

Abstracting and Verifying Flash Memories

Sandip Ray
University of Texas at Austin
sandip@cs.utexas.edu

Jayanta Bhadra
Freescale Semiconductor Inc.
jayanta.bhadra@freescale.com

Memory verification entails showing that the transistor network implements a state machine storing and retrieving data at addressed locations. For SRAM designs, verification proceeds by extracting a *switch-level model*, namely a graph of nodes connected by transistor switches. A node has state 0, 1, or X; a switch has state “open”, “closed”, or “indeterminate”; state transitions are specified by switch equations.

However, switch-level models cannot be used for flash memories, which contain both CMOS and *Floating Gate* (FG) transistors. FG transistors (Fig. 1A) have, in addition to the

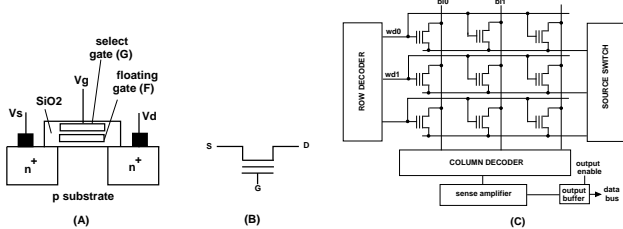


Fig. 1. (A) Structure of an FG transistor. The polysilicon layer between the Gate G and Substrate provides the capacitive coupling. (B) Schematic representation of an FG transistor in a larger design. (C) Implementation of a NOR Flash Configuration with FG transistor.

drain (D), gate (G), and source (S) terminals, a *floating gate* (F) — a polysilicon layer physically disconnected from both S and D. Capacitive coupling between G, F, and the substrate is exploited to design a bitcell with a single transistor, by regulating the threshold voltage V_{th} through control of the charge stored: low threshold voltage (V_{th}^L) represents logic 1 and high threshold voltage (V_{th}^H) represents 0. Unfortunately, this coupling breaks the view of a transistor as a simple switch, making traditional switch-level analysis untenable. Consequently, there is a “verification gap” in current industrial practice for SoC designs containing flash components: a high-level (C/C++) description is used to model the *interface* of the flash with surrounding SoC blocks, but the underlying transistor network is not guaranteed to implement the description.

We circumvent this problem through a new approach for abstracting memory designs. Instead of extracting switch-level models by structural analysis, we model the *behavior* of the network. The viability is based on the observation that a custom memory is designed by interconnecting cohesive, logical units such as bitcells, sense amplifiers, etc. The units are architected to operate over a limited sequence of stimulus patterns, each validated by analog SPICE simulation across process corners and operating conditions. The behavior of each

unit *under operating condition* is formalized as a parameterized state machine, using guarded transitions to encode its operating constraints. The behavior of a complete memory core is modeled as an interacting state machine composition.

Consider the use of the method in abstracting the NOR flash design in Fig. 1C. *Reading* is performed by applying a voltage v ($V_{th}^L < v < V_{th}^H$) at G which is driven by the selected wordline, keeping other wordlines at ground. If the cell has logic 0, the transistor does not turn on and no current flows to the sense amplifier; otherwise the bitcell turns on and current is detected, reading a 1. For *programming*, the Channel Hot-Electron Injection procedure injects negative charge into FG, raising its V_{th} to V_{th}^H . The bitcell is then read with a gate voltage v ($> V_{th}^H$); a result of 0 indicates successful programming. *Erasing* is performed along a sector using Fowler-Nordheim tunneling, and involves (i) raising the V_{th} s of the bitcells in the sector to V_{th}^H , (ii) charge removal to lower all the V_{th} s to V_{th}^L , and (iii) normalization, which employs soft programming to increase the V_{th} of cells that have fallen below V_{th}^L . While the analog effects involved in these operations are complex, the behavior of each constituent unit, namely bitcell, sense amplifier, etc., in the range of operation, can be modeled as a state machine by characterizing the constraints on the stimuli provided. Verification reduces to assume-guarantee argument showing that the constraints for each unit are satisfied by the stimuli from neighboring units.

We have used the approach to verify parameterized models of both NOR and NAND flash configurations. To our knowledge, our work provides the first platform for formal functional verification of realistic flash designs. Since individual memory units are modeled as state machines, traditional simulation and verification tool flows can be easily adapted to handle these models. Furthermore, the specification is extracted from the flash interface with its surrounding SoC blocks; functional verification of digital components can be hierarchically composed with flash models for full SoC verification. Finally, a key feature of our framework is the direct correspondence between components used for analog simulation and behavioral models for individual units. This facilitates corroboration of models with readily available simulation data. This correspondence makes it viable to use learning techniques automate extraction of the behavioral models from simulation patterns as follows. Traces from SPICE simulation can be used to learn the parameters of the state machines for each unit through iterative refinement and the iterations can be seeded by the operating constraints used in the SPICE simulation.