Oil & Natural Gas Technology

275° C DOWNHOLE MICROCOMPUTER SYSTEM
Final Report
TECHNICAL PROGRESS REPORT

For Period

10/1/05 to 8/31/08
by

Dr. Chris Hutchens, Principal Investigator/Program Manager
Dr. Hooi Miin Soo, Principal Investigator
MSVLSI Group ATRC
CEAT-ECEN
Oklahoma State University.
Stillwater, OK 74078

DOE Award No.: DE-FC26-05NT42656

Prepared for:
United States Department of Energy
National Energy Technology Laboratory

.

Stillwater, OK 74078

Office of Fossil

Abstract

An HC11 controller IC and along with serial SRAM and ROM support ICs chip set were developed to support a data acquisition and control for extreme temperature / harsh environment conditions greater than 275 °C. The 68HC11 microprocessor is widely used in well logging tools for control, data acquisition, and signal processing applications and was the logical choice for a downhole controller. This extreme temperature version of the 68HC11 enables new high temperature designs and additionally allows 68HC11-based well logging tools and MWD tools to be upgraded for high temperature operation in deep gas reservoirs, The microcomputer chip consists of the microprocessor ALU, a small boot ROM, 4 kbyte data RAM, counter/timer unit, serial peripheral interface (SPI), asynchronous serial interface (SCI), and the A, B, C, and D parallel ports. The chip is code compatible with the single chip mode commercial 68HC11 except for the absence of the analog to digital converter system. To avoid mask programmed internal ROM, a boot program is used to load the microcomputer program from an external mask SPI ROM. A SPI RAM IC completes the chip set and allows data RAM to be added in 4 kbyte increments. The HC11 controller IC chip set is implemented in the Peregrine Semiconductor 0.5 micron Silicon-on-Sapphire (SOS) process using a custom high temperature cell library developed at Oklahoma State University. Yield data is presented for all, the HC11, SPI-RAM and ROM. The lessons learned in this project were extended to the successful development of two high temperature versions of the LEON3 and a companion 8 Kbyte SRAM, a 200 °C version for the Navy and a 275 °C version for the gas industry.

Keywords: HC11, SRAM, SPI-SRAM, SOS, SOI, high temperature electronics, CMOS.

Table of Contents

1 Overview

This is the final report for documenting the work done by Oklahoma State University (OSU) for producing a down-hole microcomputer system (DMS) capable of operating at 275 °C for 1000 hours. This project was funded by Department of Energy/National Energy Technology Laboratory (DOE/NETL), contract number DE-FC26-05NT42656. The base DMS consists of a 68HC11 single chip microcomputer with boot ROM, static RAM, counter/timer unit, parallel input/output (PIO) unit, and serial peripheral interfaces (Asynchronous Serial Communications Interface (SCI), and Synchronous Serial Peripheral Interface (SPI)). The DMS 68HC11 microcomputer chip consists of the microprocessor arithmetic logic unit (ALU), a small 512 byte boot read-only-memory (ROM), 4kbytes (4Kx8) data static-random-access-memory (SRAM), counter/timer unit, synchronous serial peripheral interface (SPI), asynchronous serial interface (SCI), and the A, B, C, and D parallel ports operating off a 3.3V supply. The peripherals: analog-to-digital (A/D) converter and Port E were not implemented. The main timer system is a 16 bit free running timer; it has three 16-bit asynchronous input-capture lines, five 16-bit output-compare lines, a computer operating properly (COP) watchdog subsystem unit, and a real-time interrupt unit. The COP watchdog is used to protect against software failures. An 8-bit pulse accumulator subsystem is included also. Interrupt handler works with timer, reset and HC11 peripheral units to alert MCU of pending interrupts (priority encoded) and reset asynchronous logic. The internal Boot ROM triggers the CPU to load programs over SPI or SCI into the internal RAM from external RAM or ROM, and it also contains self-test code to assist in peripheral and memory diagnostics for proper functionality. An additional scan chains is built in to support the debugging process during chip development.

