MECHATRONICS DESIGN AND VERIFICATION

Using VHDL-AMS to Bridge the Gap

Thomas Heurung
Mentor Graphics

Abstract: Mechanical parts that are controlled by electronic circuitry and software are commonly referred to as mechatronic components and systems. The design of such systems is classically divided into the design of the mechanical hardware, the electronic circuitry and the software - with little communication going on between the different engineering domains. This separation often results in communication problems based on incomplete or incorrect specifications and an inability to optimize the complete system instead of just certain aspects. Part of the reason for this failure is that although tools and methodologies are available that address the domain specific tasks, few tools are available that cross domain borders. This paper illustrates a VHDL-AMS simulation-based design methodology for software controlled, electro-mechanical components using an autonomous robot as an example. Starting with considerations for the overall architecture the design is successively augmented with implementation details that allow the verification of software control algorithms, the design and layout the electronics circuitry as well as the specification of parameters for the mechanical and electro-mechanical hardware. The example used is the design of a TekBot™, which is part of the platforms for learning™ concept created by the Oregon State University to teach students about digital logic and analog circuitry. VHDL-AMS based simulation is used to facilitate this task by making good decisions about the required components as well as designing and verifying the implementation. The same design is used for both PCB layout and as a testbench for the software for debugging the software algorithms. The final simulated design includes models of mechanical and electro-mechanical hardware, digital and analog electronic circuitry at behavioral and component levels as well as software executed on an instruction set model.

Keywords: Mechatronic Design, VHDL-AMS

1. INTRODUCTION

The TekBot™ is part of the platforms for learning™ concept created by Oregon State University to teach students about digital logic and analog circuitry. This paper uses the Tekbot (figure 1) design to illustrate a design and verification methodology for a mechatronic system – an autonomous automotive robot. VHDL-AMS, the Analog-Mixed-Signal extension (IEEE 1076.1) of the digital hardware description language VHDL, was used as the main modeling language. Starting with the drivetrain, the complete system was modeled and simulated with VHDL-AMS, moving from abstract, behavioral level to SPICE transistor level. The goals of simulating on different abstraction levels vary from estimating the power consumption to verifying the software control algorithm. The participating students modeled some of the components directly in VHDL-AMS, some by using a macro modeling approach and some by using SPICE components.

Figure 1. The Tekbot

Some of the requirements for the Tekbot are:

- Act autonomously
- Sort randomly distributed balls (black and orange) into color-coded corners of a 3m by 3m arena
- A blue diagonal between the colored corners is available for navigation

2. ARCHITECTURE

Based on the requirements, the design of the general architecture is straightforward. Sensors provide data about the environment to the control system that controls the movement of the Tekbot and activates the ball capturing system. At this stage of the design, the engineer needs to make decisions such as the kind of sensors to use for the color detection, how to find out the direction and color of the ball that's closest to the current position, and what propulsion system offers enough speed and control at the lowest cost. Design variants and their impact on cost, power consumption and performance need to be analyzed.

Budget constraints led to a sensor system consisting of an ultra-sonic rangefinder and three-color LEDs with photo-sensors. Using the LEDs to emit different colored light pulses and measuring the reflected light intensity allows identifying the color of an object. In the case of the Tekbot this is used to detect the color of a captured ball and the color of the current location in the arena.

Figure 2. Tekbot Architecture

For this level of analysis, the simulation needs to be fast and not very detailed. The models need to work with fairly abstract parameters, such as

energy consumption and delivered torque or the range of sonic sensors. This can be handled in VHDL-AMS with ideal, highly abstract models for the power supply, color sensors and range finder, control unit, motor controller, motor and wheel.

As the verification of the software algorithm for the control of the Tekbot is a central part of the simulation a realistic testbench is required. In this case the testbench is a model of the arena, including the distributed balls. Based on the current position and orientation of the Tekbot, the arena model returns the signal run-times of the range finder signals as well as the amount of reflected light for the optical sensors.

