White Paper

Benefits of Using ESP in Memory Designs

May 2010

Author

Ken Hsieh
Product Marketing Manager,
Synopsys, Inc.

Clay McDonald
R&D Manager,
Synopsys, Inc.

Introduction

ESP is an equivalence checking tool commonly used for full functional verification of custom designs such as memories, custom macros, standard cell, and IO cell libraries. It is used to ensure that two design representations are functionally equivalent. Many people are familiar with the logic cone based equivalence checking which is very effective for synthesizable designs with gate-level implementations, but it cannot be easily adapted to full custom circuits like memories and custom macros. ESP, using formal technique, is based on symbolic simulation to remove many of the restrictions that imposed by the logic cone based equivalence checking.

Symbolic simulation approach that ESP is based on can be considered a combination of two methodologies, the conventional simulation and the logic cone based equivalence checking. As in conventional simulation, equivalence is checked by simulating both designs and checking that their outputs match over a number of cycles. However, by borrowing some of the techniques from cone-based equivalence checking, symbolic simulators can perform this comparison for extremely large (potentially exhaustive) sets of test vectors. While conventional simulation applies vectors of 0s and 1s to the inputs, symbolic simulation applies vectors of symbols, where each symbol can represent both 0 and 1 simultaneously. Internally, the symbolic simulator propagates equations of these symbols towards the outputs, producing a symbolic result which is then used to check equivalence.

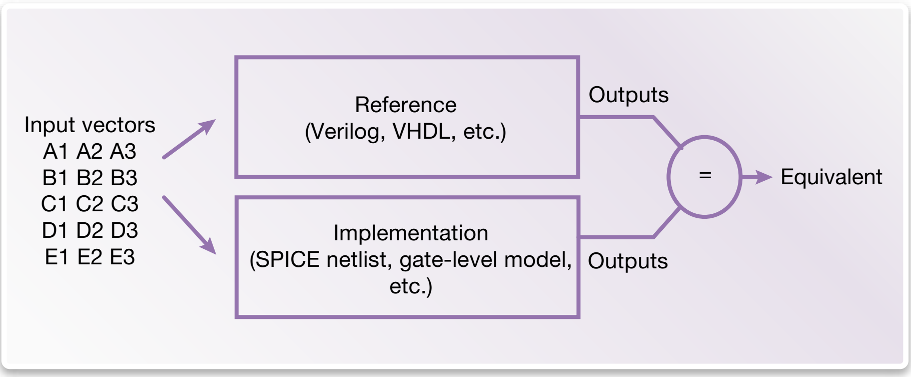
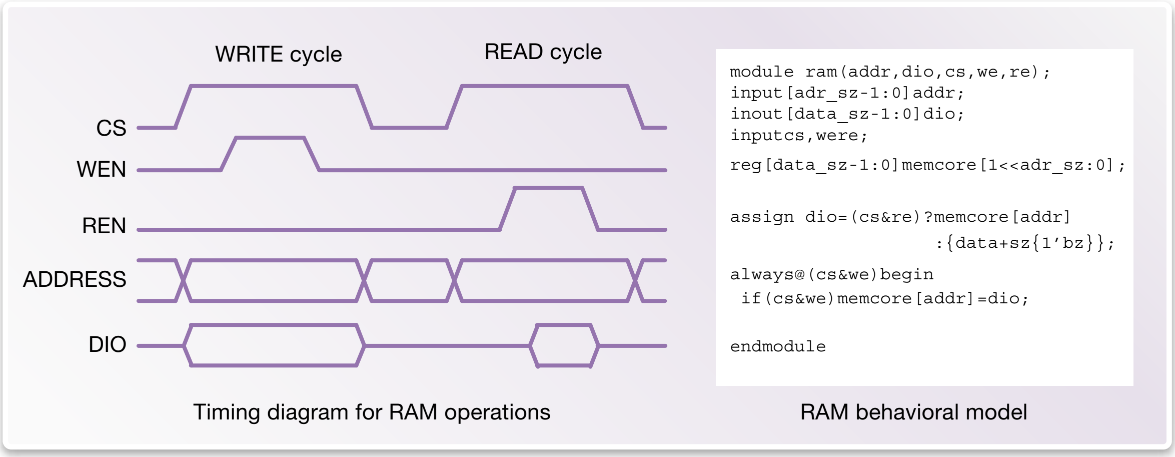

Input vectors
A1 A2 A3
B1 B2 B3
C1 C2 C3
D1 D2 D3
E1 E2 E3
Reference (Verilog, VHDL, etc.)
Implementation (SPICE netlist, gate-level model, etc.)
Outputs
Outputs
=
Equivalent

Figure 1: Functional equivalence checking based on symbolic simulation

The output from the symbolic simulator encodes the results of all input test vectors that could have been created by assigning a 1 or 0 to any of the input symbols. Therefore, these results contain exactly the same information as would be obtained by simulating with an exhaustive set of traditional vectors, providing vastly improved coverage over conventional simulation. Furthermore, it does so without any of the restrictions imposed by cone-based logic equivalence checking. In other words, symbolic simulation is capable of comparing two designs at any abstraction level (even behavioral RTL, or transistor-level SPICE), even if the two designs have completely different state encodings.