The High temperature digital controller and support chips, 68HC11 microcontroller, including a 4k-SRAM and a 2k-ROM with the serial peripheral interface (SPI) interface were submitted for fabrication in the Peregrine 0.5um SOS process to verify the 275 °C functionality of the design. All were tested on wafer for functionality, timing and temperature tolerance, and all were demonstrated suitable for control and data acquisition related applications across the extreme temperature industry. The exception

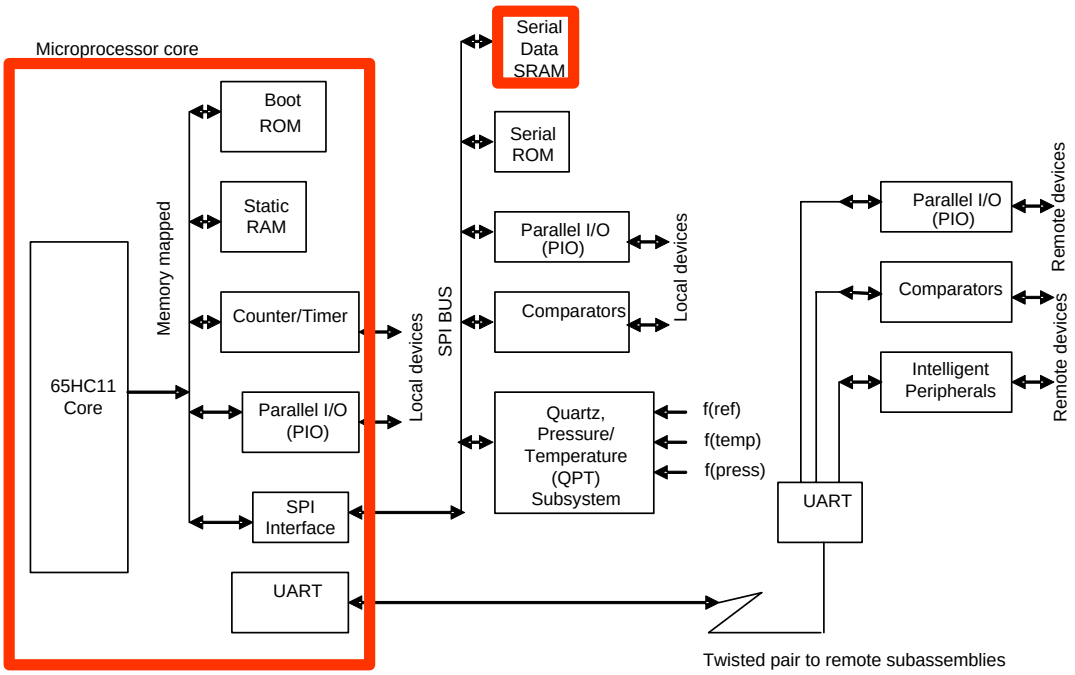


Figure 1. Block diagram of the proposed down-hole microcomputer system that consists of microprocessor, memory devices and serial peripherals.

being a ldaa,x instruction conflict detailed below. Figure 1 shows the DMS uses the SPI serial bus to interface to local peripherals within the electronic control subassembly based on the 68HC11. The HC11 use the SCI serial communications standard to interface to remote devices and slave microcomputer systems in locally remote electronic subassemblies. The 275 °C 68HC11 requires a 3.3 volt supply voltage, and uses a 12 MHz (maximum oscillator frequency), resulting in a 3 MHz instruction cycle time or E clock. The HC11's parallel outputs are designed for a maximum source and sink currents in excess of 10mA, for CMOS output logic levels of V_{OH} = 2.8 volt and V_{OL} = 0.4 volt. The total power consumption of the 68HC11 at room temperature is approximately 67mW at 8MHz with a 3.3V source. At 275 °C, power dissipation, is 72mW at 8MHz and 3.3V and dominated by leakage currents. Across temperature leakage remains constant at approximately 15mA until 275 °C. Switching current at 8 MHz is approximately 7mA and from room to 275 °C., while from 275 °C to 280 °C switching currents increase by 50% and leakage currents increase by a factor of 4 at 3.3V. See

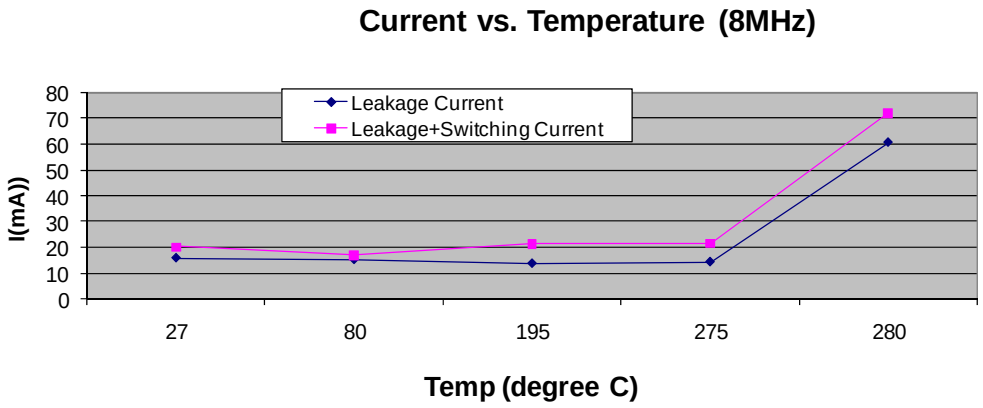


Figure 2. The OSU 68HC11's leakage and switching current of different temperature from room to 275 °C At 8MHz and 3.3V.

