Formal and model driven design of the bright light therapy system Luxamet

FAUST, Oliver <http://orcid.org/0000-0002-0352-6716> and YU, Wenwei

Formal and model driven design of the bright light therapy system LUXAMET

Oliver Faust

School of Science and Engineering, Habib University, Karachi, Pakistan

Wenwei Yu

Chiba Graduate School of Engineering, Chiba University, Chiba, Japan

Received (Day Month Year)
Accepted (Day Month Year)

Seasonal depression seriously diminishes the quality of life for many patients. To improve their condition, we propose LUXAMET, a bright light therapy system. This system has the potential relieve patients from some of the symptoms caused by seasonal depression. The system was designed with a formal and model driven design methodology. This methodology enabled us to minimize systemic hazards, like blinding patients with an unhealthy dose of light. This was achieved by controlling race conditions and memory leaks, during design time. We proof that the system specification is deadlock as well as livelock free and there are no invariant violations. These proves, together with the similarity between specification model and implementation code, make us confident that the implemented system is a reliable tool which can help patients during seasonal depression.

Keywords: Formal methods, biomedical engineering, Depression, Bright light system

1991 Mathematics Subject Classification: 22E46, 53C35, 57S20

1. Introduction

Major depression is recurrent disorder, which has neurological bases [1]. The consequences of this brain disorder are serious, because patients experience reduced well being, diminished role functioning, especially in occupational and social roles [2]. As a consequence, patients face a low quality of life, high medical morbidity and above average mortality [3, 4]. In a global study, the World Health Organization (WHO) has identified depression as the fourth leading cause of disability [5]. The same organization projects that by 2020, depression will be the second leading cause [6]. Unfortunately, direct information on the prevalence of depression does not exist for most countries and the available data shows a high variation in the prevalence rates [7]. In the past, depression was not considered to be a serous disease, because the mortality rate is low, especially when compared with cardiovascular diseases. Fortunately, recent studies and indeed position papers from relevant organizations indicate that mortality rate alone is a insufficient indicator for the suffering, caused

by a disease and experienced by the patient as well as his or her social environment. The Global Burden of Disease (GBD) study was launched, by the WHO in 1996, to find a better indicator for this suffering [8]. A major result of the study was that unipolar depressive disorders are the forth leading cause of suffering, this amounts to 3.7% of total Disability Adjusted Life Years (DALYs). Furthermore, the study identified depression as one of the main reasons for Years Lived with Disability (YLD), causing about 10.7% of the total YLD. In 2002, the WHO updated the report with new data, such as epidemiological estimates [9].

Seasonal Affective Disorder (SAD) is a subgroup of major depression [10], it is diagnosed when patients, who have normal mental health throughout most of the year, experience depressive symptoms during specific times of the year [11]. Studies show that light therapy can provide a potent alternative or adjunct to antidepressant drug treatment which softens the effect of SAD. Tam et al. state found that 50% of patients with SAD experience remission within a few days after starting the light therapy [12]. In a similar study, Terman et al. found an even higher success rate of 67% [13]. These successes in light treatment of major depression with [14] or without seasonal pattern [15, 16] underscore the need for biomedical systems which provide this type of treatment to patients.

This paper describes the formal and model driven design of LUXAMET, a bright light therapy system for seasonal depression. We show that the design methodology was key in improving the system such that both speed and reliability requirements could be met. To be specific, the system features shared memory for high speed data transfer, a technique which can introduce race conditions and system instabilities, if it is not implemented correctly. In order to ensure formal correctness, the system creation followed the systems engineering meta model which structures the design process into different phases. In the specification phase, we have used a formal CSP||B model to define the system functionality. The model enabled us to control and ultimately rule out the dangers which are inherent to shared memory data transfer. With a guiding Communicating Sequential Processes (CSP) model we prove the absence of deadlock and livelock. With the underlying B-machine we prove the absence of invariant violations. These proves, together with the similarity between the specification model and the implementation code, make us confident that the implemented system is indeed a reliable tool which can help patients during seasonal depression.

The structure of the paper follows the systems engineering meta model. Section 2 introduces requirements capturing, specification refinement and implementation. Section 3 presents the results of our design effort. The first set of results comes from measurements which confirm that the implemented system complies with the requirements. Use and failure case testing yields another set of results with which we support our claim that the implementation behaves like the specification. These results are put into a wider perspective when we compare them with other biomedical system designs in Section 4. We conclude this paper with Section 5.

2. Materials and methods

Formal and model driven design methods deliver reliable and dependable biomedical systems which work in real-world scenarios [17]. This section introduces the design steps of requirements capturing, specification refinement and implementation. With these design steps we control race conditions and memory leaks which can arise in the proposed system.

2.1. *Requirements*

The requirements were captured during stakeholder discussions. These discussions took place after it was agreed that there is a need and a commercial case for building a therapeutic system for SAD and major depression patients. Figure 1 shows an overview diagram of this therapeutic system. On this abstract level, the LUXAMET system consists of a PC based control program, an embedded Microprocessor Control Unit (MCU) and Light Emitting Diode (LED) panels which deliver the light necessary to achieve therapeutic effects. The communication between the embedded MCU is established via Universal Asynchronous Receiver/Transmitter (UART) over Universal Serial Bus (USB). The brightness of the LED panels is controlled via Pulse Width Modulation (PWM).

Fig. 1. LUXAMET overview block diagram.

The individual components must meet the following requirements:

- Cost effective. The implemented bright light system must meet all the requirements as cost effective as possible.
- Reliable. The system is used adjacent tool to reduce the symptoms of seasonal depression. Therefore, patients have to trust the system, i.e. it should work according to specification for a long period of time.
- PC based. That means the system is controlled via a Graphical User Interface (GUI) on a PC.
- Four channels. It must be possible to connect up to four LED panels to one controller unit.
- Intensity mode. Set the brightness of the LED panels. This intensity control should be done with PWM which has a period of not more then 100 µs and a resolution of at least 255 steps.
- Trigger mode. Use buttons to switch the LED panels on and off.