An inexpensive and fast way to find the best architecture is to simulate different configurations and design variants and automatically test them against the specification – this is virtually impossible with hardware prototypes.

3. COMPONENT DESIGN

Two motors for the “drivetrain” are part of the basic Tekbot chassis. Using different rotational velocities for the wheels control the direction of the movement and opposite turning directions allow for tight turns and turning on the spot. The simulation models serve as part of a meaningful testbench for the H-Bridge driver design as well as the verification of the software algorithm. If the motor is not already specified, this kind of simulation allows to identify the required motor parameters and to qualify a suitable motor. A model of an ideal motor (figure 3) is based on electrical and mechanical differential equations that can be found in standard literature. Together with a simple model for the wheels the electrical signals are translated into rotational movement of the motor shaft and then into a position of the wheels within the arena. This also provides an estimate about system properties such as speed, turning radius and power consumption.

```

architecture ideal of motor is
    quantity v across i through p to n;
    quantity w_shaft across torq_shaft through shaft to
rotational_velocity_ref;
    quantity w : real;
    quantity torq : real;
begin
    torq == -1.0*kt*i + d*w + j*w'dot;
    v == kt*w + i*r_int + l*i'dot;
    w_shaft == w/kg;
    torq_shaft == torq/kg;
end architecture ideal;

```

Figure 3. Motor Model

Using this multi-technology simulation provides the designer with a view onto the complete system with the right physical units and parameter values right from the datasheet of components. A simple behavioral model of the motor controller with the H-bridge driver (figure 4), the motors and the wheels is sufficient for early verification of the control software.

```

architecture ideal of motor_controller is
    quantity L_motor_v across L_motor_i through L_motor_p to
L_motor_n;
    quantity Vcc_c across Vcc to ref;
begin
    if L_en = '0' and L_dir = '1' use
        L_motor_v == motor_voltage;
    elsif L_en = '0' and L_dir = '0' use
        L_motor_v == -motor_voltage;
    else
        L_motor_v == 0.0;
    end use;
end architecture ideal;

```

Figure 4. Abstract Model of H-Bridge

VHDL-AMS supports moving from a high-level, abstract model to a more realistic implementation by using different model descriptions, so called architectures, for the same model. After the engineer has made

decisions about the configuration and components for the actual implementation of the H-Bridge, it could look like the transistor design in figure 5.

Correct tuning of the H-bridges requires detailed device behavior. Therefore it becomes important at this stage to be able to use SPICE models for simulation, as many manufacturers provide detailed and characterized SPICE models on their websites. With the use of a suitable tool, such as SystemVision, the schematic from figure 5 can then be used to layout a PCB board and get the data required for manufacturing the actual board.

Figure 5. Transistor-level H-Bridge

4. SOFTWARE VERIFICATION

Using different architectures in VHDL-AMS can be used to facilitate not only the hardware design, but also the software design. With all the hardware models on an abstract level, the implementation of the algorithm as C-code or as a state-chart is sufficient to validate the control strategy. With the strong digital capabilities of VHDL-AMS it is possible to move closer towards implementation than with usual signal-flow based tools. This adds an additional level of reality to the simulation that is helpful when used at the right stage in the design flow.

The digital brain of the robot is a board produced by the Oregon State University with an Atmel Atmega128 microcontroller (MCU) plus

peripheral hardware such as extended SRAM. An open source VHDL model of an Atmega103 was modified to model the Atmega128, including peripherals such as multiple interface ports and an analog-to-digital converter (ADC). This allows using target compiled code in the simulation in order to get very detailed information on how the software interacts with the MCU hardware, giving the benefits of:

- obtaining the correct connectivity of the individual components
- easier design and debug of the low level software drivers
- validation of correct control algorithm before hardware is available
- automatic regression testing of the complete system

For software designers the simulation approach offers the possibility of starting the debugging earlier in the design flow, with realistic stimuli and responses. Knowing all the conditions of the physical system at every time-step by analyzing the simulation results (figure 6) enables the software developer to correlate the actual behavior of the system with the execution of the software in the debugger. This effectively leads to more confidence in the code when trying to run the first prototype and less painstaking debugging on the hardware.

