SIEMENS
Ingenuity for life

Siemens Digital Industries Software

Finding the perfect balance

Flexibility and quality in SRAM layouts

Executive summary

SRAM verification is a challenge, yet critical part of SoC tape-outs. The challenge for SoC designers at advances nodes is to challenge their need to customize SRAM blocks to obtain better performance, against the potential impact of those changes on yield. Debugging SRAM modification errors effeciently requires designers to be able to quickly and precisely locate an SRAM error and determine the correct fix. By enhancing the SRAM debugging process with pattern matching and similarity checking, Siemens Digital Industries Software enables SRAM designers to find a better, more precise balance between design flexibility and yield.

Elven Huang
Siemens EDA

Introduction

Memory is a critical component in system-on-chip (SOC) designs, especially at advanced nodes. SoCs require a variety of memory blocks, such as static random-access memory (SRAM) and dynamic RAM (DRAM), to perform computations. SRAM blocks, which consist of an assembly of specialized calls that abut or overlap one another in a specific arrangement that matches the circuit specifications (figure 1), are typically used for operations requiring high-speed access to frequently used data. Most integrated circuit (IC) design companies use proven SRAM blocks from the foundries as intellectual property (IP), rather than creating their own SRAM designs.

Because SRAM blocks typically consume a large part of die area, the quality of these SRAM blocks has a major impact on yield. While the foundry ensures that the SRAM IP they supply will provide good yield, there will always be a need for SoC designers to perform some customization on the SRAM blocks they incorporate into their designs. As process margins get tighter with every node, foundries have less and less tolerance to adjust or fine-tune their manufacturing process to allow for these intentional (or unintentional) modifications made to the SRAM blocks by design companies.These adjustments can create potential defects and impact yield.

Due to these process limitations, and knowing that SRAM modifications are inevitable, the foundries provide the most accurate design rules possible to help SoC designers verify that their memory blocks are design rule compliant and robust in the SoC layout before tape-out. The challenge for SoC designers at advanced nodes is to find the balance between their need for some customization to an SRAM block to obtain better performance, and the potential impact of those changes on yield.

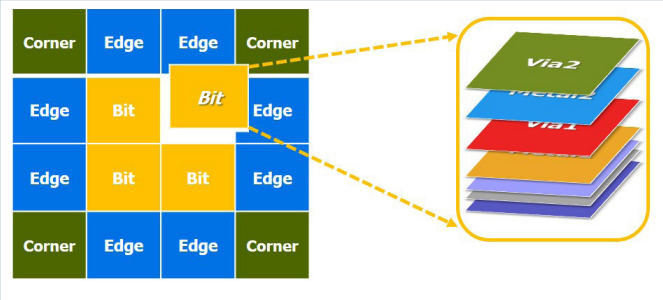

Figure 1: An SRAM block consists of multiple cells, and each cell includes multiple layers.

