The Aurora RAM Compiler

Ajay Chandna, C. David Kibler[+], Richard B. Brown, Mark Roberts, Karem A. Sakallah

University of Michigan, Department of Electrical Engineering & Computer Science, Ann Arbor, MI 48109-2122

[+] Hewlett Packard Company, 3404 East Harmony Rd., Ft. Collins, CO 8052

Abstract - This paper describes a RAM compiler for generating and characterizing highly manufacturable optimized SRAMs using GaAs E/D MESFET technology. The compiler uses a constraint-driven design flow to achieve process tolerant RAMs. This compiler was built using a flexible design framework that can be easily adapted to optimize and characterize memories in different MESFET processes.

I. INTRODUCTION

RAM compilers have grown in sophistication from early implementations which were merely layout generators[1] to more recent implementations[2,3] which also perform automatic buffer sizing to meet delay requirements. The compiler presented in this paper incorporates additional features which are needed to produce manufacturable memories in GaAs. The principles used in this compiler have evolved over the design, test, and yield analysis of numerous GaAs SRAMs[5].

Two compilers which represent the approaches currently used in SRAM compilers are the Memorist compiler reported in [3] and the Cascade Design Automation (CDA) multi-port RAM compiler reported in [2]. Both compilers strive to provide flexibility for the user. The Memorist compiler provides this by allowing a choice of either look-up table or SPICE delay calculations, and by allowing the user to choose from a wide range of physical organizations that best meet their power-delay-area requirements. The CDA compiler uses timing driven buffer sizing and process independent layout generators to provide flexibility.

The Aurora RAM compiler shares the general goals of these compilers, while introducing the following new objectives:

1. Develop a CAD tool which can provide process-tolerant sub-2.5ns memory designs of up to 8Kb in size using GaAs E/D MESFETs.
2. Build a compiler framework that can easily adapt to changes in processing technology.
3. Develop a framework that allows the systematic trading of power for speed in an intelligent manner by providing timing-driven automatic transistor sizing.

A new design methodology was developed to build a compiler to meet these goals.

A. Motivation for New Methodology

Look-up tables or equation-based macromodels are commonly used in CMOS SRAM compilers for calculating delays and power dissipations. An advantage of this approach is that the calculations are orders of magnitude faster than performing simulations using a transistor level circuit simulator such as HSPICE. Another advantage is that SRAM compilers which are developed as part of a larger CAD framework can make extensive use of lookup tables and macromodels that are already in place. There are, however, a number of serious limitations associated with this approach, especially when applied to E/D MESFET GaAs SRAM design.

The first disadvantage is the considerable cost associated with macromodel development. Once the macromodels are developed, they are tied to circuit structures that are designed in a specific process. If the compiler were to be used for a more advanced process, or if new and improved circuit structures were needed to provide enhanced performance, new macromodels would need to be developed to provide fits to the new process, or to provide good fits to the new circuit structures.

Another drawback of the macromodel approach is related to E/D MESFET GaAs circuit design. The signal swing of most logic signals is only 550mV and noise margins are typically below 150mV. Also, leakage currents in GaAs are orders of magnitude higher than those found in MOS devices. Thus, even when a transistor is turned off, its output conductance is high enough that it can have a substantial impact on the voltages of the nodes to which it is connected. These low noise margins require macromodels for signal levels as well as delays, to ensure signal integrity in critical portions of the RAM. The low noise margins of GaAs would demand very accurate macromodels since errors in the modeling can reduce the already low circuit noise margins. As transistor equations evolve to cope with the ever-shrinking channel lengths, the long-term cost associated with maintaining tran-

32nd ACM/IEEE Design Automation Conference ®

sistor models and a circuit simulator within the compiler becomes very large.

A final deficiency of conventional approaches to delay modeling and transistor sizing used in CMOS RAM compilers is that they lack the fundamental notion of process tolerance. The large process variations in GaAs make it essential to use a methodology that is rooted in process tolerant design. Even CMOS manufacturers are finding that as feature sizes shrink below 0.35μm, nonuniformity in channel lengths are making it very important to invest time and effort in modeling process variations.