2.2. *Specification*

In this specification section we focus on the formal model which defines the MCU functionality. Formal models help the designer to find fault states in the system state space [18]. On a very basic level, these fault states can be detected by formalizing the design, because formal models are more abstract than the implementation. This abstraction makes the models much more discussable and accessible even for non-specialists [19, 20]. It also prevents designers from introducing a fault state due to a lack of overview. The real advantage of formal models comes from the fact that mechanized model checking can be applied. There are two fundamentally different methods for this mechanized model checking. The first one is theorem proofing and the other one is state space checking. These two distinct methods sparked a rich variety of different tools for a range of applications [21].

During the design of LUXAMET, we have used a CSP||B [22, 23] model to capture the MCU functionality. The next section introduces the guiding CSP model. Section 2.2.2 discusses the Abstract Machine Language (AML) which defines the B-machine that models a shared memory data transfer.

2.2.1. *Guiding CSP model*

CSP is a process algebra which was conceived by Hoare in 1978 [24, 25]. In their practical paper Cichocki and J. Górski show how CSP and the associated model checking tool FDR can be used to support failure modes and effects analysis of software intensive systems [26]. Alur and Henzinger show the importance of logic-based and automata-based languages and techniques for the specification and verification of real-time systems [27]. Lightweight formal methods promise to yield modest analysis results in an extremely rapid manner [28]. The paper by Easterbrook et al. describes three case studies in the application of lightweight formal methods to requirements modeling for spacecraft fault protection systems. They conclude that the benefits gained from early modeling of unstable requirements more than outweigh the effort needed to maintain multiple representations [29]. We have selected CSP as formal modeling method because it captures very well the parallel nature of the system functionality [30].

The process network, shown in Figure 2, gives just an overview of the possible communication between the processes. However, such diagrams are by no means detailed enough to make profound statements about system stability and reliability. To proof these properties, it is necessary to translate the process network into algebraic equations. This process starts by realizing that the two processes *UART_RX* and *UART_TX* abstract the UART functionality of the MCU. The *PROTOCOL* interprets the commands received from the PC in order to pass on appropriate information to the *CNT* process. The *CNT* process itself assembles the low level commands for the PWM. In trigger mode, these commands depend on the *TRIGGER* input. The *FEEDER* process fills one side of a double buffer with the

low level commands from the *CNT* process. Once this buffer is full it initiates a buffer change. Both buffer and toggle variables are shared between *FEEDER* and *PWD* processes. There is no synchronized communication between *PWM* and the rest of the network, therefore the network diagram in Figure 2 shows *PWM* as an unconnected process.

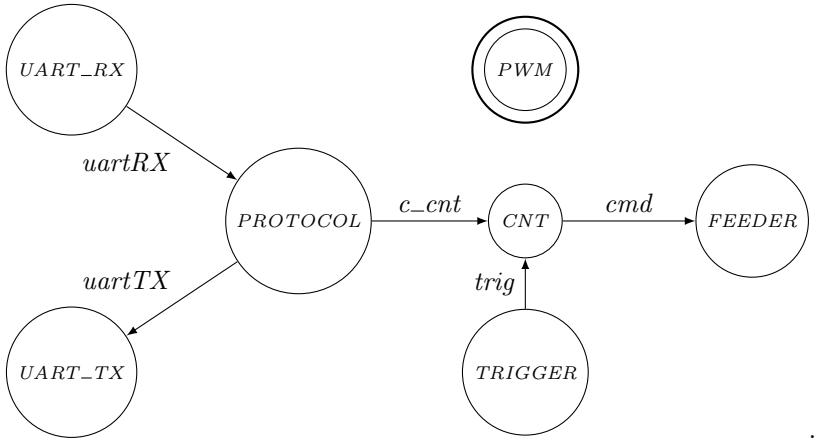

Fig. 2. *MAIN* network architecture of LUXAMET

The first step, in the creation of a CSP model, is to define data types. For the *command* data type, t indicates trigger mode and i indicates intensity mode.

$$\begin{array}{l} \textbf{datatype}\ command = t \mid i \\ \textbf{datatype}\ DchanTX = ack \end{array} \tag{1}$$

Next, all the channels, shown in Figure 2, are defined. The construct $command.\{0..1\}$ expresses that a channel communicates the *command* as well an intensity value (0 or 1) and these two messages are separated by a dot.

$$\begin{array}{l} \textbf{channel}\ c\_cnt, uartRX, chanRX : command.\{0..1\} \\ \textbf{channel}\ uartTX, chanTX : DchanTX \\ \textbf{channel}\ cmd : \{0..1\} \\ \textbf{channel}\ tick, button, trig \end{array} \tag{2}$$

The algebraic expressions for both *UART_RX* and *UART_TX* are straight forward: Whatever message comes in is transferred to the output channel.

$$\begin{array}{l} UART\_RX = uartRX?x \rightarrow chanRX!x \rightarrow UART\_RX \\ UART\_TX = chanTX?x \rightarrow uartTX!x \rightarrow UART\_TX \end{array} \tag{3}$$

The *PROTOCOL* process acknowledges each received message $(x.y)$ before the

message is send on to the *c_cnt* channel.

$$\begin{aligned} PROTOCOL = {} & chanRX?x.y \rightarrow chanTX!ack \\ & \rightarrow c\_cnt!x.y \rightarrow PROTOCOL \end{aligned} \tag{4}$$

The *CNT* process determines whether the received command (x) is trigger (t) or intensity (i) mode and branches accordingly to *T_CNT* or *I_CNT_OFF*. The $\Box$ construct ensures that the *CNT* process consumes a *trig* event without taking action.

$$\begin{aligned} & CNT = \\ & \quad c\_cnt?x.y \rightarrow (\textbf{if } x == t \textbf{ then } T\_CNT\_OFF(y) \textbf{ else } I\_CNT(y)) \\ & \quad \Box trig \rightarrow CNT \end{aligned} \tag{5}$$

The *I_CNT* process sends out the intensity value (y) over *cmd* channel before the process behaves like *CNT* again.

$$I\_CNT(y) = cmd!y \rightarrow CNT \tag{6}$$

The *T_CNT_OFF* process sets the intensity value to zero by sending *cmd*!0, this effectively switches off the LED panels. Once the trigger event *trig* is received, the process behaves like *T_CNT_ON*. Furthermore, the *T_CNT_OFF* process has to take care of any new incoming commands from *c_cnt* channel. The external choice construct $\Box$ ensures that the process reacts to messages from both *trig* and *c_cnt* channels.