Figure 2 is a simple SRAM example showing a typical memory specification model written in behavioral Verilog, and such behavioral models are easy to write and are very efficient for full-chip simulation, but cannot be synthesized into flip-flops and combinational logic. The behavioral models usually represent the intended behavior of the design rather than the actual structure or implementation.



```
module ram(addr,dio,cs,we,re);
input[adr_sz-1:0]addr;
inout[data_sz-1:0]dio;
inputcs,were;
reg[data_sz-1:0]memcore[1<<adr_sz:0];

assign dio=(cs&re)?memcore[addr]
                  :{data+sz{1'bz}};
always@(cs&we)begin
 if(cs&we)memcore[addr]=dio;

endmodule
```

Figure 2: Sample RAM behavioral model and timing diagram

Because logic cone based equivalence checkers require the models to be represented as combinational logic, the behavioral models must be translated to a functionally equivalent state-based representation such as RTL. This is a very difficult task which imposes unnecessary restrictions on logic designers and decreases productivity.

Self-timed circuits, another example in the memory design, are often used to improve the timing of SRAM read operations. Inverter delay chains are simple to design but difficult to scale, especially for memory compilers that need to generate memory blocks with sizes ranging from kilobits to megabits. Furthermore, inverter chain delays will not reliably track the read-circuitry timing across process variations. Thus, a replica technique using a dummy column is often selected for advanced SRAM designs, such as the one shown in Figure 3..

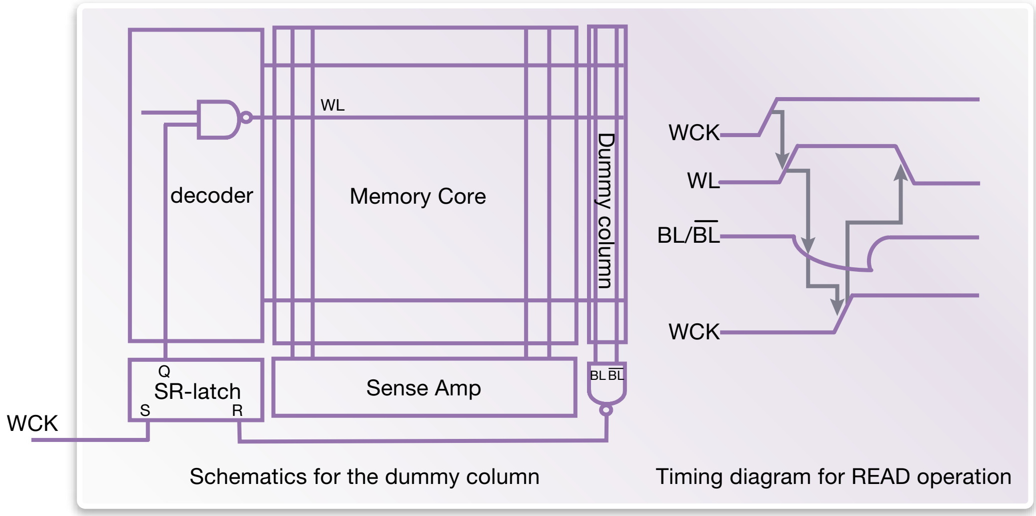


Figure 3: Self-timed circuit to improve READ operation

Cone-based logic equivalence checkers do not model delay, and cannot handle self-timed circuits. Thus they cannot verify equivalence of any memory designs where the timing is determined by dummy columns.

Other circuits like bi-directional transistors and sense-amp logic are also common circuits implemented in the memory designs. The bi-directional transistors as shown in the following diagram have column decoder circuit requires four pass transistors that are controlled by col_sel[0] and col_sel[1]. The directions of these transistors depend on whether the memory is currently being read or written. This bi-directional behavior cannot be easily modeled at the gate level, in which the closest equivalent is most likely a mux. Since cone-based equivalence checkers rely on abstracting transistors to the gate level, they cannot handle bi-directional transistors unless a fairly complicated manual model is provided. This further increases the modeling efforts and reduces confidence in the result.