These issues have led to the development of a design framework for the compiler that makes extensive use of SPICE simulations and relies on a functionality and a timing-constraint driven design methodology.

In Section II, we briefly describe the SRAM produced by the compiler. The objective functions and constraints involved in the transistor sizing problem are formally defined in Section III. This definition has guided the development of the compiler framework, which is described in Section IV. In Section V, we show examples of RAMs that were generated with this compiler. Finally, concluding remarks are offered in Section VI.

II. Circuit Description

The Aurora RAM compiler has read and write ports that are accessed synchronously during each clock cycle. Both 1-read, 1-write and 2-read, 1-write configurations are supported. A memory clock signal is used to initiate read or write operations at the beginning of each cycle. This signal is used to generate an internal equalization pulse which is used to precharge the bit lines. Details of the circuits used in this compiler can be found in [4] and [5].

The major cells used in the RAM are shown in the block diagram of Fig. 1. The pulse generator, sense-amplifiers, write circuitry, equalization circuitry, memory cell array, cell-ground drivers, and word-line drivers are tiled so that routing is achieved by simply abutting the cells. The row and column address buffers and predecoders are made using standard cells and are routed using automated place-and-route tools.

The layout generators for these cells were written using CDA's Compiler Development System. This environment provides a layout generation language which readily facilitates variable transistor sizing. This allows transistors in each of the cells to be sized independently so that the compiler can use buffer sizes that are most appropriate for the size of the generated RAM.

Extensive simulations have shown that fifteen transistor sizes and ratios have significant impacts on the noise margins, functionality, and overall speed of the RAM. In the next section, we define the problem of finding these sizes.

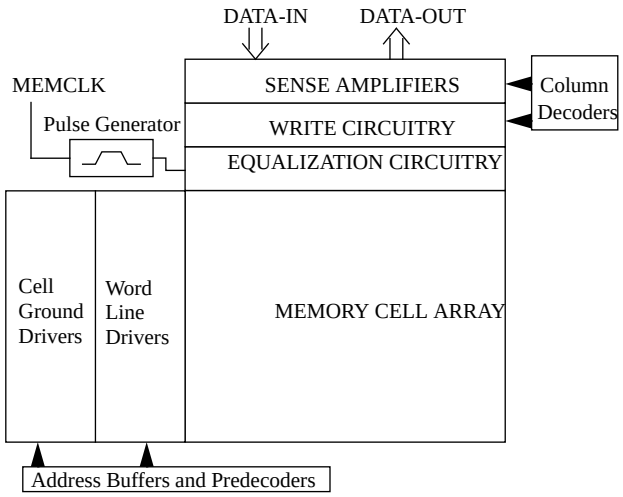

Fig. 1. Block Diagram of the RAM Layout.

III. Transistor Sizing Problem Definition

The transistor size selection problem can be defined as follows. We are in search of a set of n transistor sizes, $x_1,x_2,..,x_n$, which minimize an objective function f of these n sizes, subject to a number of constraints. Formally, this can be written as a search for

$$min \cdot f(\mathbf{x})$$

where

$$f(\mathbf{x}) = P(\mathbf{x}) \qquad \text{(O-1)}$$

subject to

$$t(\mathbf{x}) = max(t_{access}, t_{write}) \leq t_{target}$$

where $\mathbf{x}$ is the vector of transistor sizes $x_1,x_2,..,x_n$, $P(\mathbf{x})$ is the power dissipated by the RAM, $t(\mathbf{x})$ is the larger of the access time and the write time, and t_{target} is the target clock period for the RAM.

The solution to the objective function, O-1, represents a RAM that will achieve the desired access and write times while minimizing the power-delay product.

To guarantee that the vector of n sizes, $\mathbf{x}$, will produce a process tolerant, functional design, the objective function needs to be minimized with respect to a number of additional constraints. Before introducing these constraints, we present some definitions.