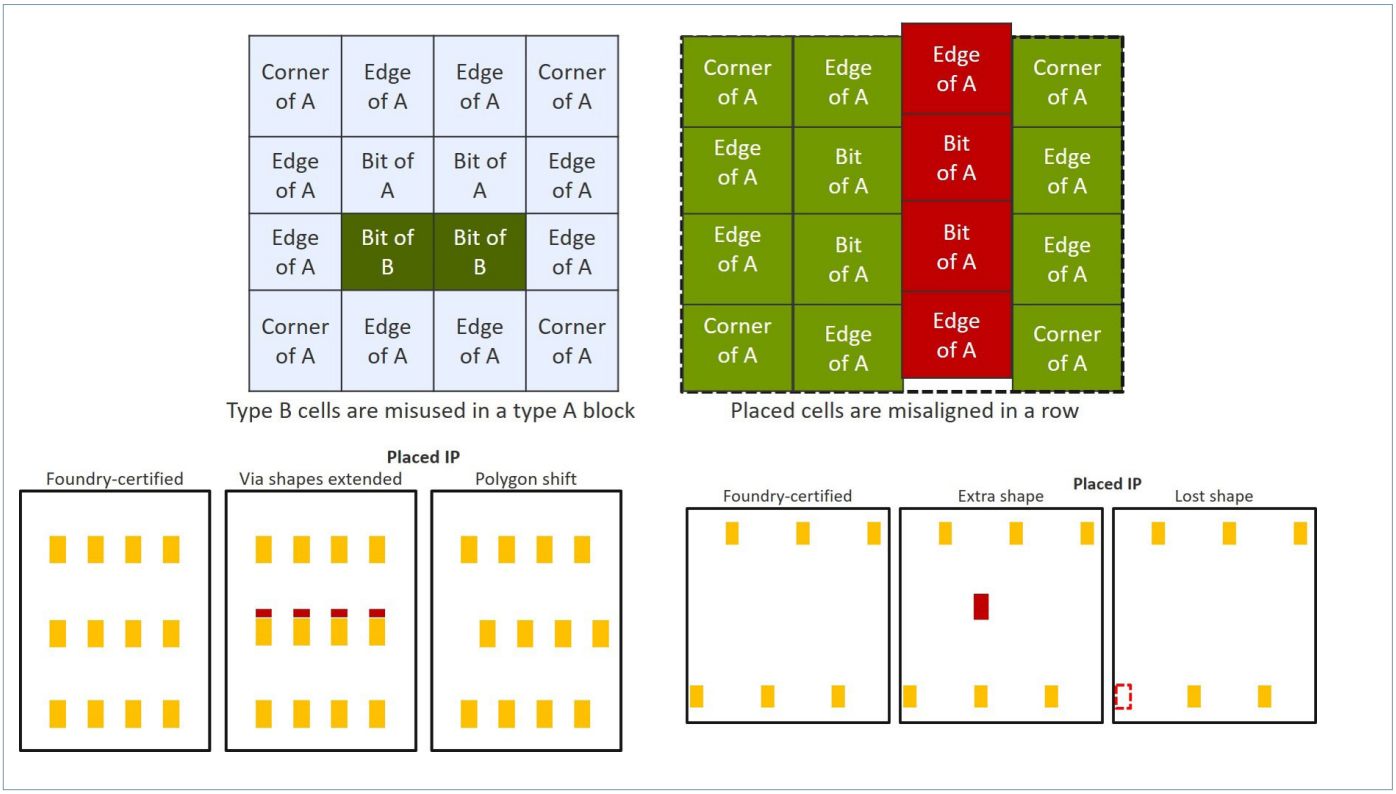

Figure 2: Various potential errors in SRAM blocks.

SRAM verifications challenges

So, what are these difficulties designers face in the verification and debugging of memory blocks in a SOC chip? The structure of a SRAM block is complicated, not only because it involves multiple layers and multiple cells, but also because it requires all cells to be placed in a specific configuration. Additionally, there are usually multiple types of SRAM blocks used in a SOC, and they may not have the same cells or configurations. In general, a SRAM block is just too complicated to verify with traditional design rule checking (DRC) alone. Figure 2 shows some of the common errors found in SRAM blocks.

Debugging these errors presents yet another challenge for SoC designers and manufacturers. Since traditional DRC rules are unlikely to identify all the types of errors that can occur in a complicated 2D configuration layout, considering all involved layers and cells, they must use supplementary verification and debugging techniques.

Cell-based checking

Some designers use a cell-based checking method to determine if any contents inside the SRAM cells were modified, and use the DRC rules to find errors at the top level. Together, these verification techniques can cover most known error types and identify any cell contents that have been modified. However, the cell-based method not only requires the foundry to include all possible cell layouts in the process design kit (PDK), but it can only identify which cells were modified, and where there are unmatched cells. Because it cannot identify exactly where or what the errors are, designers often spend much more time than expected to locate errors (particularly errors created by unintentional changes), as shown in figure 3. Typically, designers must ask the foundries if these errors are valid, and/or how to fix the layout to correct them.

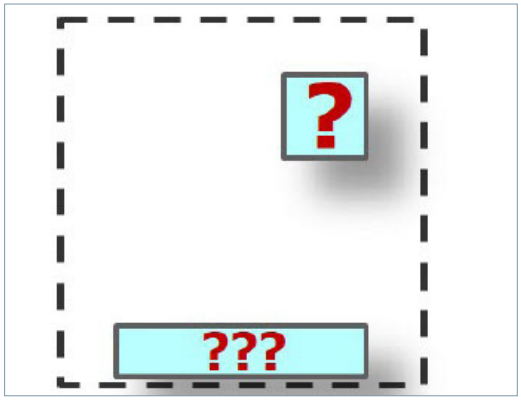

Figure 3: Error results reported by cell-based SRAM verification methods. Designers have no clues to help them identify error locations and involved layers.