$$\begin{aligned} & T\_CNT\_OFF(y) = cmd!0 \rightarrow (\\ & \quad c\_cnt?x.y \rightarrow (\textbf{if } x == t \textbf{ then } T\_CNT\_OFF(y) \textbf{ else } I\_CNT(y)) \\ & \quad \Box trig \rightarrow T\_CNT\_ON(y) \\ & \quad) \end{aligned} \tag{7}$$

The *T_CNT_ON* process sets the intensity value to the desired value y by sending this value over the *cmd* channel, this effectively switches on the LED panels. Once another trigger event *trig* is received, the process behaves like *T_CNT_OFF*. Furthermore, the *T_CNT_OFF* process has to take care of any new incoming commands from *c_cnt* channel.

$$\begin{aligned} & T\_CNT\_ON(y) = cmd!y \rightarrow (\\ & \quad c\_cnt?x.y \rightarrow (\textbf{if } x == t \textbf{ then } T\_CNT\_OFF(y) \textbf{ else } I\_CNT(y)) \\ & \quad \Box trig \rightarrow T\_CNT\_OFF(y) \\ & \quad) \end{aligned} \tag{8}$$

The functionality of the *TRIGGER* process is straight forward, a *button* event is converted into a *trig* event.

$$TRIGGER = button \rightarrow trig \rightarrow TRIGGER \tag{9}$$

The *FEEDER* process is a placeholder process which just consumes all *cmd* messages.

$$FEEDER = cmd?x \rightarrow FEEDER \tag{10}$$

The *PWM* simply exhibits a sequence of *tick* events.

$$PWM = tick \rightarrow PWM \tag{11}$$

The *UART* process combines *UART_RX*, *UART_TX* and *PROTOCOL*. *UART_RX* and *UART_TX* do not share a communication channel, they make progress independently from each other, therefore they are combined using the $|||$ statement. However, the combined processes share all messages which are send over the channels *chanRX* and *chanTX* with the *PROTOCOL* process. Therefore, the parallel statement $||$ was used to combine the processes.

$$UART = \Big(UART\_RX \,|||\, UART\_TX \Big) \underset{\{|chanRX, chanTX|\}}{||} PROTOCOL \tag{12}$$

The *MAIN* process, shown in Figure 2, combines *PWM*, *TRIGGER*, *UART*, *CNT* and *FEEDER*. *TRIGGER* and *UART* do not share a communication channel, therefore they are combined with the interleaved statement. However, this combination shares all events over the channels *c_cnt* and *trig* with the *CNT* process. The *FEEDER* process is connected via the *cmd* channel, this is modeled in the second line of Equation 13. The *PWM* process can make progress independently from the rest of the network, therefore the interleaved operator was used to ad this process.

$$\begin{aligned} MAIN = {} & PWM \\ & ||| \left(\begin{array}{l} \Big(\Big(TRIGGER \,|||\, UART \Big) \,||_{\{|c\_cnt, trig|\}}\, CNT \Big) \\ ||_{\{|cmd|\}}\, FEEDER \end{array} \right) \end{aligned} \tag{13}$$

The *MAIN* process consists of seven processes and it is modeled with just 13 equations. The main purpose of this straight forward process network is to guide a B-machine which models the shared memory data communication. This guidance is based on events, which are communicated over channels when the CSP model makes progress.

2.2.2. *B-machine specification*

The B-method is a state-based method, which was developed by Abrial [31,32], for specifying, designing and coding software systems. It is based on Zermelo-Fraenkel set theory with the axiom of choice [33]. Sets are used for data modeling, "Generalised Substitutions" are used to describe state modications which model executing the system. The B-method uses refinement calculus to relate modes at varying levels of abstraction, furthermore it is equipped with a number of structuring mechanisms (machine, renement, implementation) which can be used to organize a development.

The B-machine models the shared memory data transfer for the bright light controller. In this case, shared memory means that the *FEEDER* process communicates with the *PWM* over two global variables: buffer and toggle. In AML all variables are global, i.e. they can be manipulated by all operations. Hence, the setup of the

SHARED_MEMORY B-machine is straight forward.

MACHINE SHARED_MEMORY
CONSTANTS
MAX
PROPERTIES
MAX = 2
VARIABLES
output, buff, toggle
INVARIANT
output ∈ 0 .. MAX ∧ buff ∈ $[\![0,1]\!]$ → (0 .. 1) ∧ toggle ∈ $[\![0,1]\!]$
INITIALISATION
output := 0 ‖ buff := {0 ↦ 0, 1 ↦ 0} ‖ toggle := 0

The B-machine has only two operations: tick and cmd(command). These operations are executed whenever there is a corresponding event from the guiding CSP model. When there is a new *cmd* event, the cmd operation will flip the boolean value of the toggle variable assign a new brightness value to buff(toggle). When there is a new *tick* event, the tick operation will execute. When the buffer(toggle) is 0, output will count upwards. When the buffer(toggle) is 1, output will count downwards.

OPERATIONS
tick =
PRE output ∈ 0 .. MAX ∧ toggle ∈ 0 .. 1
THEN
IF buff(toggle) = 0 **THEN** output := (output + 1) mod (MAX + 1)
ELSE output := (output + MAX) mod (MAX + 1) **END**
END;
cmd(command) =
PRE command ∈ 0 .. 1
THEN
toggle := (toggle + 1) mod 2;
buff(toggle) := command
END
END

The B-machine execute as follows: Whenever there is a new command, i.e. a new output value is set, the buffer, which is not used in the tick operation, is manipulated. After all changes to this buffer are done, the buffer is switched with the toggle variable. Having two buffers means that the tick operation can use one buffer to generate the PWM signals for the LED panels and the cmd operation can manipulate the other buffer without interfering with the PWM generation. This effectively implements a double buffer scheme where *PWM* and *FEEDER* each 'own' one buffer and ownership is switched when a new command has been processed.

2.3. *Implementation*

In accordance with the systems engineering design methodology [34,35], this section describes the translation of the formal CSP||B model into an implementation. Such a translation is always specific to the targeted MCU architecture. For this study, we have selected the XS1-G4 processor from XMOS [36] as MCU. This particular architecture supports two high level languages with which the desired functionality can be implemented. The first language is the XMOS-originated 'XC' language [37] and the second language is the well known 'C' language [38]. 'XC' language was used to implement the CSP specification and 'C' was used to implement the B specification.