We define the operation of successfully writing data D to a cell in row i and column j, given a previous operation or condition Y, as

$$W_D(c_{i,j}) \mid Y$$

The write operation is said to be successful if the cell is properly written to, and all cells not being written maintain their state.

Similarly, the output voltage of a sense-amplifier after reading data *D* from a cell in row *i* and column *j*, following a previous condition Y, is defined as

$$R_D(c_{i,j}) \mid Y$$

To perform process-tolerant design, the functionality constraints must be met at each process corner that defines the process space. Thus, if our process space is defined by *k=1,2,...,m* process corners, and we have *n* different functionality constraints, then $m \cdot n$ constraints must be satisfied to produce a process tolerant functional design.

Using this notation, the functional constraints, ***g(x)***, that must be met by the RAM can be defined as

$$g_{1,k}(\boldsymbol{x}) : W_0(C_{i,j}) \mid C_{i,j} = 1 \quad \text{(C-1)}$$

$$g_{2,k}(\boldsymbol{x}) : W_1(C_{i,j}) \mid C_{i,j} = 0 \quad \text{(C-2)}$$

$$g_{3,k}(\boldsymbol{x}) : W_1(C_{i,j}) \mid C_{i,j} = 1 \quad \text{(C-3)}$$

$$g_{4,k}(\boldsymbol{x}) : W_0(C_{i,j}) \mid C_{i,j} = 0 \quad \text{(C-4)}$$

$$g_{5,k}(\boldsymbol{x}) : (R_1(C_{i,j}) \mid W_0(C_{z,j}) \geq V_{IH}) \quad \text{(C-5)}$$

$$g_{6,k}(\boldsymbol{x}) : R_0(C_{i,j}) \mid W_1(C_{z,j}) \leq V_{IL} \quad \text{(C-6)}$$

$$g_{7,k}(\boldsymbol{x}) : (R_1(C_{i,j}) \mid W_1(C_{z,j}) \geq V_{IH}) \quad \text{(C-7)}$$

$$g_{8,k}(\boldsymbol{x}) : R_0(C_{i,j}) \mid W_0(C_{z,j}) \leq V_{IL} \quad \text{(C-8)}$$

$$g_{9,k}(\boldsymbol{x}) : (R_0(C_{i,j}) \mid R_1(C_{z,j}) \leq V_{IL}) \quad \text{(C-9)}$$

$$g_{10,k}(\boldsymbol{x}) : (R_1(C_{i,j}) \mid R_0(C_{z,j}) \geq V_{IH}) \quad \text{(C-10)}$$

$$g_{11,k}(\boldsymbol{x}) : W_0(C_{i,j}) \mid \{C_{i,j} = 1, C_{z,j} = 0\} \quad \text{(C-11)}$$

$$g_{12,k}(\boldsymbol{x}) : W_1(C_{i,j}) \mid \{C_{i,j} = 0, C_{z,j} = 1\} \quad \text{(C-12)}$$

$$g_{13,k}(\boldsymbol{x}) : W_1(C_{i,j}) \mid \{C_{i,j} = 0, C_{z,j} = 0\} \quad \text{(C-13)}$$

$$g_{14,k}(\boldsymbol{x}) : W_0(C_{i,j}) \mid \{C_{i,j} = 1, C_{z,j} = 1\} \quad \text{(C-14)}$$

$$g_{15,k}(\boldsymbol{x}) : V_{revbias} = V_{word} - V_{cell,storage} > leak \quad \text{(C-15)}$$

$$g_{16,k}(\boldsymbol{x}) : (\boldsymbol{x}_{min} \leq \boldsymbol{x} \leq \boldsymbol{x}_{max}) \quad \text{(C-16)}$$

These 16 constraints are applied for all m process corners that define the process space, i.e., *k=1,2,..,m*. In C4, C5 and C9-C14, we specify that $z \neq i$, while in C6 and C7 there are no constraints on *z*.