This lack of location information for reported errors is not just a time-consuming annoyance for both designers and foundries, but the major obstacle in achieving a balance between design flexibility and block quality in SRAM verification. As previously discussed, a SRAM block contains multiple cells and involves multiple layers, and each SRAM type must follow a particular configuration in placement. The foundries may not know which SRAM types are being used in a layout, and which answer is needed to correctly verify a problematic SRAM block in a customer's design. Even when SoC designers know where those unmatched areas are, they must still spend a significant amount of time determining exactly where the errors are and how to fix them by examining multiple SRAM cell libraries and then checking every layer to find the differences.

Pattern matching verification

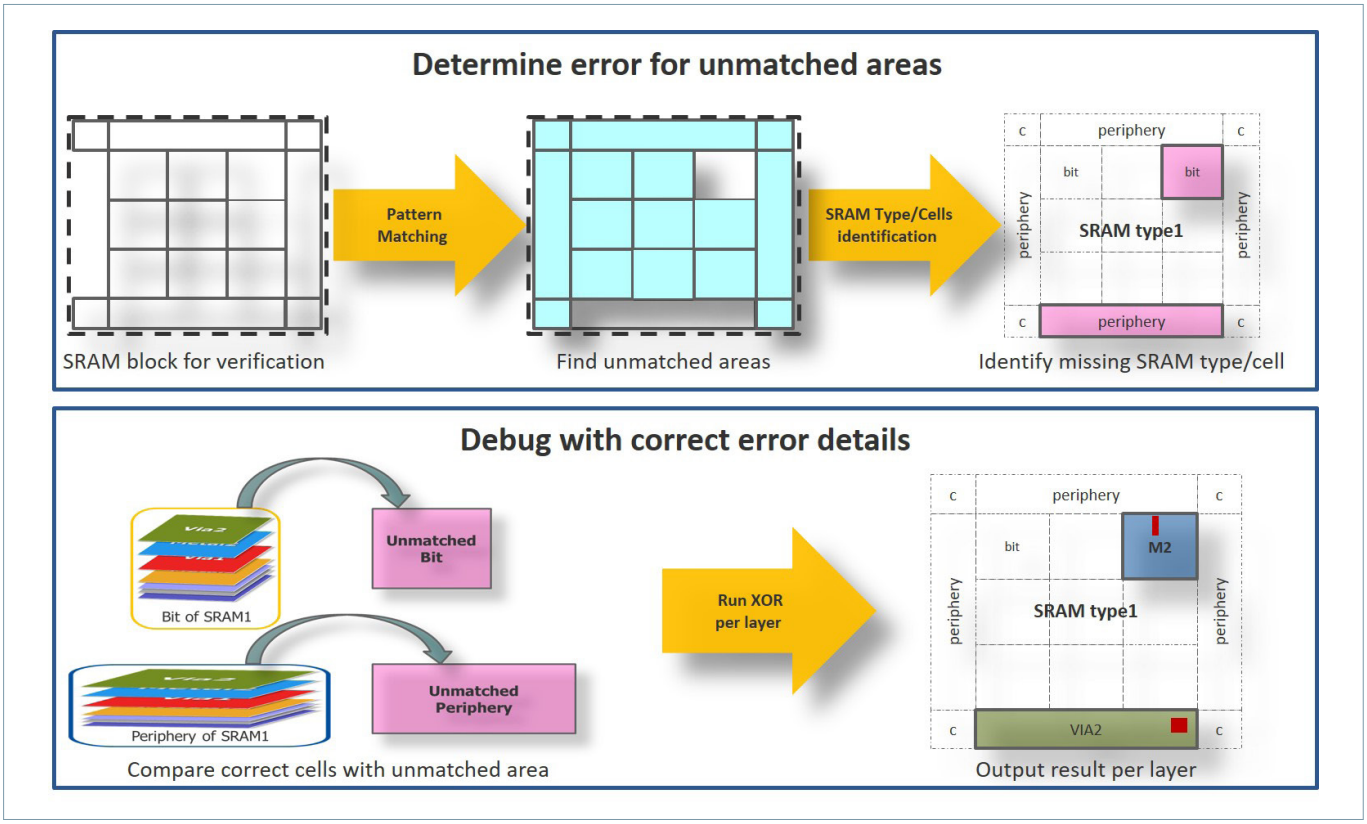

Figure 4: A basic SRAM debugging process using pattern matching.

In an ideal verification flow, designers should be able to see what kind of errors occurred and what differences were found, providing them with the information they need to determine root cause. With this information, designers can then fix the errors in the layout immediately, or pass those identified differences back to the foundry for evaluation and further discussion on solutions, which may include adding those differences into the verification rules as allowed tolerances.