The first step to realize the system was to implement an XThread network, which reflects the guiding CSP model, as shown in Section 2.2.1. Figure 3 shows the top level XThread setup of the implemented system. This diagram follows closely the specification model shown in Figure 2. Each process, in the guiding CSP model, has been implemented as an XThread. Similarly, each CSP channel has been implemented as XMOS channel. Apart from XThreads and XMOS channels, Figure 3 also shows two buffers `buff(0)` and `buff(1)`. These two buffers facilitate the shared memory data transfer between `feeder` and `pwm`. Hence, this part of Figure 3 reflects the B-machine.

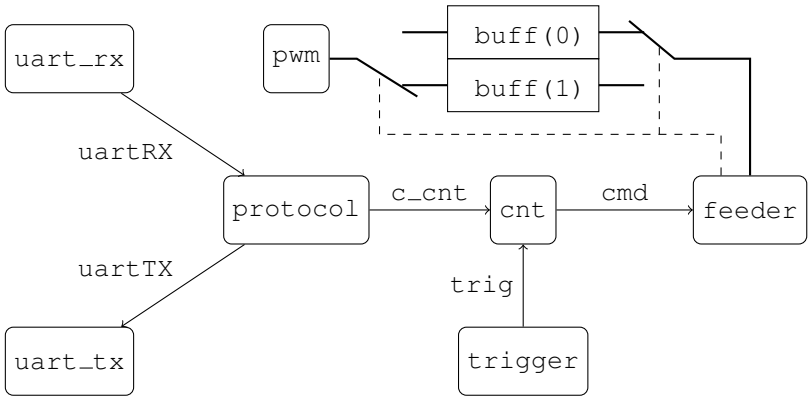

Fig. 3. XThread setup of the LUXAMET implementation

The XThread setup, shown in Figure 3, is implemented in the XC function `main`. Listing 1 shows the source code of this function. The commands in Line 2–5 declare the channels `uartRX`, `uartTX`, `c_cnt`, `cmd` and `trig`. The **`par`** statement, stretching from Line 6 till Line 21, instructs the compiler to interpret all function calls, within the scope of this command, as XThreads. The **`on stdcore`** command instructs the compiler to map the XThread to a specific XCore. For example, the commands in Line 16 ensure that the `protocol` XThread executes on XCore 0. In

contrast, both pwm and feeder are on XCore 1. These two XThreads were mapped to XCore 1 because the PWM signals, which were used for the PWM measurements, come from XCore 1. Furthermore, pwm and feeder share memory, therefore these two XThreads must execute on the same XCore. Each XMOS channel establishes a duplex communication between two XThreads. For example, cnt is connected to the feeder XThread via the cmd channel. The setup for uart_rx and uart_tx is marginally more sophisticated, because the UART transfer needs special configuration parameters.

```
int main() {
  chan chanTX, chanRX;
  chan c_cnt;
  chan cmd;
  chan trig;
  par {
    on stdcore[0] : {
      unsigned char tx_buffer[64];
      unsigned char rx_buffer[64];
      tx <: 1;
      par {
        uart_rx(rx, rx_buffer, ARRAY_SIZE(rx_buffer),
            BAUD_RATE, 8, UART_TX_PARITY_NONE, 1, chanRX);
        uart_tx(tx, tx_buffer, ARRAY_SIZE(tx_buffer),
            BAUD_RATE, 8, UART_TX_PARITY_NONE, 1, chanTX);
      }
    }
    on stdcore[0] : protocol(chanTX, chanRX, c_cnt);
    on stdcore[0] : trigger(trig);
    on stdcore[0] : cnt(c_cnt, cmd, trig);
    on stdcore[1] : pwm();
    on stdcore[1] : feeder(cmd);
  }
  return 0;
}
```

The second and final listing, which is discussed in this paper, is concerned with the source code that writes the PWM values onto the output port of XCore 1. Listing 2 shows the C source code for the pwm process. The function starts by defining the character variable x[a]. The **while** structure implements an infinite loop which feeds PWM values onto the output port. The function in Line 4 is the heart of this listing and indeed the heart of the shared memory transfer mechanism. outBuff is a pointer which points to a character [x] that belongs to either buff(0) or buff(1). The pointer address changes whenever a buffer switch occurs and this change is initiated by the the feeder XThread. This implements the pwm part

[a]The variable can have a value from 0 to 255

of the double buffer scheme shown in Figure 3. `pushOut(.)` is a system function which puts the values on a predefined output.

```
void pwm(){
  unsigned char x = 0;
    while(1){
      pushOut(outBuff[x]);
      x++;
  }
}
```

3. Results

This section documents the tests which were conducted to verify that the implementation behaves like the specification. In an initial test, we have established that the PWM signals are output. Figure 4 shows two measurements of PWM signals. The $pwm_{10}(t)$ signal is 10 time intervals high and 245 time intervals low. The $pwm_{128}(t)$ signal is 128 time intervals high and 127 time intervals low. A time interval was selected to be:

$$\text{time interval} = \frac{1}{100\ \text{MHz}} \times 32 = 0.32\,\mu\text{s} \tag{14}$$

The factor 32 was chosen to divide the 100 MHz system clock, because the PWM signal levels have to be produced according to the commands send by the PC. To be specific, the loop, shown in Listing 2, takes 21 XMOS assembly instructions for one cycle, i.e. with a 100 MHz system clock the maximum speed, with which a sample can be produced, is once every $1/100\ \text{MHz} \times 21 = 0.21\,\mu\text{s}$. The XMOS architecture requires the division factor for output clocks to be of the form 2^n where $n \in \mathbb{Z}^+$, hence we have chosen the nearest clock scaling factor which is both greater than 21 and allowed by the system.

The time interval is important, because it sets the period length of one PWM cycle. In this case, one these cycles takes 255 time intervals or 81.6 µs.