Figure 6. Simulation Results in EZWave

The Tekbot software designers for example were required to know the position of the Tekbot in the arena for the design and debugging of the line-following algorithm. Important analog waveforms therefore are the x and y coordinates of the left and right wheel, the speed of the wheels and the feedback from the sensors. Figure 6 shows the corresponding waveforms for the right wheel. Displaying the coordinates over time doesn't result in an

easily understandable spatial representation, but displaying the x over the y coordinate does. Exporting the simulated waveforms and importing them into Microsoft Excel allowed a representation of the simulation results that is easy to relate with the function of the control algorithm (figure 7 and figure 8).

Figure 7. Implemented Algorithm

Figure 8. Intended Algorithm vs. Implemented Algorithm

5. VERIFICATION AND OPTIMIZATION

Setting up the simulation of the full system or just parts of it is often time-consuming, but it enables automatic regression testing. Changes in the software or the hardware can be simulated and the results automatically verified against the specification or previous results – using exactly the same, reproducible tests repeatedly and saving engineer's time. While this is nothing new for IC designers, this methodology represents a fundamental improvement for the mechatronic design flow.

The ability to view the whole mechatronic system, including mechanical as well as electrical effects, means it is easier to find ways to optimize cost and robustness of the design. It enables engineers to move a problem into a design domain where it can be addressed more efficiently. The drivetrain, for example, has mechanical tolerances that make it behave less than ideal. Instead of purchasing a more expensive motor with fewer tolerances, the motor controller, or even cheaper – the software, can be designed to compensate for the tolerance, leading to the same system behavior at a lower cost.

6. OUTLOOK

Time constraints left a couple of aspects un-addressed in this project. One aspect is the analysis of different options for the control system implementation. A software implementation on a MCU is not necessarily the most efficient way to control the hardware. Other alternatives, such as FPGA or ASIC designs could be better options for some designs. With modern tools it is possible to use C code to drive hardware design flows for these technologies, possibly helping to identify the best implementation solution quicker and easier.

Another aspect is the use of more advanced analyses. For example engineers often use sensitivity or statistical analyses, such as Monte-Carlo analysis, as part of a structured process to identify areas for optimization. VHDL-AMS language constructs allow parameter variations, but libraries with statistical analyses are not part of the language definition. Such libraries currently become available as part of efforts of standardization committees.

7. CONCLUSION

The Tekbot design team used this VHDL-AMS based design methodology. According to the team, this project would not have been

completed without the ability to simulate electrical and mechanical hardware and software together. The simulations uncovered many flaws in the algorithms used for ball detection and line following. The ability to use a waveform viewer to pinpoint the cause of a bug greatly aided in the process of optimizing the ball detection and line following routines.

One of the biggest benefits experienced was the ability to model and simulate the design very early in the process. The design team was able to find problems prior to hardware prototyping. Changes could be made and re-simulated quickly. This enabled more time to be spent on making the design right before testing it on physical hardware and to stay within the very limited budget.

REFERENCES

- Dan Block, Tekbot technical paper, 2004, Tekbot design team member; http://www.mentor.com/system/tech_pubs/
- Mike Loudonback et. al., 2004, Tekbot technical manual; <http://classes.engr.oregonstate.edu/eecs/fall2003/ece441/groups/g20>
- Scott Cooper, 2004, "Design Team Collaboration within a Modeling and Analysis Environment", Mentor Graphics; http://www.mentor.com/system/tech_pubs/
- IEEE 1076.1 (VHDL-AMS) Working Group; <http://www.eda.org/vhdl-ams/>