implement a specifically designed soft-core processing unit for software PWM signal generation as explained previously. However, space and performance for this CPU design are estimated being at the same complexity as the architecture using 180 tiny PWM modules we had in hour mind initially.

Based on the discussion of the described potential approaches we decided to explore the following straight-forward architecture.

1) Design a PWM module in VHDL (very high speed integrated circuit hardware description language) with the following characteristics: - The PWM frequency is 100 kHz. - The PWM accuracy is 8-bit. - Logarithmic duty cycle calculation for linear LED light intensity - Constrain the synthesis tool in a way that an optimized small area design is generated.

2) Generate 180 PWM modules and create an interface to the according duty cycle signals.

3) Design static FSMs implementing cool looking light patterns and the functionalities (clock, push-up counter, etc.) (see the video)∗

**IV. Faced Challenges and Design Limitations**

1) **Available Pins of Evaluation boards**

The first problem we ran into at checking our available evaluation boards at the institute was that we do not have a board where 180 freely usable GPIO (general purpose input/output) ports are available. More precisely, the FPGAs mounted on the board (e.g., Intel’s DE1 development board) has enough free programmable pins, but they are hard-connected to the boards interfaces and peripherals such as VGA, Ethernet PHY, audio decoder, external memory, etc.

2) **Selection of LED chips**

LEDs satisfying the given requirements are hard to discover, especially having an appropriate narrow aperture angle. In the initial design, we planned to use wired LEDs having three emitting diodes within one package. The LEDs are soldered on the outer edge of the circular PCB board. But at the time of writing, there were no wired LEDs at a competitive price available meeting these requirements. Thus, we switched to a solution using SMD LEDs in combination with lenses.

3) **LED and lens holders**

As proposed in the previous paragraph we planned to use an SMD RGB LED chip in combination with a lens of a laser pointer device. Thus, we need a holder for mounting the LED and lens at the outer edge of the PCB. The final adjustment of lenses and LEDs (distance, angle, etc.) is a critical part and motivate to design a robust case which can be 3D printed.

**V. Implementation and Discussion of Results**

In this section, we focus on implementation details and results by following the step by step implementation plan as proposed in Section III.

A. **Implementation of a single PWM module**

The first implementation step is the design of a single PWM module in VHDL. Thus, the entity of the module has a clock input sourced by the desired 100kHz PWM frequency, an 8-bit duty cycle input and a single PWM signal output. The behavior of the module is rather simple. For the consideration of the logarithmic function according to a linear illumination characteristic, we implemented a hard-coded conversation table. Figure 1 shows the simulation results of 16 PWM units. As seen: 1/30 18

*LightClockV2 https://www.youtube.com/watch?v=-zU6-N0Ddps last seen: 1/30 18
a simulation software we used Mentor Questa tools) The signal **slow_clock** indicates the divided clock signal derived from the 50 MHz oscillator clock. This **slow_clock** is connected to each PWM unit for achieving the desired 100 kHz PWM frequency. Each of the 16 PWM modules is sourced with an 8-bit duty cycle vector individually (the signal is not shown in the figure). **sig_led_pwm(0..15)** indicate generate PWM signals which have different duty cycles between 0 and 50 percent. Of course, we generated all 180 PWM modules also for simulation, but only 16 are shown in Figure the figure.

### B. Port expansion design

A very early design decision to overcome challenge 1 is that we will use a second FPGA development board as a port expansion board. Thus, for interfacing the 180 LEDs we use an architecture as illustrated in Figure 2. Master of the system is an Intel (former Altera) DE1 board [4]. An Intel Cyclone II mini development board [5] is used as a port expansion. But the second board is not just a simple output port expansion. The Cyclone II mini also implements the PWM modules and a state machine for the setup of according duty cycles. The data communication between the boards is realized via an 8-bit unidirectional bus where state vector information is transferred from the DE1 to the Cyclone II mini board.

We tested the described communication between the two boards at a real hardware setup, where the DE1 board directs the FSM implemented in the mini board into various states. Each FSM state drives a single LED port with a 50 percent PWM signal (others are set to 0). The corresponding scope plot is illustrated in Figure 3. The number of states and accordingly driven PWM signals is 29, due to limited ports of the used logic analyzer (channels B13 to B15 are used for tracing debug signals). During the test, we just increment the state vector sequentially to illuminate one LED after each other at a PWM-level of 50 percent. For the rest of the design in the Cyclone II mini board we inductively assume that the methodology works for all 95 LEDs and other duty cycle values.

