Modeling the ”Genie” Controller with Ivy

Calvin Passmore
Department of Electrical and Computer Engineering
Utah State University
Logan, Utah; USA
A02107892@aggies.usu.edu

***Abstract*—This document discusses the methodology of modeling the Radio Controller (Genie) in Ivy.**

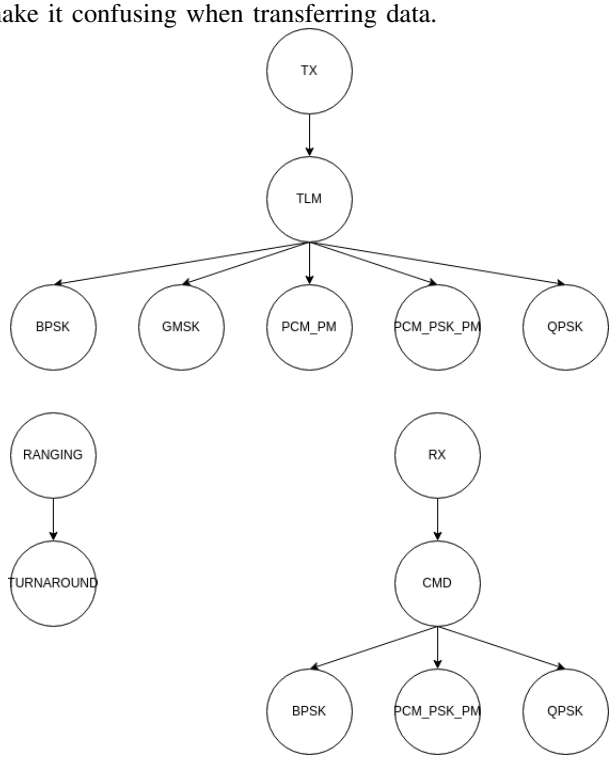
***Index Terms*—Co-conflicting: Only one of the nodes in the co-conflicting list can be enabled at a time**

I. Introduction

This document discusses the final project for ECE 6790: Cyber-Physical Systems.

II. The System to Model

The system being modeled is from an unclassified Space Dynamics Lab software-defined radio. The system holds nodes representing operating modes of the radio or different modulation schemes for both transmitting and receiving. The names of the nodes indicate where they lay in the tree, for example, rx_cmd indicates that there is a node named ’cmd’ that is a child of ’rx’. However, this is not a hard rule. There are modulation modes that are named ’pcm_psk_pm’, which does NOT indicate a tree structure. I would have fixed the naming convention for this project, however, in the actual system this is how they are named (not up to me) and I didn’t want to make it confusing when transferring data.



III. Transferring the model to Ivy

I chose to instantiate an object for every individual node. This is probably not the most efficient way to model this in Ivy, however, it works and I wasn’t sure how I would handle things like parents/children/conflicting nodes with a module. So I made each node a separate object and gave each an ’after init’ block, an ’on’ action, and an ’off’ action.

IV. Requirements

The purpose of the model was to fulfill the requirement that some modes cannot be enabled while other modes are also enabled. To be more specific, children nodes cannot be enabled without the parent node being enabled, and some modes are in conflict. The conflicting nodes are as follows:

- All of the children nodes of tx_tlm are co-conflicting - All of the children nodes of rx_cmd are co-conflicting - ranging_turnaound conflicts with tx_tlm

The ranging_turnaround conflict with tx_tlm in effect means that ranging_turnaound is in conflict with all the children of tx_tlm.

V. Environment

The environment was simulated by giving ivy all possible inputs to the controller and allowing ivy to decide the order that the inputs are called. The possible inputs are an on and off action for each node.

VI. Results of Simulation

I did run into some issues while running the code. The first is where I created an infinite recursion sequence when turning off rx_cmd. rx_cmd.off incorrectly called rx.off which correctly called rx_cmd.off and so on. It took me a while to realize that’s what happened, where I assumed Ivy was just having issues. Once I realized that I had to go over all the objects again and verify their on and off actions.

After that, I implemented the model to alter the current node mode before altering anyone else’s state. This causes an issue where if you were attempting to turn on tx_tlm, it would turn on tx_tlm before turning on tx. This violated the invariant that a child node is not enabled without its parent node. This is implemented correctly is the original system, I just modeled it incorrectly.

The last major issue I had was I had forgotten to list all of the conflicts in the rx_cmd_pcm_psk_pm on action, but did list them correctly in the invariants. This caused rx_cmd_qpsk to

be enabled at the same time as rx_cmd_pcm_psk_pm, causing a conflict violation. This raises a serious concern in regard to the original system. There is no empirical way to guarantee that all the conflicts are correctly listed. The only way to verify that all conflicts are listed correctly would be to do unit testing/system testing to test all possible cases, which is not always feasable.

VII. Improvements

One major way that I could improve the model is to find an efficient way to model it as modules in ivy so that every node isn't represented individually as objects. This would have prevented the infinite recursion issue.