A high-level overview of “Accurate Booleanization of Continuous Dynamics for Analog/Mixed-Signal Design”

Abdus Salam Azad

January 7, 2018

The dissertation investigates the problem of modelling AMS systems for BUG-free AMS Design. One of the core challenges in designing AMS systems is to choose a model that can accurately capture the properties of the underlying system and is also amenable to analysis and fast verification. Traditionally, the analog and digital circuits have been modelled accurately using DAEs and FSMs (or, similar discrete representations) respectively—two quite different languages which do not mix well. However, the existing modelling techniques for the AMS systems (which contain both analog and digital components closely interacting with each other) lack the desired accuracy (for simplifying assumptions) and scalability (limited to only a few continuous variables) for real-world practical AMS systems. The dissertation presents a method called “Booleanization” that can accurately model AMS systems with purely Boolean representation (e.g., FSMs). Boolean representations/models have a clear benefit as they allow the AMS systems to be verified by existing state-of-the-art Boolean verification tools, which can verify large industry level real-world digital systems. The dissertation proposes three different techniques of “Booleanization”, among which ABCD-NL is the most general and powerful.

ABCD-NL is a “Booleanization” technique, that takes an AMS system as an input (as a SPICE netlist or, a system of DAEs) and constructs an FSM as a Boolean approximation of the underlying system to accurately capture its I/O behavior. As the first step of the Booleanization technique, ABCD-NL first discritizes the given circuits inputs and outputs with a finite number of bits. In the next step, the FSM is constructed. ABCD-NL works under the assumption that, a DC input to the given system should eventually result in a DC output. Hence, upon a change of input the system first shows a transient behavior for a while and eventually settles into a constant DC voltage. To accurately capture the behavior of both the DC and transient behavior of the underlying system, the ABCD-NL-constructed FSMs feature two distinct type of states: 1) DC states and 2) Transient states (between each pair of DC states). Each DC state correspond to a distinct combination of the input symbols and represent the DC operating point of the system for that input. When the input changes and kept fixed(constant) for a ‘reasonable’ amount of time, the FSM transitions from one DC (or, transient too [1]) state into another DC state and converges [2]. It is on the path to the DC states, the transient states are visited and the appropriate transient output behavior is exhibited.

The transition between different states is triggered by a change of input (both in DC and transient states) or, by the advancement of time (while in a transient state). These transitions are represented by three different types of arcs: 1) self-looping arcs of the DC states, 2) The outbound arcs of the transient states leading them from one DC state to another, and 3) the outbound arcs of the transient states corresponding to input changes during the transient state of the system. ABCD-NL utilizes SPICE simulations of the underlying AMS systems to determine the output labels of the arcs to correctly portray the desired I/O behavior. Determining the value of the self-looping arcs to the DC states are straight-forward. The input value corresponding to the arc (from the input label) is fed to the SPICE simulator and the discretized DC (steady) output is used as the label. For labeling the arcs of the transient states between two DC states e.g., DC1 and DC2, ABCD-NL simulates the transient behavior of the system in SPICE feeding it a step input transitioning from the DC level of DC1 to the DC level of DC2. Afterwards, a number of transient states (the number is determined using the estimated settling time and FSM time step) are inserted between the two DC states and their output labels are determined from the

[1]If an input is changed, while the underlying system is still in a transient state due to a previous change in input, the transition to a DC state can start from a transient state, too.

[2]In case of a constant input kept fixed for a “reasonable” amount of time, the corresponding FSM converges into one of the DC states, similar to the corresponding DAEs it is approximating. This “convergence” in the FSM is achieved by a self-loop in the DC states. The self-loop is labeled by the corresponding constant input.

trajectory of the simulation output. For constructing and appropriately labeling the third kind of arcs, 'Jumping heuristic's are used. A jumping heuristic finds a 'similar' state for the transient state in consideration. The next state and output label of the 'similar' state are used by the transient state.

One of the key challenges in AMS modelling is the trade off between modelling accuracy and scalability. While, ABCD-NL can accurately capture the I/O behavior of the underlying AMS systems, the issue of scalability can still arise. Consider an AMS system with 2 inputs and we want to discretize each input with 16 bits. The number of DC states in the FSM will be $2^{2\times 16}$. In general, if there are n input signals and we discretize each with b bits, the total number of DC states in the corresponding FSM is 2^{nb}, i.e., the number of DC states is exponential to the number of bits required to discretize the input signals. The number of the transient states are also exponential to the number of input bits. As there are 2^{nb} DC states, there are ${}^{2^{nb}}P_2$ (i.e., $(2^n)^2 - (2^n)$) distinct transient paths. If we consider t as the average number of states in the transient paths, the total number of transient states is $t \times {}^{2^{nb}}P_2$. Hence, the size of the FSM is exponential to the number of input bits nb.

Now, the algorithm ABCD-NL proposes to construct the FSMs, takes each ordered pair of the DC states (i.e., the nested loop between Line 4 to 14 of Algorithm 1) to construct the transient paths between them. The number of iterations is hence, $(2^n)^2 - (2^n)$. For simplicity, if we assume that each such transient path takes a constant time to construct, the nested loop is of exponential order i.e., $O((2^n)^2 - (2^n))$ or, simply $O(4^n)$. The other nested loop that the algorithm features between Line 15 to 19, is also of exponential order, i.e., $O((t \times (2^n)^2 - (2^n)) \times 2^{nb})$ or, simply $O(2^{n(2+b)})$. Hence, the overall time complexity of the algorithm is of exponential order, which limits the capability of ABCD-NL by limiting the number of inputs and discretization levels it can work with.

An alternative way to model the true I/O behavior of the AMS systems can be using machine learning techniques. Artificial Neural Networks(ANN) can be well suited for this task. Some of the reasons are discussed below:

- ANN does not need to save every possible input combination. Rather, it will need one input node per input variable.
- ANN can take both discrete and continuous values as input. The output is typically continuous, which can be discretized if needed.
- Most importantly, in theory feed forward neural nets are universal approximators [1]. In practice, ANNs have shown to closely approximate complex linear and non-linear functions.
- The training samples for the ANN can easily be generated from SPICE simulations.
- Each evaluation of the function can be done with one forward pass, which takes constant time.
- There is also an interesting analogy between a Recurrent Neural Network, a special kind of ANN, and a Finite State Machine, which is also worth exploring ([2] and [3]).

To summarise, this report takes its view on ABCD-NL—an accurate modelling technique for AMS systems. ABCD-NL constructs an FSM to accurately capture the underlying I/O behaviour. However, it suffers in terms of scalability, another important property of the AMS models. The report eventually proposes ANN as a modelling alternative. Proper investigation with ANNs may result in an accurate and scalable AMS modelling technique.

References

[1] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[2] P. Tino, B. G. Horne, and C. L. Giles, "Finite state machines and recurrent neural networks–automata and dynamical systems approaches," tech. rep., 1998.

[3] M. Casey, "The dynamics of discrete-time computation, with application to recurrent neural networks and finite state machine extraction," *Neural Computation*, vol. 8, pp. 1135–1178, Aug 1996.