ARCHITECTURE OF LOW-POWER EMBEDDED ROMS

Arnaud Turier◇, Lotfi Ben Ammar◇ and Amara Amara*

◇ATMEL-ES2, Zone industrielle, 13106 ROUSSET Cedex, FRANCE
* ISEP/LIP6, 21 rue d'Assas, 75270 PARIS Cedex 06, FRANCE
email : aturier@es2.fr

Abstract: In ASICs, memories take an important part from the power consumption point of view. Large and low-power embedded memories are today required in many designs. This paper describes a new ROM compiler dedicated to low-power embedded applications. Several circuit techniques for reducing the power consumption such as the multi-block architecture, selective precharge, pipe-line decoding, auto-timing structure have been used. The design methodology and the user interface of the compiler are presented. Achieved results are satisfactory compared to those obtained with a former compiler.

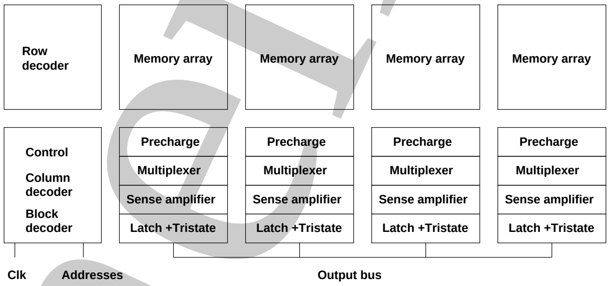

Fig. 1. Multi-block architecture

I. INTRODUCTION :

In the era of System-On-Chip, embedded memories such as ROMs to store a microcontroller code, SRAMs for LCD-drivers or a data cache memory [1] [2], are becoming key components. For embedded applications, density and power consumption are very important goals to reach. Reducing the power consumption of the memories is a necessary step to reduce the power consumption of the whole chip. Moreover, as the time to market is getting shorter and shorter, it is necessary to be able to characterize and generate the memories as soon as possible. From this point of view, the compiler approach seems to be the most suitable : when the design of the circuit is starting, system designers know immediately the size, power consumption, and the access time of the memory. All the necessary views (symbolic, functional, abstract) are ready to be used in the design flow.

In this paper we present a ROM compiler for low-power embedded applications. In the first part we describe the circuit techniques to reduce the power consumption. The design and validation flow of the compiler and the user interface are presented in section IV. In section V, a comparison of the characteristics and the performances is done with the former ROM compiler [3].

II. ARCHITECTURE

A. Block partition and divided word-line

Traditional architecture of a ROM is composed of four parts : a control block, a row decoder, a memory array and a datapath (column decoder, sense amplifier and output stage) [4]. This kind of architecture is no longer suitable for large memories since global signals such as word-lines, have to drive a huge number of connected devices. In this case, the capacitance is dramatically increased and both power and delay are affected. In order to reduce both power consumption and delay, the ROM is partitionned into several blocks (Fig. 1).

All bits of a given word are in the same block, so during a read access, only one block needs to be activated, while the others can be idle. The memory is no longer organized into 2 dimensions but into 3 : X for the row decoder, Y for the column decoder and Z for the block decoder. We use the divided word-line concept [5] instead of the hierarchical word-line concept [6] since the number of blocks driven by the word-line decoder is never greater than four.

By using the divided word-line concept (Fig. 2), we reduce the capacitance of the word-line : the whole capacitance mainly depending on connected device gates is shared out inside each block. A local decoder drives a reduced number of gates and the global decoder drives a few number of AND gates. However the routing capacitance remains unchanged.