The first four constraints are very basic and state that the write operation should be able to store a value in a cell regardless of the previous state of the cell. Constraints five through eight ensure that a read operation is successful when the read occurs immediately after writing the opposite data or the same data to another cell in the same column. The inequality in these constraints ensures that the sense-amp behaves functionally and that the output voltage of the sense-amp meets minimum noise margin requirements. Constraints nine and ten assure that the sense amplifiers can successfully read either data value from a cell after reading the opposite value from a cell in another row of the same column.

Constraints eleven through fourteen are related to an effect specific to the memory cell used in this design, in which the states of the memory cells within a column influence how low the bit lines can be brought during a write. If all of the cells in a column store the same data value, then when the bit line associated with the high-voltage storage node side is pulled down, it is prevented from being completely lowered to ground. This can cause slow or incorrect write operation.

Constraint fifteen is used to minimize the leakage currents associated with unselected rows during a read or write operation. Arbitrary scaling of the transistors in the memory cell, word line driver and cell-ground driver may reduce this reverse bias. In order to produce a RAM tolerant of this known problem, constraint fifteen is used to ensure that the reverse bias achieved across process variations maintains a minimum, predetermined value. The final constraints place physical bounds on the sizes of transistors chosen.

The transistor sizing problem has been cast as a classical nonlinear optimization problem, where we are trying to optimize an objective function *f*, (i.e., the power dissipation) of a set of *n* transistor sizes that will satisfy a number of constraints, *g*, that include achieving a target read access and write time, and ensuring functional behavior over a range of possible failure modes, given the physical bounds on the transistor sizes that can be used.

A number of techniques can be used to solve such a problem. Since the nature of the solution space is well understood, a simple gradient search approach was taken to guide the transistor size selection. This approach is described in the next section.

IV. Compiler Framework

A flow-chart showing the organization of the RAM compiler is given in Fig. 2. The compiler takes as input a Verilog description of the RAM. This description includes a specification of the number of rows and columns in the RAM, and the number of address bits. The user must also specify a target cycle time.

The RAM compiler consists of two main software modules. They are the physical design module and the transistor sizing module. Considerable communication takes place between these modules.

An important feature of these modules is that they operate in a flexible framework that can adapt to changes in processing technology. This feature is illustrated by items in the shaded boxes in the flow chart which denote process definition files.

The transistor sizing module, for instance, uses the transistor HSPICE models and process spread information as input. Since the transistor sizing and RAM characterization rely on this information, the compiler can be used to construct memories in a different process technology that has different transistor transconductances, thresholds, or process control by simply using a new transistor model file or a new process spread file.

As a second example of its flexibility, the compiler performs iterative transistor sizing, parasitic extractions, and simulations to achieve desired delay goals. The extraction is performed using an inter-layer capacitance file and a metallization sheet-resistance file. Thus, transistor sizing decisions, delay calculations, and power dissipation calculations can readily adapt to process changes such as different metallization thicknesses or changes in dielectrics, by updating the appropriate process-files.

The compiler produces three outputs. The first is a design-rule-correct and layout-versus-schematic correct layout of the RAM. The second output is a SPICE net-list for the RAM in case further verification is desired. The third output is a delay-annotated Verilog file containing the read and write times, and address and data setup and hold times. In the following sections, we describe in greater detail the physical design module, a circuit simulation used to characterize the generated RAM, and the transistor sizing module.

A. Physical Design Module

The physical design module consists of a layout generator, a capacitance extractor and a cell area program. These programs take as input the Verilog description of the SRAM which contains physical organization information, and a file containing transistor sizes and power rail sizes.

Since the RAM is a very regular structure, the extraction of capacitances on the bit-lines, word and cell-ground lines, predecode address lines, and a handful of control signals is adequate for accurate circuit simulation.