3.1. *Use and Failure Case Testing*

Use and failure case testing instills trust in the implemented system, because these tests establish that the implementation has the same functionality as the specification. Use cases test the normal or desired functionality of the implemented system. In the first use case test we have programed the GUI to steep up the intensity of all four PWM channels from 0 to 255 one step every 5 s. The results were observed on the LED panels and the 4 different PWM signals were measured with an oscilloscope. The LED panels slowly lit up and the PWM signals were exactly is we expected them. This test has verified the communication with the PC and the PWM signal generation. The second use case test is concerned with the trigger functionality. We have verified that the LED panels light up and switch off in accordance

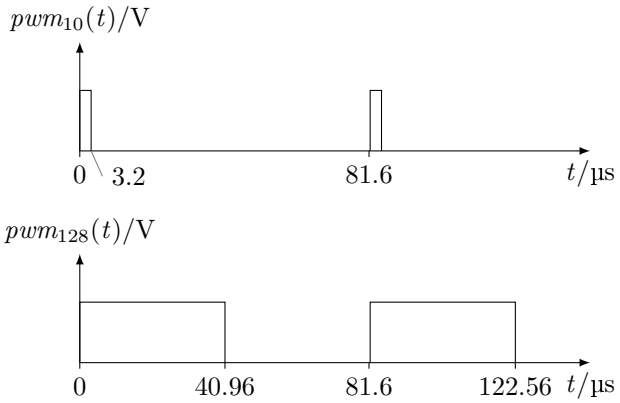

Fig. 4. PWM signals for intesity 10 ($pwm_{10}(t)$) and for intensity 128 ($pwm_{128}(t)$)

.

with the buttons pressed. With the GUI it was possible to set the brightness of the panels as stated in the requirements.

Failure case testing was conducted by decreasing the step interval of the first use case test. To be specific, we have implemented the intensity increments with a hard loop in the PC system. We found that the MCU could cope with the continuous data stream from the PC system and a change of intensity took about 1 ms. This timing was determined by the relatively slow baud-rate of the UART connection. However, the important point, which was established trough this failure case testing, is that the embedded system could cope with the highest rate of commands from the PC system.

4. Discussion

Recent developments in biomedical engineering provided us with new mental disorder diagnosis support systems [39–42]. These systems detect symptoms of depression, autism and alcoholism in an early stage of the disease when therapeutic methods, such as bright light therapy, is most effective. Therefore, the need for safe, reliable and functional light therapy systems will increase. With LUXAMET we address that need with a formal and model driven system design.

Bright light therapy for SAD has been investigated and applied for over 20 years [43]. Physicians and clinicians are increasingly confident that bright light therapy is a potent, specifically active, non-pharmaceutical treatment modality [44]. In the past, the studies assessing the efficacy of light therapy in nonseasonal depression have been controversial. While some controlled studies reported significant improvements [45,46] others failed to do so [10,47]. Research by Kripke suggests that, at least some non-seasonal depressives respond to light therapy, however the improvement might

be less than for SAD [48].

Having a good medical agreement on the therapy, the biomedical engineers need to step in and build the therapy system. The design must follow a well thought out process in order to create reliable and safe systems for humans to use [49–51]. Therefore, we have extended the systems engineering methodology with formal methods in the specification phase [52]. The beauty of this approach is that we can fill all the different design phases with meaning. In the need definition phase we have discussed the medical evidence for bright light therapy systems. This need was translated into a requirements list which contained all the system properties. Once this list was compiled, we refined the information into a formal CSP||B model which serves as specification [53]. Based on this formal model we prove the absence of deadlock and livelock as well as the absence of invariant violations. Having these proves is an important achievement, because they document that it is possible to create the specified functionality in a reliable and safe way. The next step in the systems engineering design process is implementation. In our case, the implementation was a translation of the formal model into source code for the MCU target. We conclude the description of the design process with use and fault case testing which establishes that the implementation behaves like the specification. Hence, we are fairly confident that we have created a reliable bright light therapy system which is safe for humans to use. The discussion of the design process does not include considerations about product life-cycle support and considerations about sustainability of both design and system. These limitations need to be addressed in another context where the focus is more on an holistic approach to design.

With the current design we had to solve practical engineering problems as we progressed with the systems engineering design methodology. The biggest engineering problem, we were phasing, was caused by standard data transfer mechanisms which were not fast enough to create the required PWM period length of 100 µs. To solve the problem, we had to abandon the slow, but safe, channel communication and replace it with a shared memory approach. Shared memory is the fastest way of transferring data between threads. However, shared memory, like any shared resource, has the potential to introduce race conditions because it can be accessed by multiple entities [54, 55]. The general solution for this problem is to synchronize access to the shared resource, in this case shared memory [56]. Unfortunately, synchronization techniques cost runtime, i.e. they are slow. In case of the PWM process, taking care of access synchronization during runtime impacts on the performance. To be specific, the PWM, synchronized via channels, was around 4.6 times slower as the PWM with shared memory. That means, using synchronized data transfer, the PWM period was longer than 375 µs.

To solve the problem of shared memory is a formidable challenge, because the selected MCU does not provide hardware support for detecting race conditions and memory leaks. We have solved the problem with formal and model driven design. Conceptually, we have shifted the task of ensuring that no race conditions

are possible from the run-time to design time. In other words, there is no need for the implemented system to check and establish synchronized data transfer, because with the CSP||B model we could prove the absence of race conditions during the specification phase of the design.

The formal CSP||B model is composed from just 13 equations in the CSP part and 25 lines of AML code for the B-model. That means, the formal model is not opaque, on the contrary, it is short and abstract. We regard this as an advantage, because a formal model should be a medium for communication and discussion amongst stake holders in the project. Larger models, which use hundreds or even thousands of lines of formal description, are not understandable for non-experts. The drawback of this abstraction comes from the fact that the model does not contain enough information to cover the complete implementation. For example, the formal model implies that the buffer switch between `buff(1)` and `buff(1)` must be atomic[b] but it does not indicate how to establish or ensure this requirement in the implementation.

In general, electronic processing is used extensively in biomedical engineering [57–59]. Therefore, formal and model driven design can benefit a wide range of different application areas. For example, breast imaging for cancer detection relies on sophisticated image processing algorithms [60, 61]. Similarly, Computer-Aided Diagnosis (CAD) systems for plaque [62–65], cardiac disease [66, 67] and diabetes [68, 69] relay also heavily on computerized processing. Hence, these CAD systems stand to benefit from formal and model driven biomedical systems design, because the design methodology helps us to realize systemic safety and reliability.

5. Conclusion

This paper describes the formal and model driven design process of LUXAMET, a bright light therapy system. The process starts with need definition, where we collect the medical evidence that bright light can relieve the symptoms of seasonal depression. Once the need for a physical problem solution was established, the next step is requirement capturing. The requirements answer the question: What system do we want to build? In this case the system is based on a micro-controller which receives commands from a PC and which controls LED panels via PWM. The specification phase refines the requirements into a formal CSP||B model. This model answers the question: How do we build the system? During the implementation phase we have translated the formal model into source code for the target micro-controller. Tests and measurements confirmed that this translation was indeed a truthful representation of the specification.