figure 2. The 275 ºC HC11 microcontroller utilizes an area of 6.5mm x 6.5mm and has a gate count of 50449 (cells = 19852).

The resulting 8 bit microcontroller is supported by the following; functions and peripheral ports as shown in Figure 1 and 4:

- The central processing unit consists of a state machine controller, a clock control circuit and an ALU implemented using simple 8/16 bit Von Neumann architecture. The M68HC11 Family of microcontrollers uses 8-bit opcodes. Each opcode identifies a particular instruction and associated addressing mode to the CPU. There are a total of 311 68HC11 opcodes. With an 8-bit number spanning 256 values, it is clear that to implement 311 opcodes requires a scheme using a "special" opcode to indicate that the real opcode is in a different table or page. This special opcode is referred to as a "prebyte" since it is a byte which is fetched before the actual page-N opcode. The HC11 controller implements 3 different prebytes which results in “branching” to 3 different auxiliary pages. As implemented, the primary page or page 0, includes 233 valid opcodes plus 3 prebyte codes, resulting in 20 undefined opcodes. If the processor encounters one of these undefined opcodes while running, for example, the number $42 in hex notation will throw an illegal opcode exception. Of the full instruction set the controller does not implement the ADC instructions and the DIV instruction. Figure 3 is a block diagram of the overall architecture of the controller.

The six addressing modes can be used to access memory: Immediate, Direct, Extended, Indexed, Inherent, and Relative.

- The interrupt logic handles 16 maskable (I) interrupts from the timer, SPI, SCI, ports and interrupt request (IRQ) pin (prioritized by the HPRIO<3:0>), 1 non-maskable (X) interrupt from XIRQ pin, 4 resets (e.g., power-on, external, clock monitor, and watchdog, prioritized internally). The software interrupt, illegal opcode and wait are handled by CPU.
- A small masked boot ROM (512 bytes) contains the self-test and bootloader program, which is used to control the HC11 immediately after reset. First, it performs a sequence of peripherals and registers self-test process (which can be skipped by setting the 'slftst' input pin low). Second, it controls the bootstrap process to boot from either SCI or SPI interface.
- A 4kx8 data RAM serves as both program and internal data memory.
- A serial peripheral interface (SPI) unit is implemented for serial synchronous communication to local devices.
- A serial command interface (SCI) unit is implemented for serial asynchronous communication to distant devices.
- The timer unit is a 16 bit free running counter with a programmable 4 bit prescaler. It provides the clock source to the pulse accumulator system. The timer registers are

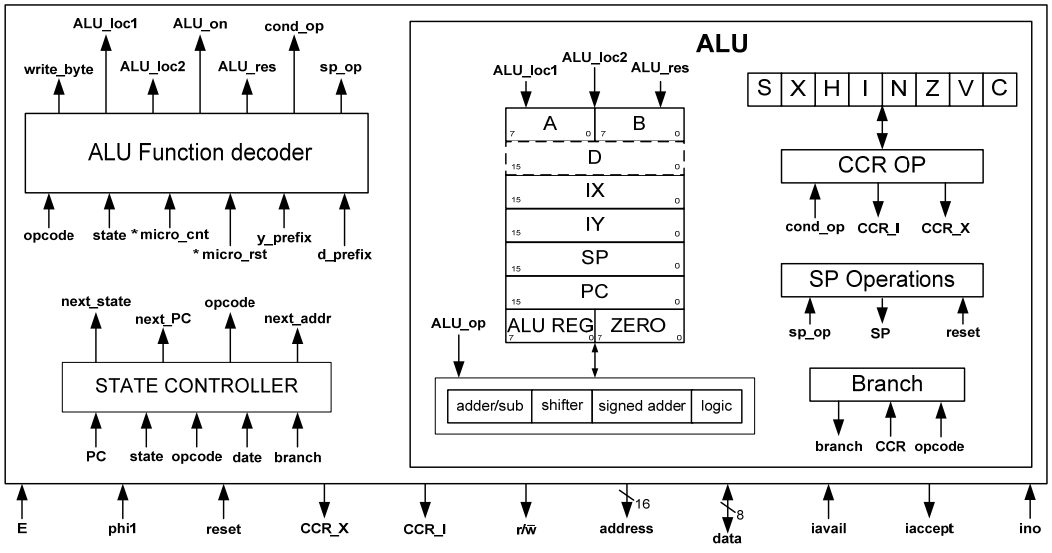


Figure 3. Controller architecture block diagram.