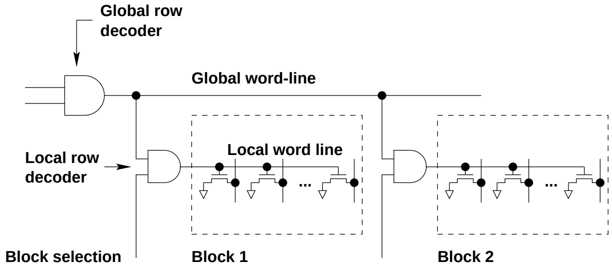

Fig. 2. Divided Word-Line concept

This concept of signal division has been extended to all the signals : command signals and buses. Global buses go through all blocks and a signal from the block decoder enables the right block to activate. Thus, only one datapath has a dynamic consumption during a read cycle.

B. Selective precharge

Usually, a read access starts with a precharge of the whole memory array [8, page 162], that is the case with the former compiler. Thus, in a block, some bit-lines are charged but not used because they are not selected by the multiplexer and they may be discharged if they are connected by a NMOS device to the word-line. In Fig. 3, black dots represents the NMOS devices connecting the word-line to the bit-lines. The power used to charge these bit-lines is lost. In order to avoid this problem, we use the selective precharge concept [7] : only the useful bit-lines for a read access are precharged. And the precharge is conditioned by an internal clock since we use a dynamic circuitry for the precharge. There is not just a signal from the control to enable the precharge in a block, but also a precharge bus to determine the bit-lines to precharge. Since the precharged bit-lines and selected bit-lines are the same one, we use the same decoder (the column decoder) for both the precharge and multiplexer parts. Because we want to reduce the bus capacitance of the precharge signals, we place the precharge part just between the memory core and the multiplexer, as depicted in Fig. 3, and not on the top of the memory as we can see it in some designs [6]. As we want to use the memory at low-voltage closed to Vt, we use PMOS devices to do the precharge in order to have the maximum voltage on a precharged bit-line.

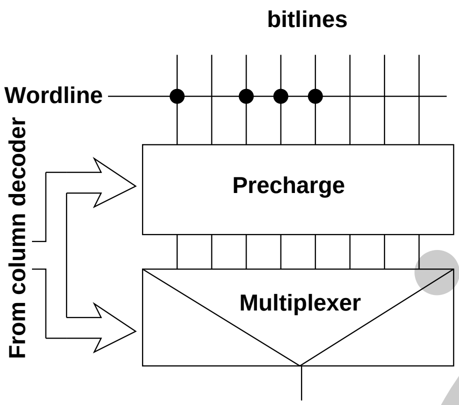

Fig. 3. Selective precharge

C. Pipe-Line

The drawback of selective precharge is that we need to decode the address before starting the precharge cycle. This is a lack of time for both cycle and access time. To limit the effect of this drawback, we use a 2-stage pipe-line. The first stage performs the decoding of the addresses and the second stage precharges and reads the bit-lines. So when the memory is activated, both the column and block decoding are already done and the precharge phase can start immediately. In Fig. 4 the pipe-line structure is described. When the signal activation, Ck, is low, the addresses go through the first latch stage and the decoding is performed. When Ck rises, the decoded values are taken into account for the precharge and read cycles since the second latch stage is enabled. At this time the values of the addresses outside the memory can change because the first latch stage is locked.

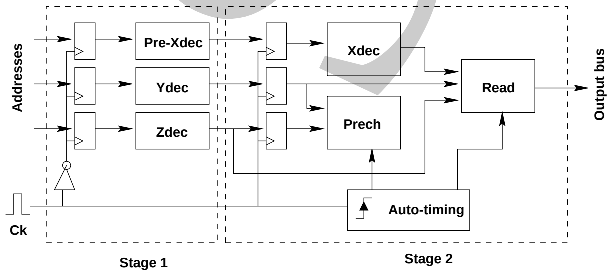

Fig. 4. Pipe-lined decoders

D. Auto-Timing

In static memories, a precharge cycle is starting just after the end of the reading cycle, in order to precharge the memory array for the next read access. Since we use the selective precharge and a pipe-line structure, we precharge just the bit-lines to read. So, the conventional management of the reading and precharge cycle is modified : the precharge cycle is starting just before the read cycle and it has to be shorter as possible, in order to have the lowest impact on the read access time. With this new management (Fig. 5), the cycle time remains unchanged whereas the access time becomes quite longer as the cycle time. Moreover, by limiting the during of the precharge cycle, we limit the leakage current through the memory array. Thus, at the end of a read access, all the internal signals are returning to an idle state and the power consumption is decreased.