With the design of LUXAMET we show that formal and model driven design is an easy way to create biomedical systems which are safe and reliable. The design

[b]One processor instruction.

process was governed by the systems engineering methodology. During the design process we have advanced scientific understanding by modeling a shared memory system with CSP||B. The benefits of this model come from a deeper insight into the system, i.e. we learned how to tackle the dangers of shared memory during design time and not during runtime. This conceptional shift enabled us to speed up the system by 4.6 times and only through this speedup we were able to meet all the requirements. Furthermore, having formal proves, instilled confidence that the proposed system will work as a real world problem solution. This confidence was further boosted by the fact that the formal model corresponded very well with the implementation.

We feel, contrary to wide spread believe[c], that formal and model driven design, within a well thought out design strategy, helped us to focus on the innovations necessary to build a physical solution for the medical problem of seasonal depression. Having made this experience, we predict that formal and model driven design will get more and more important for biomedical engineering. Another reason to support this statement comes from the area of reliability and quality of service. We have used this technique to model and ultimately overcome the dangers of shared memory. Such dangers lurk in most electronic support systems and for biomedical support systems these dangers might directly endanger human well being or even human life. Formal models can help to understand critical sections of the system functionality and they are abstract and clear enough to invite a wide audience to participate in the solution of these problems. Hence, the proposed formal and model driven design methodology can lead to more reliable systems which serve human needs.

Appendix A. Acronyms

AML Abstract Machine Language
CAD Computer-Aided Diagnosis
CSP Communicating Sequential Processes
DALY Disability Adjusted Life Year
GBD Global Burden of Disease
GUI Graphical User Interface
LED Light Emitting Diode
MCU Microprocessor Control Unit
PWM Pulse Width Modulation
SAD Seasonal Affective Disorder
UART Universal Asynchronous Receiver/Transmitter
USB Universal Serial Bus
WHO World Health Organization
YLD Years Lived with Disability

[c]Think outside the box.

Appendix B. References

References

1. W. C. Drevets, J. L. Price, and M. L. Furey, "Brain structural and functional abnormalities in mood disorders: implications for neurocircuitry models of depression," *Brain structure and function*, vol. 213, no. 1-2, pp. 93–118, 2008.
2. J. Spijker, R. de Graaf, R. V. Bijl, A. T. F. Beekman, J. Ormel, and W. A. Nolen, "Functional disability and depression in the general population. results from the netherlands mental health survey and incidence study (nemesis)," *Acta Psychiatrica Scandinavica*, vol. 110, no. 3, pp. 208–214, 2004.
3. T. B. Üstün, J. L. Ayuso-Mateos, S. Chatterji, C. Mathers, and C. J. L. Murray, "Global burden of depressive disorders in the year 2000," *The British Journal of Psychiatry*, vol. 184, no. 5, pp. 386–392, 2004.
4. O. Faust, P. C. A. Ang, S. D. Puthankattil, and P. K. Joseph, "Depression diagnosis support system based on eeg signal entropies," *Journal of Mechanics in Medicine and Biology*, vol. 14, no. 03, 2014.
5. C. J. L. Murray and A. D. Lopez, "Evidence-based health policy–lessons from the global burden of disease study," *Science*, vol. 274, no. 5288, pp. 740–743, 1996.
6. ——, "Alternative projections of mortality and disability by cause 1990–2020: Global burden of disease study," *The Lancet*, vol. 349, no. 9064, pp. 1498–1504, 1997.
7. E. Bromet, L. Andrade, I. Hwang, N. Sampson, J. Alonso, G. de Girolamo, R. de Graaf, K. Demyttenaere, C. Hu, N. Iwata, A. Karam, J. Kaur, S. Kostyuchenko, J.-P. Lepine, D. Levinson, H. Matschinger, M. Mora, M. Browne, J. Posada-Villa, M. Viana, D. Williams, and R. Kessler, "Cross-national epidemiology of dsm-iv major depressive episode," *BMC Medicine*, vol. 9, no. 1, pp. 1–16, 2011.
8. C. J. L. Murray, A. D. Lopez, World Health Organization, World Bank, and Harvard School of Public Health, *Global health statistics: a compendium of incidence, prevalence, and mortality estimates for over 200 conditions*, ser. Global burden of disease and injury series. Published by the Harvard School of Public Health on behalf of the World Health Organization and the World Bank, 1996.
9. World Health Organization, "World health report 2002. reducing risks, promoting healthy life," 2002, geneva, Switzerland.
10. A. Magnusson and D. Boivin, "Seasonal affective disorder: An overview," *Chronobiology International*, vol. 20, no. 2, pp. 189–207, 2003.
11. S. J. Lurie, B. Gawinski, D. Pierce, and S. J. Rousseau, "Seasonal affective disorder," *American Family Physician*, vol. 74, no. 9, pp. 1521–1524, 2006.
12. E. M. Tam, R. W. Law, and A. J. Levitt, "Treatment of seasonal affective disorder: a review," *Canadian Journal of Psychiatry*, vol. 40, pp. 457–466, 1995.
13. M. Terman, J. S. Terman, F. M. Quitkin, P. J. McGrath, J. W. Stewart, and B. Rafferty, "Light therapy for seasonal affective disorder. a review of efficacy." *Neuropsychopharmacology*, vol. 2, no. 1, pp. 1–22, 1989.
14. K. Kräuchi, A. Wirz-Justice, and P. Graw, "High intake of sweets late in the day predicts a rapid and persistent response to light therapy in winter depression," *Psychiatry Research*, vol. 46, no. 2, pp. 107–117, 1993.
15. A. Tuunainen, D. F. Kripke, and T. Endo, *Light therapy for nonseasonal depression*. John Wiley & Sons, Ltd, 2004.
16. M. Fritzsche, R. Heller, H. Hill, and H. Kick, "Sleep deprivation as a predictor of response to light therapy in major depression," *Journal of Affective Disorders*, vol. 62, no. 3, pp. 207–215, 2001.
17. O. Faust, U. R. Acharya, and T. Tamura, "Formal design methods for reliable

