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. The rest of this report provides a brief overview on how ABCD-NL works and some directions that can be investigated to improve it further.  
  
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 circuit’s 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 show 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 [footnote BB]) state into another DC state and converges [footnote - AA]. 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 with a change of input (both in DC and transient states) or, by the advancement of time (while in a transient state). Three different types of transitions are possible within the FSM: 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 transient state of the system.

ABCD-NL utilizes SPICE simulations of the underlying AMS system to determine the output labels of the arcs. 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 a step input, transitioning from the voltage corresponding to 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 label are determined from the trajectory of the simulation output. For constructing and labeling the third kind of arcs, `Jumping heuristic's are used to approximate an appropriate next state and output symbol. Using the jumping heuristics a DC or transient state is selected which is similar to the transient state in consideration and the next state and output label of the selected state is also used for the transient state too.   
  
   
The previous section briefly described how ABCD-NL constructs FSM models to approximate the I/O behavior of an AMS system. The current section discusses some of the future directions---

One of the key challenge in for AMS modelling is the trade off between modelling accuracy and scalability. While, ABCD-NL can accurately capture the I/O behavior of the underlying system, 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 2^32. In general, if the total number of bits to represent all the input signals is $n$, the number of DC states the FSM needs is $2^n$, 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, $n$. As there are $2^n$ DC nodes, there are $2^n P 2$ TRAN paths ($ (2^n)^2 - (2^n) $), which is also exponential to the number of total input bits. If we consider, the number of average transient states between two DC nodes is $t$, the total number of transient states is $ t \* 2^n P 2 $. Hence, the size of the FSM is exponential to the number of input bits and hence to the number of inputs.

AA - 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 labelled by the corresponding constant input.   
   
BB- 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.

footnote CC: For correctly capturing the system's behavior in case of a change in the input during a transient state, arcs are needed to be constructed from each transient state for all input symbols (One of the input symbol---the symbol corresponding to the DC state the transient is leading to---is already covered during constructing the second set of arcs).