Investigating the Simulink Auto-Coding Process

July 28, 2016

Matthew J. Gualdoni, EG2 intern

Michael A. Buttacoli, EG2 mentor

Abstract

Model based program design is the most clear and direct way to develop algorithms and programs for interfacing with hardware. While coding "by hand" results in a more tailored product, the ever-growing size and complexity of modern-day applications can cause the project work load to quickly become unreasonable for one programmer. This has generally been addressed by splitting the product into separate modules to allow multiple developers to work in parallel on the same project, however this introduces new potentials for errors in the process. The fluidity, reliability and robustness of the code relies on the abilities of the programmers to communicate their methods to one another; furthermore, multiple programmers invites multiple potentially differing coding styles into the same product, which can cause a loss of readability or even module incompatibility.

Fortunately, Mathworks has implemented an auto-coding feature that allows programmers to design their algorithms through the use of models and diagrams in the graphical programming environment SIMULINK, allowing the designer to visually determine what the hardware is to do. From here, the auto-coding feature handles converting the project into another programming language. This type of approach allows the designer to clearly see how the software will be directing the hardware without the need to try and interpret large amounts of code. In addition, it speeds up the programming process, minimizing the amount of man-hours spent on a single project, thus reducing the chance of human error as well as project turnover time. One such project that has benefited from the auto-coding procedure is Ramses, a portion of the GNC flight software on-board Orion that has been implemented primarily in SIMULINK.

Currently, however, auto-coding Ramses into C++ requires 5 hours of code generation time. This causes issues if the tool ever needs to be debugged, as this code generation will need to occur with each edit to any part of the program; additionally, this is lost time that could be spent testing and analyzing the code. This is one of the more prominent issues with the auto-coding process, and while much information is available with regard to optimizing SIMULINK designs to produce efficient and reliable C++ code, not much research has been made public on how to reduce the code generation time. It is of interest to develop some insight as to what causes code generation times to be so significant, and determine if there are architecture guidelines or a desirable auto-coding configuration set to assist in streamlining this step of the design process for particular applications.

To address the issue at hand, the SIMULINK coder was studied at a foundational level. For each different component type made available by the software, the features, autocode generation time, and the format of the generated code were analyzed and documented. Tools were developed and documented to expedite these studies, particularly in the area of automating sequential builds to ensure accurate data was obtained. Next, the Ramses model was examined in an attempt to determine the composition and the types of technologies used in the model. This enabled the development of a model that uses similar technologies, but takes a fraction of the time to autocode to reduce the turnaround time for experimentation. Lastly, the model was used to run a wide array of experiments and collect data to obtain knowledge about where to search for bottlenecks in the Ramses model. The resulting contributions of the overall effort consist of an experimental model for further investigation into the subject, as well as several automation tools to assist in analyzing the model, and a reference document offering insight to the autocoding process, including documentation of the tools used in the model analysis, data illustrating some potential problem areas in the autocoding process, and recommendations on areas or practices in the current Ramses model that should be further investigated.

Several skills were required to be built up over the course of the internship project. First and foremost, my SIMULINK skills have improved drastically, as much of my experience had been modeling electronic circuits as opposed to software models. Furthermore, I am now comfortable working with the SIMULINK Auto-coder, a tool I had never used until this summer; this tool also tested my critical thinking and C++ knowledge as I had to interpret the C++ code it was generating and attempt to understand how the SIMULINK model affected the generated code. I had come into the internship with a solid understanding of MATLAB code, but had done very little in using it to automate tasks, particularly SIMULINK tasks; along the same lines, I had rarely used shell script to automate and interface with programs, which I gained a fair amount of experience with this summer, including how to use regular expression. Lastly, soft-skills are an area everyone can continuously improve on; having never worked with NASA engineers, which to me seem to be a completely different breed than what I am used to (commercial electronic engineers), I learned to utilize the wealth of knowledge present at JSC. I wish I had come into the internship knowing exactly how helpful everyone in my branch would be, as I would have picked up on this sooner.

I hope that having gained such a strong foundation in Simulink over this summer will open the opportunity to return to work on this project, or potentially other opportunities within the division. The idea of leaving a project I devoted ten weeks to is a hard one to cope with, so having the chance to pick up where I left off sounds appealing; alternatively, I am interested to see if there are any opening in the future that would allow me to work on a project that is more in-line with my research in estimation algorithms. Regardless, this summer has been a milestone in my professional career, and I hope this has started a long-term relationship between JSC and myself. I really enjoy the thought of building on my experience here over future summers while I work to complete my PhD at Missouri University of Science and Technology.