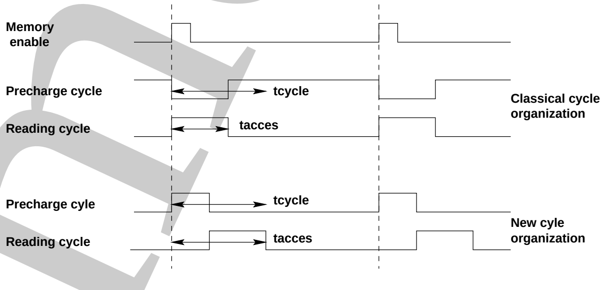

Fig. 5. Precharge and reading cycle management

As our ROMs are from a compiler, the number of rows and blocks changes from a configuration to another one. Because we want to avoid partial precharge and because we want to be independent from the technology process used (the compiler is currently used with different processes), we have designed a self-timed structure to activate the various signals in the memory. The during of both precharge and read cycles are automatically determined by using a dummy bit-line. This dummy bit-line represents the worst case for a bit-line inside the memory array since the maximum of the NMOS devices are connected to this bit-line. Moreover the cycle ratio of the memory activation signal, Ck, is independent of the time to precharge and read.

III. Design flow and user interface

A. Compiler design

Logical schemes of several configurations are created and validated by simulation to ensure the functionality of the memories. Then, the sizes for the devices are determined to reach the objectives about delay and power consumption. Several simulations using Spice on critical paths are required. From the schemes with the sizes for the devices, the design of the layout is starting. Then, conventional checks are performed : design rule checking (DRC), extraction and a layout vs. schematic (LVS) comparison. Spice simulation with full extracted blocks enables to take into account the influence of parasitic capacitances on both delay and power consumption. Many iteration may be used to reach the requirements as shown in fig 6. The leaf cell layouts are hand-packed to have the maximum density.

As it is not possible to check all the possible configurations, a set of critical configuration is defined and validated on a testchip.

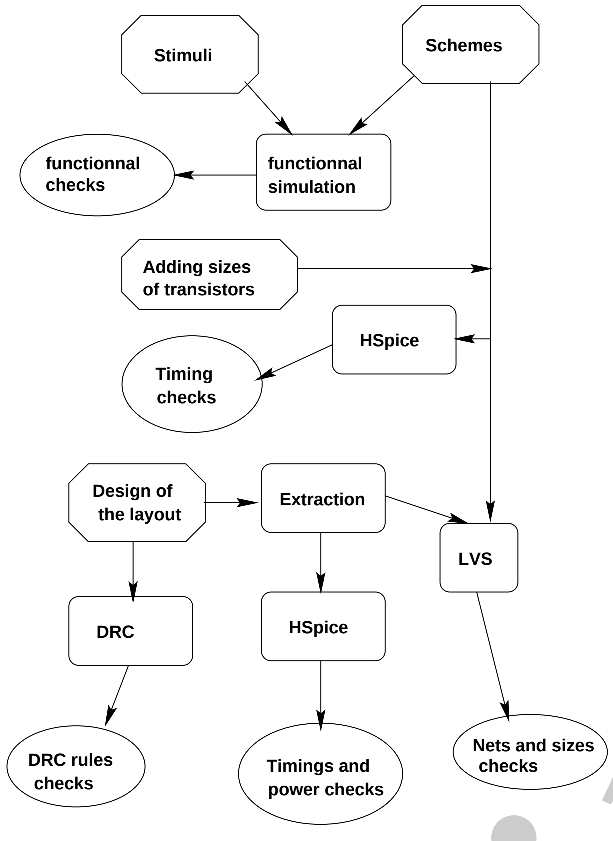

Fig. 6. Design flow

B. Characterization

Data (timings and power consumption) given by the compiler have ever been computed during the simulations achieved on critical paths. These critical paths are parametrable to take into account the flexibility of the compiler such as the number of rows, columns, blocks and aspect ratio. A model based on controlled current generators is used to represent the dynamic capacitances of the devices [9]. These data are matched with measurements on a testchip.