The dimensions of individual cells and of the entire SRAM array are calculated in the cell area program by using the design-rule independent variables, the transistor sizes, and the

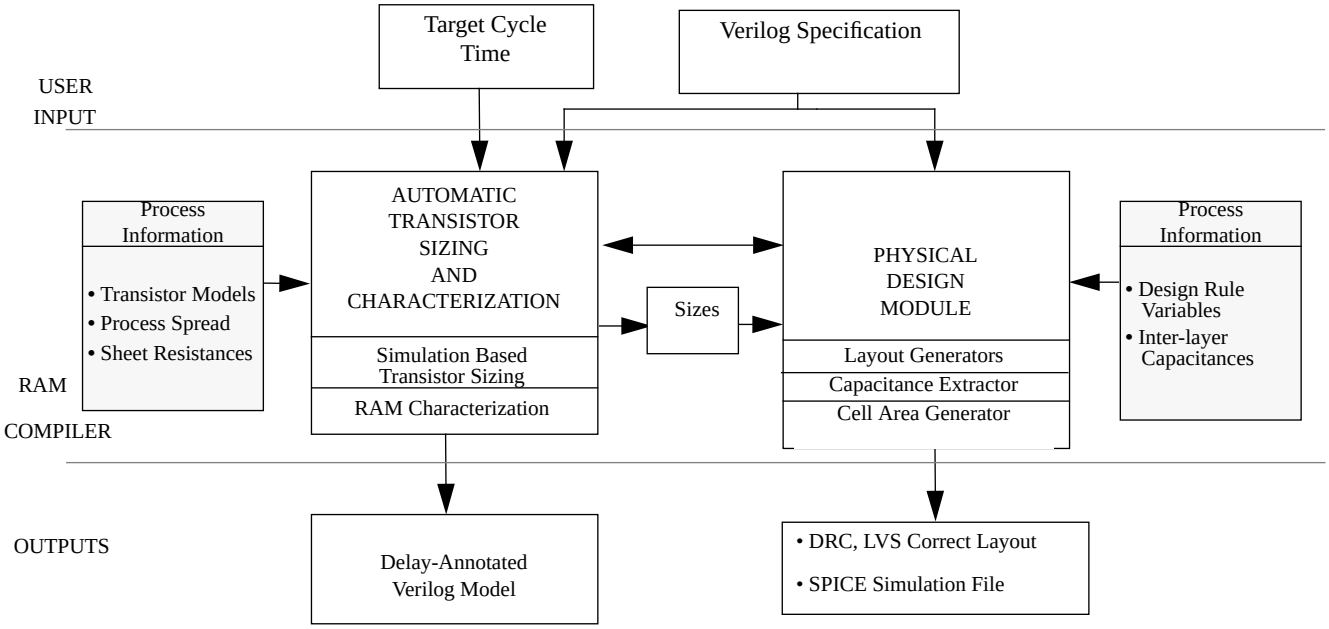

Fig. 2. Compiler Structure Flowchart.

power-rail sizes. This process is very rapid and does not require any layout generation.

At various stages in the sizing, there are a number of different ways in which the circuit can be sized to improve performance. The components of the physical design module are used to perform accurate circuit simulation to help guide the transistor sizing.

B. Circuit Simulation

In Section III, a number of constraints were defined which ensure a process tolerant, functional memory design. In this section we describe a simulation that has been developed which allows these constraints to be checked using only a small number of simulation cycles.

Due to the symmetrical nature of the memory cell and the read and write operation, a number of pairs of constraints can be collapsed into individual constraints. Also, a number of constraints can also be checked simultaneously.

A timing diagram of the simulation used to check these constraints is given in Fig. 3. The sequence of operations consists of two writes followed by three reads followed by two writes and one read.

The first two writes check that the same and the opposite type of data can be written to a cell. These simulations also check the ability of the memory to write with and against the grain of all of the cells in the column. The first read is to the same address that was just written. This tests the seventh and eighth constraints, reading after the same data was written to a cell in the memory. This also checks the ability of the sense-amplifier to read a zero. This set of three reads performs a read-0, read-1, read-0, checking constraints C-9 and C-10. The following two writes cause new data to be written to each of the memory cells. Thus, after these operations, we have checked for writes which cause a change in data with and against the grain of the data stored within the column. The last read is a read after a write of the opposite type of data within the column. Using symmetry, this checks for constraints C-5 and C-6.