### C. Illumination patterns

As a next step, as we have a template state machine including the 8-bit communication for the Cyclone II mini board and the corresponding 95 PWM modules, we start to implement the final LED sequencer. Patterns we plan to use for the final project presentation are:

- Single color LED beam going around clockwise and counter-clockwise
- Pulsing fade in and out of the full blue circle
- Star pattern rotating in both directions
- VU meter presentation
- Rotating gap going around
- Additive white mixing color scheme fading up and down
- Analog clock representation including hands for hours and minutes.
- Simultaneous clockwise and counter-clockwise rotating pulse
- Push-up counter including the fitness level using red and green bars

For the design of each pattern, more precisely for filling the duty cycle arrays as shown in Figure 5 we created an Excel spreadsheet. This allows designing the patterns in an abstract and clear way. Finally, the corresponding duty cycle values are automatically exported as VHDL array assignments.

Figure 4 shows the simulation results of the proposed star pattern. Signals indicated in the simulation trace are sent to LEDs which are sequentially mounted. As you can see the first not fully visible bit vector signal indicate the duty cycle values which are sent to the corresponding PWM units. Beneath you can see the according generated PWM patterns which fade from narrow pulses to steady switched on signals (in location direction). This results in the designed star illumination pattern. For rotation of the star pattern the presented pulse pattern is shifted up or down for clockwise or counter-clockwise rotation respectively.

### D. Full system simulation

For the simulation of the full system, both designs (DE1 and CycloneII mini board) are combined in a single testbench. The physical 8-bit connection is replaced by a VHDL bitvector signal. For the full system functionality we implemented nine finite state machines for the previously proposed patterns. One master FSM controls switching between the implemented patterns. The selection of a dedicated pattern is called a “mode”. As illustrated in Figure 5 this mode may switch on and off the according sub FSMs displaying the defined LED illumination patterns. The signal **data** indicates the 8bit state vector which is unequal to 0 if a dedicated LED has to be illuminated and its output is physically located on the Cyclone II mini board. The signal **Zustand – engl. state** indicates the according state of the sub FSM is responsible for the behavior of the pattern (selected by **mode**). The array signals **array_duty_red** and **blue** are directly connected to the PWM units of the according DE1 or Cyclone II mini design.

### E. Synthesis results

For the synthesis and implementation flow, we used the Intel Quartus design environment. As proposed we have two designs. The “master” deployed to the DE1 evaluation board which implements the pattern control and 85 of 180 PWM modules, and the port expansion design on a Cyclone II mini development board which controls the rest of the LEDs. Synthesis results of both designs are given in Table 1.

As indicated the Cyclone II mini (EP2C5T144C8 device) gets quite heavy loaded by implementing 95 PWM modules and a control FSM. The 74 % usage of the total capacity of logical elements is a remarkable result from the synthesis tool. The FPGA on the DE1 board (EP2C20F484C7 device) is larger and the deployed design uses just 21 % of totally
Fig. 1. Simulation results of 16 implemented PWM units. Each unit is sourced by the 100 kHz slow_clock signal which indicates the desired PWM frequency. Each PWM module is set up by an according 8-bit duty cycle value which is between 0 and 50 percent in this figure. The generated PWM signals are labeled as sig_led_pwm(0..15).

85 LEDs

8-bit state vector transfer

Altera Terasic - De1 Cyclone II EP2C20F484C7

Altera Cyclone II mini EP2C5T144C8

95 LEDs

TABLE I. CHARACTERISTICS OF THE USED DEVELOPMENT BOARDS

<table>
<thead>
<tr>
<th>Boards</th>
<th>Altera DE1</th>
<th>Altera Cyclone II mini</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>&quot;Master&quot; design</td>
<td>Port expansion</td>
</tr>
<tr>
<td>Family</td>
<td>Cyclone II</td>
<td>Cyclone II mini</td>
</tr>
<tr>
<td>Device</td>
<td>EP2C20F484C7</td>
<td>EP2C5T144C8</td>
</tr>
<tr>
<td>Total logic elements</td>
<td>3,995 / 18,752 (21 %)</td>
<td>3,404 / 4,608 (74 %)</td>
</tr>
<tr>
<td>Dedicated logic registers</td>
<td>901 / 18,752 (5 %)</td>
<td>574 / 4,608 (12 %)</td>
</tr>
<tr>
<td>Maximum frequency</td>
<td>94.55 MHz</td>
<td>91.31 MHz</td>
</tr>
</tbody>
</table>

available logical elements. This allows several extensions for future applications on the master FPGA device. The usage of registers is at a relatively low level (3 % and 12 %) due to the high amount of combinatorial logic of the modules. There are no complex algorithms or specific calculations where lots of variables have to be stored in registers. The low complexity of the design also reflects the achievable maximum clock frequencies of about 90 MHz. Hence, combinatorial paths are short, and there is a valuable buffer to the used board oscillator frequency of 50 MHz.