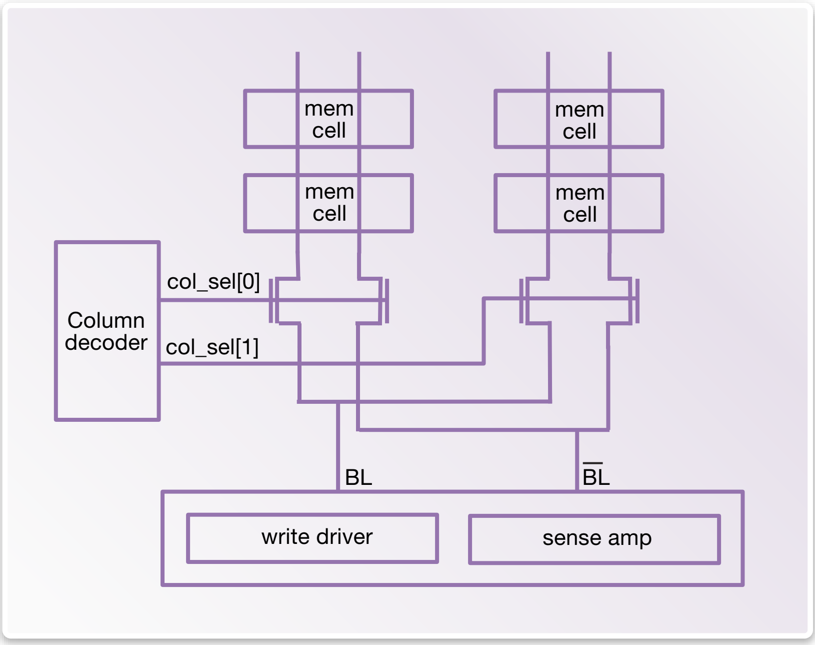


Figure 4: Column decoder with bi-directional pass tranistors

Sense-amp techniques have become increasingly common in high performance data path and memory circuits. The following figure shows a sense-amp based transparent latch with its timing diagram.

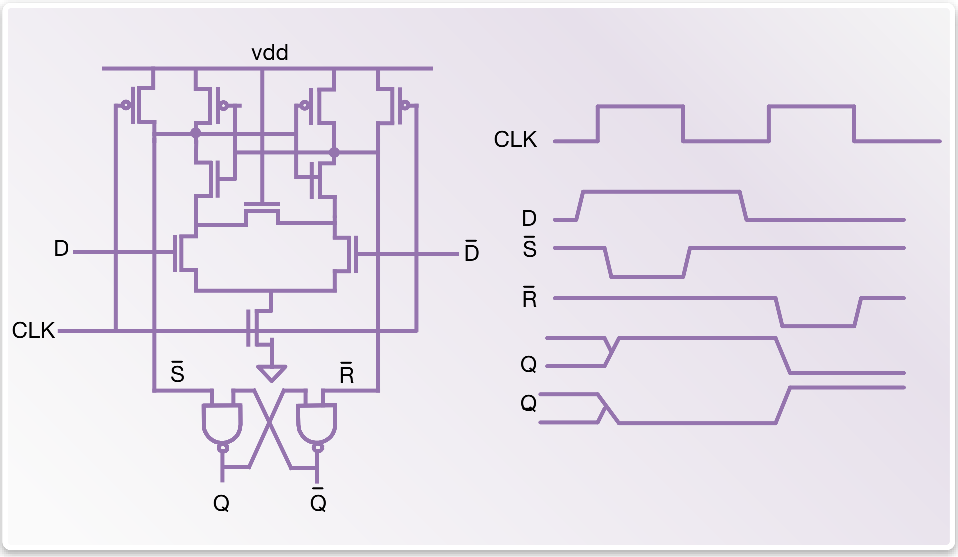


Figure 5: Schematic and timing diagram for a sense-amplifier based transparent latch

Since logic equivalence checkers assume all signals are full rail signals, any circuits with sense amplifiers cannot be equivalence checked unless the circuit block containing the sense amplifier is replaced with a manually generated model. This modeling is tedious and error-prone, especially if additional logic is embedded inside the sense amplifier block.

In summary, equivalence checking has been widely adopted as part of the verification flow for IC designs. Logic cone based equivalence checking is very effective for synthesizable designs with gate-level implementations, but cannot be easily adapted to full custom circuits such as memories, standard cell and IO cell libraries and custom macros. Symbolic simulation removes many of the restrictions imposed by logic cone based equivalence checking, and provides a solution for verifying full custom designs. The unique advantages and benefits of using ESP are that it supports non-synthesizable Verilog. It uses circuit behavior and not transistor-level abstraction. ESP considers timing dependent functionality and verifies structurally different implementations such as retiming, pipeline depth, redundancy, etc. with no state point mapping required. In short, if it simulates, it's supported by ESP.