C. User interface

The user interface, written in C, enables the designer to set the characteristics of the ROM. The size of the memories is from 64kb up to 4Mb and the word-length is 8, 16 and 32 bits. These word-length are the most requested values in our designs since, very often, our ROMS are used to store the code of a micro-controller for an embedded application. For a given memory capability, several aspect ratios are available since we have 4 templates to make the memories. At least 2 aspect ratios are available for a given size. By using different aspect ratios, the delays, the power consumption and the layout can fit the requirements. Fig. 7 shows the different templates used to build the memories. The basic structure in template 1 is folded several times to give the other templates. Each memory array is made of 2 or 4 sub-blocks. The memory capacity is increasing with the number of the template : the smallest capacity (64kb) is generated with the template 1 and the biggest (4Mb) is generated with the template 4.

Several views are generated by the compiler in order to start the design of the complete chip as soon as possible.

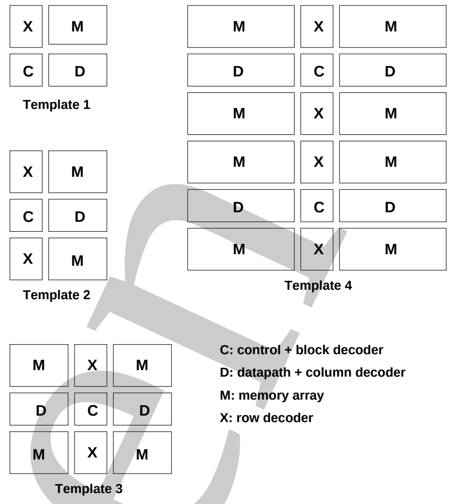

Fig. 7. Templates

The compiler provides a symbol view, a functional model view with delay and constraint timings. An abstract view is generated to start the place and route phase before to have the layout of the memory. We can delay the programmation code of the ROM very late. Figure 8 shows the different compiled views. The leaf-cells are gathered by "Structure Compiler" from Cadence.

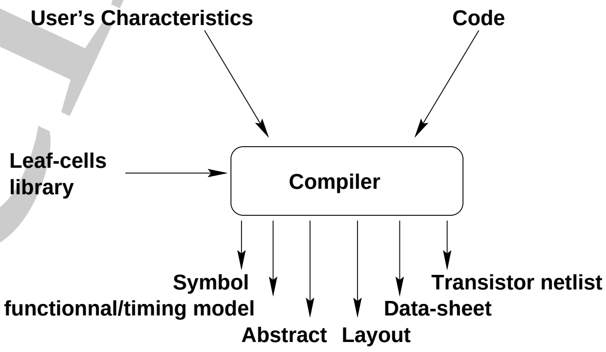

Fig. 8. Compiler design flow

D. User Validation

As not all the possible configurations are checked, the capability is given the designer to check himself the compiled views. A design rule check, an extraction from the layout and a comparison between the extracted layouts and a transistor netlist are performed. The consistency between all the views is checked (names of the pins, direction, etc...)(Fig. 9).