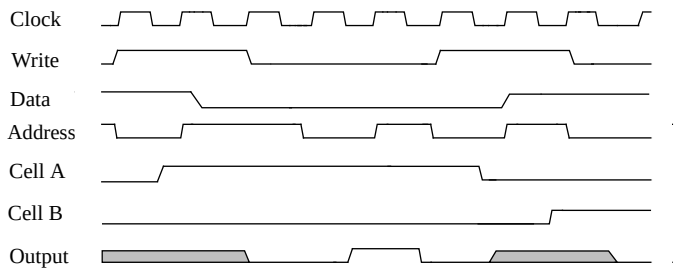

Fig. 3. Timing Diagram of the Simulation to Verify the Functionality Constraints.

In addition to checking the functionality constraints, this simulation is also used to properly determine worst-case read and write times. By subjecting the memory to a range of operating sequences, we can measure the read access time after a write of the same or opposite data type, and after a read of the opposite type of data. The write times can similarly be measured with and against the grain of data in a column, and immediately after a read or immediately after a write. This characterization methodology is much more comprehensive than the methodologies currently employed which use a single “worst-case" operation sequence for measuring these delays. While it can be argued that a particular sequence of operations tends to produce the "worst-case" delay, the different sensitivities of circuits to process variations make the validity of these arguments suspect.

C. Transistor Sizing and Characterization

The purpose of the transistor sizing and characterization module is to find a set of transistor sizes that achieve a process tolerant SRAM meeting a target clock frequency. The module consists of an initialization stage, a transistor sizing loop, and a post-processing stage.

All of the buffers are first set to their minimum sizes during the initialization stage. Parasitic resistances and capacitances are then extracted to generate an accurate simulation model of the RAM. The simulation described in the previous section is used to exercise this model to determine propagation delays, signal edge rates, power dissipation, functionality, and noise margins in the RAM.

In the timing-driven transistor sizing loop, we first check the edge rates on the buffered address lines, read/write lines, and predecode lines. If the rising or falling edge rate for a given line is too large, then the size of the buffer driving that line is increased. The edge rate of a signal can have as much as a 40% impact on the delay of subsequent stages. As a result, poor edge rates early in the readout or write paths can significantly degrade the performance of the memory. The specification of a maximum edge rate on these lines is a sound approach to minimizing the delay of the first few stages of logic.

The second step is to determine which of the read access time or the write time is more critical. Depending upon which time is more critical, a different set of transistor sizes is considered for variation.

There is a large amount of inherent parallelism in selecting which transistor size will be the most cost effective in providing improved performance. Thus, we have developed a framework which first finds the least loaded work stations on a network. Parallel simulations are then distributed to these machines to minimize the total computation time. The re-

sults of these simulations are then analyzed to choose the appropriate transistor sizing direction.

If any of the circuitry associated with the bit-lines is altered, this may impact the desired sensitivity range of the sense-amplifier. Hence, the sense-amplifier is re-optimized within the loop. Since an increase in the speed of the path may require a modification in the precharge pulse width to achieve better performance, the pulse width is also re-optimized within the loop.

Once the target access and write times are achieved, or all possible circuit parameters have been improved, or no improvement could be found for the more critical of the read and write time while maintaining the specified noise margins, we exit this loop to perform the post-processing.

In the post-processing stage, the address and data setup and hold times are determined by sweeping signal edges until one of the first fifteen constraints is not met.

V. Examples Of Generated RAMs

The RAM compiler was used to attempt to generate 1ns parts for various organizations, including a 128x64, 64x64, 64x32, 32x32 and 32x16 organization. The power-delay curves that were traced while traversing the transistor size space are shown in Fig. 4. This graph shows that for larger sized memories, the tool allows a considerable amount of speed to be traded for power dissipation. Fig. 5 shows layout plots for three different sizes of the RAM. The entire layout and routing time for the largest of these SRAMs took only 6 minutes. Each of the SRAMs was optimized in approximately 5 hours, using a small network of six HP 710 machines.