However, those allowed tolerances must be precisely placed at the exact location of the original cells of that specific cell type to prevent the SRAM block from influencing other cells or other types of SRAM block, and to prevent real errors from being waived.

For example, in some cases, an upper corner cell must be placed in the top of an SRAM block and cannot be replaced with a lower corner cell, even if the space for the bottom corner is exactly the same, and even if the upper and lower corners look similar.

Pattern matching, in which the geometries in a layout are compared to a library of approved patterns, is a proven verification technique that can be applied to SRAM verification. Patterns can be defined to require exact matches, or to allow certain tolerances. A basic verification and debugging process using pattern matching is shown in figure 4.

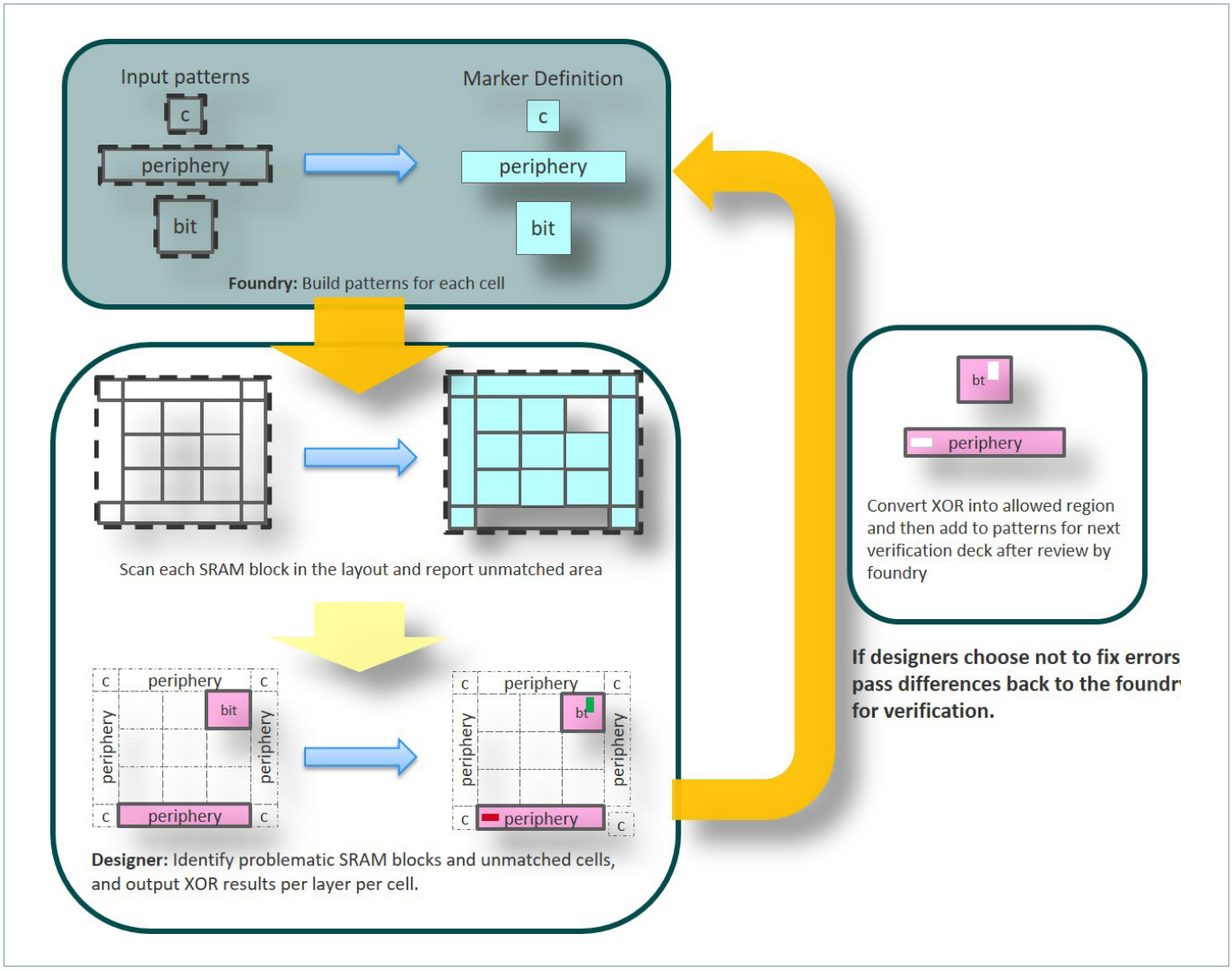

Figure 5: Ideal flow to incorporate tolerance into SRAM verification rule deck.