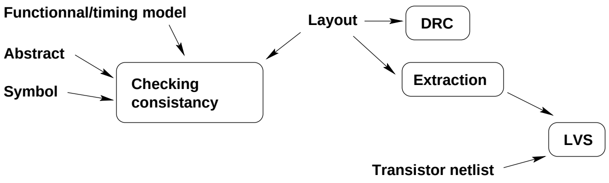

Fig. 9. User checking flow

IV. Results

The new compiler has been designed to support three different technologies : CMOS $0.5\mu m$, CMOS $0.35\mu m$ and CMOS/EEPROM $0.5\mu m$. Tables I and II give a comparison about density, power consumption and speed between the former and the new compiler. The maximum capacity with the former compiler is 128Kb. So, the comparison is done with the smallest configurations of the new compiler. The improvement from the density point of view is achieved by a more compact design of the memory cell and the use of a 3-layer metal technology. On the smallest configurations the gain is about 20% although the memory array represent a few part of the memory. For the highest configurations, the gain is about 92%. Delays are affected by low-power design : gates have often the minimum size and the circuitry to enable just a block, such as gate clocking, induces delay into the structure. From the power consumption point of view, the power has been divided by 22 for a 128 Kb instance. We notice that the power consumed by the former compiler for a 32 Kb instance is equivalent to the power consumed by the new compiler for a 4096 Kb instance. The power has been divided by 128.

With the multi-block architecture we reduce the power consumption of the bit-lines to one block. With the selective precharge, only 32 bit-lines are precharged in block. For example, in 1Mb memory with 16 blocks, 256 columns and 256 rows per block, 32 bit-lines are precharged instead of 4096 with a classical memory structure (1024 rows by 1024 columns).

TABLE I

Former compiler characteristics ($0.35\mu m$ typical conditions)

Capacity (Kb)	Density (Kb/mm^2)	Power (mW/MHz)	Frequency (MHz)
32	133	0.79	183
64	168	1.36	157
128	191	2.49	147

TABLE II

New compiler characteristics ($0.35\mu m$ typical conditions)

Capacity (Kb)	Density (Kb/mm^2)	Power (mW/MHz)	Frequency (MHz)
64	213	0.10	113
128	237	0.11	112
192	291	0.13	98
256	301	0.14	96
4096	368	0.88	71

V. Conclusion

In this paper several techniques to reduce the power consumption on a ROM compiler have been presented. These techniques are process independent since they have been validated with different process technologies. The compiler based on these techniques is portable and can easily be reused. Compared with the former compiler, we have significantly improved the density and the power consumption with an acceptable penalty of the speed.

References

[1] A. Wild, J. Quigley, J. Feddeler, S. Ledford, J. Caravella, *et al.*, "A 0.9-V microcontroller for portable applications," *IEEE J. Solid-State Circuits*, vol. 32, pp. 1049-1055, July 1997.

[2] D. Draper, M. Crowley, J. Holst, G. Favor, A. Schoy, et al., "Circuit techniques in 266-MHz MMX-enabled processor," *IEEE J. Solid-State Circuits*, vol. 32, pp. 1650-1664, Nov. 1997.

[3] ATMEL CBIC AT55K/ECAT05 Library Databook, Sept. 1997.

[4] J. M. Rabaey, "*Digital integrated circuits, a design perspective,*" Upper Saddle River, NJ : Prentice Hall, 1996.

[5] M. Yoshimito, K. Anani, H. Shinohara *et al.*, "A divided word-line structure in the static RAM and its application to a 64k full CMOS RAM," *IEEE J. Solid-State Circuits*, vol SC-18, pp. 479-485, Oct. 1983.

[6] T. Hirose, H. Kuriyama *et al.*, "A 20ns 4Mb CMOS SRAM with hierarchical word decoding architecture," *IEEE J. Solid-State Circuits*, vol. 25 , pp. 1068-1074, Oct. 1990.

[7] N. Weste, and K. Eshraghian, "*Principles of CMOS VLSI Design : A Systems Perspective, Second edition,*" Reading, MA., Addison-Wesley, p. 586, 1993.

[8] B. Prince, "*Semiconductor memories, Second edition,*" Chichester, UK, John Wiley & Sons, p. 8, 1991.

[9] A. Turier, L. Benammar, A. Amara, "An accurate Power and Timing Modeling Technique Applied To A Low Power ROM Compiler," *PATMOS'98*, pp 181-190, Lingby, Denmark, Oct. 1998.