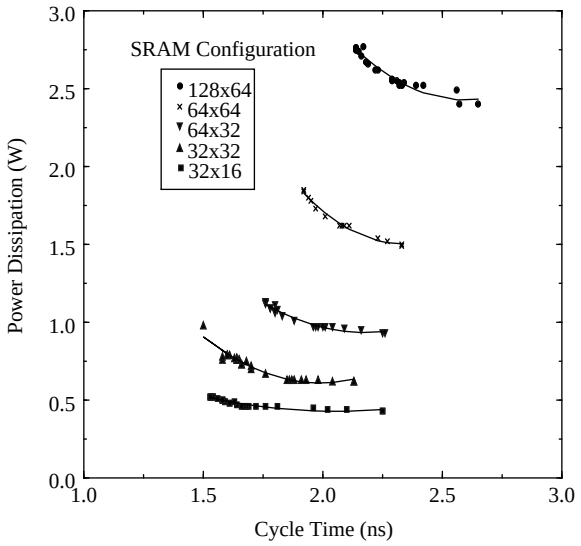

Fig. 4. Compiler-Generated Power-Delay Curves.

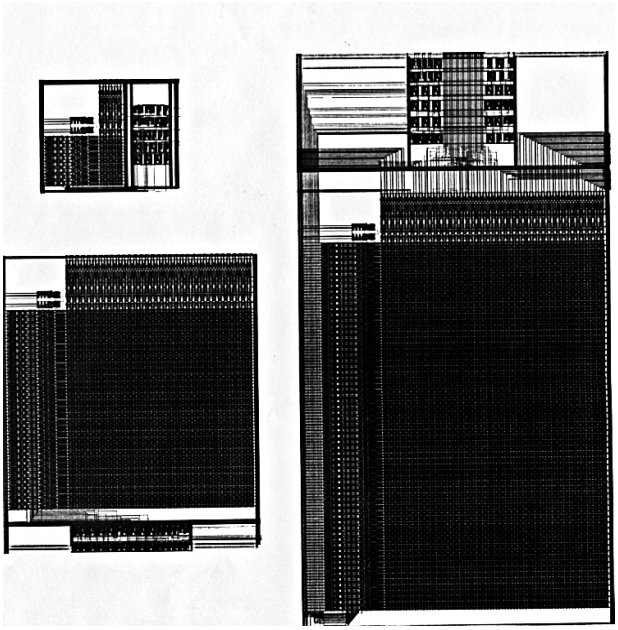

Fig. 5. Sample Compiler Generated Layouts (clockwise, starting at the top left corner) 256b, 8Kb, and 4Kb SRAMs.

VI. Conclusions

In summary, the RAM compiler described in this paper represents a departure from the conventional methodologies used in RAM compilers. Because of the low noise margins and large process variations in GaAs circuits, we developed a compiler which uses HSPICE as a simulation engine for performing delay as well as signal noise margin calculations. A process tolerant design flow has been developed for this RAM compiler which performs simulations over a sequence of operations and process corners to achieve a high design yield. This new approach should prove useful not only for GaAs SRAM compilers, but also for CMOS compilers as transistor nonuniformities become problematic with feature sizes below 0.35μm.

References

[1] W.P. Swartz et al., “CMOS RAM, ROM and PLA Generators for ASIC Applications,” in Proc. 1986 Custom Integrated Circuits Conference.

[2] H. Shinohara et al., “A flexible multi-port RAM compiler for datapath,” in Proc. IEEE 1990 Custom Integrated Circuits Conference.

[3] J. Tou et al., “A Submicrometer CMOS Embedded SRAM Compiler,” IEEE Journal of Solid State Circuits, pp. 417-424, March 1992.

[4] A. Chandna and R.B. Brown, “An Asynchronous GaAs MESFET Static RAM Using a New Current Mirror Memory Cell,” IEEE Journal of Solid State Circuits, pp. 1270-1276, October 1994.

[5] A. Chandna, “GaAs MESFET SRAM Design For Embedded Applications,” PhD. Disseration, University of Michigan, 1995.