computer-aided diagnosis: a review," *Biomedical Engineering, IEEE Reviews in*, vol. 5, pp. 15–28, 2012.
18. O. Faust, U. R. Acharya, B. H. C. Sputh, and L. C. Min, "Systems engineering principles for the design of biomedical signal processing systems," *Computer Methods and Programs in Biomedicine*, vol. 102, no. 3, pp. 267–276, 2011.
19. R. W. Butler and G. B. Finelli, "The infeasibility of quantifying the reliability of life-critical real-time software," *IEEE Trans. Softw. Eng.*, vol. 19, no. 1, pp. 3–12, 1993.
20. M. C. B. Alves, D. Drusinsky, J. B. Michael, and M.-T. Shing, "End-to-end formal specification, validation, and verification process: A case study of space flight software," *Systems Journal, IEEE*, vol. 7, no. 4, pp. 632–641, 2013.
21. E. M. Clarke and J. M. Wing, "Formal methods: State of the art and future directions," *ACM Computing Surveys*, vol. 28, pp. 626–643, 1996.
22. M. Leuschel and M. Butler, "Prob: an automated analysis toolset for the b method," *International Journal on Software Tools for Technology Transfer*, vol. 10, no. 2, pp. 185–203, Feb. 2008.
23. ——, "Combining csp and b for specification and property verification," in *Formal Methods*, I. Hayes, A. Tarlecki, and J. Fitzgerald, Eds. Springer-Verlag, LNCS 3582, January 2005, pp. 221–236.
24. C. A. R. Hoare, "Communicating sequential processes," *Communications of the ACM*, vol. 21, no. 8, pp. 666–677, 1978.
25. ——, *Communicating Sequential Processes*, 1st ed. Upper Saddle River, New Jersey 07485 United States of America: Prentice Hall, 1978.
26. T. Cichocki and J. Górski, "Formal support for fault modelling and analysis," in *SAFECOMP*, 2001, pp. 190–199.
27. R. Alur and T. A. Henzinger, "Logics and models of real time: A survey," in *Proceedings of the Real-Time: Theory in Practice, REX Workshop*. London, UK: Springer-Verlag, 1992, pp. 74–106.
28. M. S. Feather, "Rapid application of lightweight formal methods for consistency analyses," *IEEE Trans. Softw. Eng.*, vol. 24, no. 11, pp. 949–959, 1998.
29. S. Easterbrook, R. Lutz, R. Covington, J. Kelly, Y. Ampo, and D. Hamilton, "Experiences using lightweight formal methods for requirements modeling," *IEEE Transactions on Software Engineering*, vol. 24, pp. 4–14, 1998.
30. B. H. C. Sputh, O. Faust, and A. R. Allen, "Portable csp based design for embedded multi-core systems," in *CPA 2006*, F. R. M. Barnes, J. M. Kerridge, and P. H. Welch, Eds., Sep. 2006, pp. 123–134.
31. J. Abrial, M. Lee, D. Neilson, P. Scharbach, and I. Srensen, "The b-method," in *VDM '91 Formal Software Development Methods*, ser. Lecture Notes in Computer Science, S. Prehn and H. Toetenel, Eds. Springer Berlin / Heidelberg, 1991, vol. 552, pp. 398–405.
32. J. R. Abrial, A. Hoare, and P. Chapron, *The B-Book: Assigning Programs to Meanings*. Cambridge University Press, 2005.
33. E. Zermelo, "Untersuchungen ber die grundlagen der mengenlehre. i," *Mathematische Annalen*, vol. 65, pp. 261–281, 1908.
34. A. Gorod, B. J. Sauser, and J. T. Boardman, "System-of-systems engineering management: A review of modern history and a path forward." *IEEE Systems Journal*, vol. 2, no. 4, pp. 484–499, 2008.
35. S. C. Ekpo and D. George, "A system engineering analysis of highly adaptive small satellites," *Systems Journal, IEEE*, vol. 7, no. 4, pp. 642–648, 2013.
36. *XS1-G4 512 BGA Datasheet*, 3rd ed., XMOS, Bristol, UK, 2009.

37. D. Watt, *Programming XC on XMOS Devices*. Published by XMOS Limited, 2009.
38. B. W. Kernighan and D. M. Ritchie, *The C Programming Language*. Prentice Hall Professional Technical Reference, 1988.
39. S. Bhat, U. R. Acharya, H. Adeli, G. M. Bairy, and A. Adeli, "Automated diagnosis of autism: in search of a mathematical marker," *Reviews in the Neurosciences*, vol. 25, no. 6, pp. 851–861, 2014.
40. ——, "Autism: cause factors, early diagnosis and therapies," *Reviews in the Neurosciences*, vol. 25, no. 6, pp. 841–850, 2014.
41. U. R. Acharya, S. Bhat, H. Adeli, A. Adeli *et al.*, "Computer-aided diagnosis of alcoholism-related eeg signals," *Epilepsy & Behavior*, vol. 41, pp. 257–263, 2014.
42. U. R. Acharya, V. Sudarshan, H. Adeli, J. Santhosh, J. Koh, and A. Adeli, "Computer-aided diagnosis of depression using eeg signals," *European neurology*, vol. 73, no. 5-6, pp. 329–336, 2015.
43. R. N. Golden, B. N. Gaynes, R. D. Ekstrom, R. M. Hamer, F. M. Jacobsen, T. Suppes, K. L. Wisner, and C. B. Nemeroff, "The efficacy of light therapy in the treatment of mood disorders: a review and meta-analysis of the evidence." *The American journal of psychiatry*, vol. 162, no. 4, pp. 656–662, 2005.
44. M. Terman and J. S. S. Terman, "Light therapy for seasonal and nonseasonal depression: efficacy, protocol, safety, and side effects." *CNS spectrums*, vol. 10, no. 8, Aug. 2005.
45. D. F. Kripke, S. Risch, and D. Janowsky, "Bright white light alleviates depression," *Psychiatry Research*, vol. 10, no. 2, pp. 105–112, 1983.
46. D. F. Kripke, D. J. Mullaney, M. R. Klauber, S. C. Risch, and J. C. Gillin, "Controlled trial of bright light for nonseasonal major depressive disorders," *Biological Psychiatry*, vol. 31, no. 2, pp. 119–134, 1992.
47. B. E. Thalén, B. F. Kjellman, L. Mørkrid, R. Wibom, and L. Wetterberg, "Light treatment in seasonal and nonseasonal depression," *Acta Psychiatrica Scandinavica*, vol. 91, no. 5, pp. 352–360, 1995.
48. D. F. Kripke, "Light treatment for nonseasonal depression: speed, efficacy, and combined treatment," *Journal of Affective Disorders*, vol. 49, no. 2, pp. 109–117, 1998.
49. O. Faust, R. Shetty, V. S. Sree, S. Acharya, U. R. Acharya, E. Ng, C. Poo, and J. S. Suri, "Towards the systematic development of medical networking technology," *Journal of Medical Systems*, pp. 1–15, 2010.
50. O. Faust, B. H. C. Sputh, U. R. Acharya, and A. R. Allen, "A pervasive design strategy for distributed health care system," *The Open Medical Informatics Journal*, vol. 2, pp. 58–69, 2008.
51. U. R. Acharya, O. Faust, D. N. Ghista, S. V. Sree, A. P. C. Alvin, S. Chattopadhyay, T.-C. Lim, E. Y.-K. Ng, and W. Yu, "A systems approach to cardiac health diagnosis," *Journal of Medical Imaging and Health Informatics*, vol. 3, no. 2, pp. 261–267, 2013.
52. O. Faust, U. Acharya, B. Sputh, and T. Tamura, "Design of a fault-tolerant decision-making system for biomedical applications." *Computer Methods in Biomechanics and Biomedical Engineering*, p. In press, 2012.
53. Z. Song, Z. Ji, J.-G. Maa, B. H. C. Sputh, U. R. Acharya, and O. Faust, "A systematic approach to embedded biomedical decision making," *Computer Methods and Programs in Biomedicine*, vol. –, p. In press, 2011.
54. K. Gharachorloo, D. Lenoski, J. Laudon, P. Gibbons, A. Gupta, and J. Hennessy, "Memory consistency and event ordering in scalable shared-memory multiprocessors," *ACM SIGARCH Computer Architecture News*, vol. 18, no. 3a, pp. 15–26, May 1990.
55. S. Adve and M. D. Hill, "A unified formalization of four shared-memory models," *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, no. 6, pp. 613–624,