Siemens EDA, a part of Siemens Digital Industries Software, has collaborated with leading foundries and design companies over multiple nodes to develop a pattern-based utility to perform SRAM verification and debugging. Additionally, because this utility supports allowed regions during matching, differences in similarity matching (XOR) can be added back to patterns to improve precision for future verification jobs (figure 5).

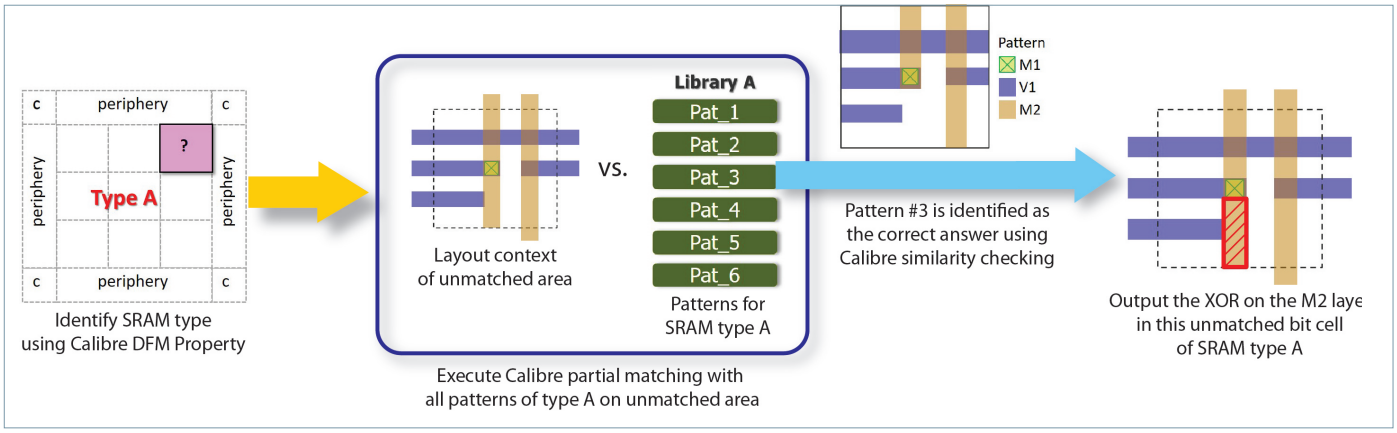

Figure 6: SRAM debugging using the Calibre Pattern Matching solution.

Siemens EDA then further enhanced the debugging flow to enable designers to more quickly identify both the error and the appropriate correction. With a comprehensive pattern debugging kit that combines the Calibre DFM Property and XOR functionality, designers can now see exactly where errors were found in various output formats, and either fix them right away or provide the details back to the foundry for inclusion into enhanced SRAM rules (figure 6).

Conclusion

SRAM verification is a challenging, yet critical part of SoC tape-out. Debugging SRAM modification errors efficiently requires designers to have access to the information that enables them to quickly and precisely locate an SRAM error and determine the correct fix. By enhancing the SRAM debugging process with pattern matching and similarity checking, Siemens EDA enables SRAM designers to find a better, more precise balance between design flexibility and yield.

Siemens Digital Industries Software

Headquarters
Granite Park One
5800 Granite Parkway
Suite 600
Plano, TX 75024
USA
+1 972 987 3000

Americas
Granite Park One
5800 Granite Parkway
Suite 600
Plano, TX 75024
USA
+1 314 264 8499

Europe
Stephenson House
Sir William Siemens Square
Frimley, Camberley
Surrey, GU16 8QD
+44 (0) 1276 413200

Asia-Pacific
Unit 901-902, 9/F
Tower B, Manulife Financial Centre
223-231 Wai Yip Street, Kwun Tong
Kowloon, Hong Kong
+852 2230 3333

About Siemens Digital Industries Software

Siemens Digital Industries Software is driving transformation to enable a digital enterprise where engineering, manufacturing and electronics design meet tomorrow. Our solutions help companies of all sizes create and leverage digital twins that provide organizations with new insights, opportunities and levels of automation to drive innovation. For more information on Siemens Digital Industries Software products and services, visit siemens.com/software or follow us on LinkedIn, Twitter, Facebook and Instagram. Siemens Digital Industries Software – Where today meets tomorrow.

siemens.com/eda

81838-C2 10/19 K