F. LED and lens holders

As described earlier, we use SMD RGB LEDs where three color diodes are packed within the same case. The electrical and optical characteristics of the LED are given in Table 2.

To display the hands of the clock visualization red, green and blue color should project a focused individual beam onto the surface underneath the clock. Hence, we use one lens of a laser pointer device to focus the three beams of each LED. The lenses and LED chips must be adjusted in a very precise way (distance, angle, rotation, etc.) Thus, we designed a small case for each LED which holds the LED itself and the lens. These 60 cases are printed with our 3D printer. For mounting the cases onto the PCB an adjustment ring is 3D printed as well. On this ring, all cases are circularly adjusted. Figure 6 illustrates an image of four LED and lens cases. As shown Red green and blue color of the LED beams are separated and project focused beams at the surface in front of the holders.

G. PCB design

The design of the printed circuit board (PCB) was mainly influenced by the circular placement of the LED and lens holders. Electrical components as power driver ICs and LED series resistors are placed close to the center of the circular PCB. For connecting the light clock with the FPGA development boards (DE1 and Cyclone II mini) we use flexible cables. Picture 7 shows the final PCB including driver ICs and partially mounted LED holders. The final size of the board is mainly influenced by the placement and size of the LED holders. Connection
Fig. 3. Results of testing the communication of the two used FPGA devices. The DE1 boards increments an 8bit state vector which is transferred to the Cyclone II mini board sequentially. The FSM in the mini board is designed to illuminate a single specific LED at a duty cycle value of 50 percent. As illustrated in the scope plot each of the 29 signals is at a 50 percent PWM duty cycle state as long as the according state vector value is present at the 8-bit connection bus.

Fig. 4. Simulation results of the star pattern. Traced signals are sent to LEDs sequentially, which results that the LEDs illumination intensity varies from nearly dark to full power. For the rotation of the star this signal pattern is shifted which also corresponds a line wise shift of the displayed simulation traces.

plugs to the according FPGA development boards are soldered on the bottom side and not visible in the picture.

VI. Conclusion and Future Work

As a conclusion of this work, we sum up design and implementation steps and discuss educational added value facilitated by this students project.

- First, we designed a simple PWM module which functionally corresponds to PWM modules known from standard microcontroller peripherals. An enhanced implemented functionality is the computation of a logarithmic characteristic for linear illumination intensity. Further, enhanced algorithmic extensions could be identified by, e.g., increasing the accuracy of the PWM at the high sensitivity parts of the logarithmic characteristic. Within this step, students learn to use the VHDL language for designing a PWM peripherals. A further activity is to implement a logarithmic calculation algorithm in hardware.

- As mentioned the single PWM module is “copied” 180 times using the VHDL generate statement. A significant lesson learned for students within this process is that the management and especially providing
The connection of the two FPGA boards using an 8-bit parallel bus is an easy to implement (compared to a serial protocol) but at the end clearly limiting design decision. The limited number of 256 states transferred to the Cyclone II mini board was fully exhausted at the designed LED illumination sequences. However, for the redesign of the system, we will use another FPGA device as the EPC15 having 484 fully controllable pins (pins not connected to external chips as implemented on the available evaluation boards). Thus, the re-designed version will definitely satisfy the requirement of a single-chip solution. In this process, students learn to discover potential alternatives in hardware design and to assess the performance decrease of implementing the "plan B".

Mounting and adjusting the LEDs was from the mechanical perspective the most critical part of the project. The adjustment of LEDs is not perfect. But this is no longer disturbing because surprisingly these uncertainties were resulting in a cool looking "flaming" effect. For a potential re-implementation, a wired LED chip can be used to avoid the manual connection of 4 wires between PCB and LED. However, the manufacturing of the holder and the mounting ring is a great task showing students the potentials of innovative 3D printing technology.

The expandability of the design using the planned 484 pin FPGA device will provide enough space for future improvements such as the connection of a Bluetooth communication module.

Shortly we plan to build an improved version of the LightClockV2 as a further student project. This would not be a complete redesign but considering the lessons learned from this implementation. Finally, we all want to have such a clock at our homes because devices having a high number of flashing LEDs are cool.

ACKNOWLEDGMENT

This work was supported from the Institute of Computer Technology at the TU Wien. It is based on the work done by M. Effenberger and St. Kerstner as their bachelor thesis project.

REFERENCES