1993.
56. R. H. B. Netzer and B. P. Miller, "What are race conditions?: Some issues and formalizations," *ACM Letters on Programming Languages and Systems*, vol. 1, no. 1, pp. 74–88, Mar. 1992.
57. O. Faust, U. R. Acharya, F. Molinari, S. Chattopadhyay, and T. Tamura, "Linear and non-linear analysis of cardiac health in diabetic subjects," *Biomedical Signal Processing and Control*, vol. 7, no. 3, pp. 295–302, 2012.
58. O. Faust, A. U. Rajendra, S. Krishnan, and M. Lim, "Analysis of cardiac signals using spatial filling index and time-frequency domain," *BioMedical Engineering OnLine*, vol. 3, no. 30, pp. 1–11, 2004.
59. O. Faust, U. R. Acharya, H. Adeli, and A. Adeli, "Wavelet-based eeg processing for computer-aided seizure detection and epilepsy diagnosis," *Seizure*, vol. 26, pp. 56–64, 2015.
60. S. V. Sree, E. Y.-K. Ng, R. U. Acharya, and O. Faust, "Breast imaging: a survey," *World journal of clinical oncology*, vol. 2, no. 4, p. 171, 2011.
61. U. R. Acharya, E. Y.-K. Ng, J.-H. Tan, and S. V. Sree, "Thermography based breast cancer detection using texture features and support vector machine," *Journal of medical systems*, vol. 36, no. 3, pp. 1503–1510, 2012.
62. U. R. Acharya, O. Faust, S. V. Sree, F. Molinari, L. Saba, A. Nicolaides, and J. S. Suri, "An accurate and generalized approach to plaque characterization in 346 carotid ultrasound scans," *Instrumentation and Measurement, IEEE Transactions on*, vol. 61, no. 4, pp. 1045–1053, 2012.
63. R. U. Acharya, O. Faust, A. P. C. Alvin, S. V. Sree, F. Molinari, L. Saba, A. Nicolaides, and J. S. Suri, "Symptomatic vs. asymptomatic plaque classification in carotid ultrasound," *Journal of medical systems*, vol. 36, no. 3, pp. 1861–1871, 2012.
64. U. Acharya, O. Faust, S. V. Sree, F. Molinari, R. Garberoglio, and J. Suri, "Cost-effective and non-invasive automated benign & malignant thyroid lesion classification in 3d contrast-enhanced ultrasound using combination of wavelets and textures: a class of thyroscan algorithms," *Technology in cancer research & treatment*, vol. 10, no. 4, pp. 371–380, 2011.
65. U. R. Acharya, O. Faust, S. V. Sree, F. Molinari, and J. S. Suri, "Thyroscreen system: high resolution ultrasound thyroid image characterization into benign and malignant classes using novel combination of texture and discrete wavelet transform," *Computer methods and programs in biomedicine*, vol. 107, no. 2, pp. 233–241, 2012.
66. U. R. Acharya, E. C.-P. Chua, O. Faust, T.-C. Lim, and L. F. B. Lim, "Automated detection of sleep apnea from electrocardiogram signals using nonlinear parameters," *Physiological measurement*, vol. 32, no. 3, p. 287, 2011.
67. U. R. Acharya, O. Faust, S. V. Sree, D. N. Ghista, S. Dua, P. Joseph, V. T. Ahamed, N. Janarthanan, and T. Tamura, "An integrated diabetic index using heart rate variability signal features for diagnosis of diabetes," *Computer methods in biomechanics and biomedical engineering*, vol. 16, no. 2, pp. 222–234, 2013.
68. U. R. Acharya, O. Faust, N. A. Kadri, J. S. Suri, and W. Yu, "Automated identification of normal and diabetes heart rate signals using nonlinear measures," *Computers in biology and medicine*, vol. 43, no. 10, pp. 1523–1529, 2013.
69. U. R. Acharya, E. Y.-K. Ng, J.-H. Tan, S. V. Sree, and K.-H. Ng, "An integrated index for the identification of diabetic retinopathy stages using texture parameters," *Journal of medical systems*, vol. 36, no. 3, pp. 2011–2020, 2012.