associated with the input captures and output compares function. Three input captures channels are available to capture the current value of the free-running counter when external trigger is detected at the corresponding timer input pin, and up to five output compares channels are available for waveform generation. The computer operating properly (COP) watchdog timer is available to detect the software failure.

- Four parallel I/O ports, port A, port B, port C and port D, are implemented as 8 bit ports except for port D (6-bit). The port A and port D are shared with the timer system and the serial communication system (SPI and SCI) respectively. Port B functions as an output port only. Port C is a bidirectional port. The SPI functions can be either 3.3V or 5V and are externally configured.
- Two serial scan chains are added for debugging/diagnostic during wafer testing.
- An external masked SPI-ROM, is available to hold a small 2K bytes monitor program to be loaded into internal SRAM for testing and debugging as the on-board ROM is only 512 bytes.
- An external SPI-SRAM, is available to provide extra 4K bytes of data memory.

The on-chip small masked boot ROM (512 bytes) consists of boot-up self-test code that both help debug the chip and validate the chip's partial functionality before executing the first line of user code. It consists of the following processes (see Appendix 1):

- Internal SRAM write/read with AA&55.
- All Ports – Loop Port B to C and C to D.
- Loop-Back test a byte over SPI (Internal).
- Loop-Back test a byte over SCI (Internal).
- SELFTEST "pass" – BIT1 of $1001 is set.
- Load code over SCI (Check Null)/SPI.
- Set BOOTSET ($1001[0]) and pin level to indicate boot done .

The self-test code followed by the bootstrap code, which allows the microprocessor to boot from SPI or SCI based on user choice, or boot from SPI if SCI connection fails.

❖ SCI Bootstrap: Initial SCI registers setup, refresh SRAM, and download PC program into SRAM. See Appendix 1.

Pseudo code:

1. Initiate related registers (STACK, SPSR, BAUD)
2. Refresh SRAM.
3. Send BREAK to PC.
4. Wait until START bit detected.
5. Receive data.
6. Download 256 bytes program from PC to SRAM.

- SPI Bootstrap: Initial SPI registers setup, refresh SRAM, and download a small monitor from external SPI_ROM into HC11 internal RAM. See Appendix 1.

  Pseudo code:

  1. Initiate related registers (DDRD, SPCR)
  2. Refresh RAM.
  3. Send Address to external SPI-ROM (write to SPDR).
  4. Wait until SPIF bit set (check SPIF bit in SPSR).
  5. Receive data (load from SPDR).
  6. Repeat step 3 to 5 until the 256 bytes program is downloaded from SPI-ROM to RAM.

The SPI-SRAM and SPI-ROM allow memory to be accessed using the simple SPI compatible serial bus. The SPI-SRAM design has been improved over the initial fabrication run to achieve lower power, and robustness to process variation. The SPI is a serial synchronous communication protocol that requires a minimum of 3 wires; a clock input (sck), data in (di) and data out (do) bus lines. The device is enabled through the chip select enable pin (/csn). The SPI-ROM and SPI-SRAM do not have an on-board clock and require an input clock. The maximum clock rate is 8MHz (E clock) with the SPI clock (sck) being 4MHz, since these are SPI slave devices. A SPI slave does not generate a sck clock, the master supplies the sck clock. The sck clock comes from the master with bit rates (sck) prescaled at 1/2, 1/4, 1/16, 1/32 of processor/system clock (E clock). Both are designed using 3.3V low-power CMOS technology. The SPI-ROM is organization as 2K x 8bit with a 280ns read cycle time (max.). The supported temperature

range is -25^{o}C to +275^{o}C. The SPI-ROM consumes 4.6mW at 3.3V and 275°C and is 2.4 mm x 2.13 mm in area. The SPI-SRAM's organization is 4K x 8bit with read and write times of 280nS and 385nS respectively over a temperature range of -25^{o}C to +275^{o}C. The SRAM consumes 5.8mW at 3.3V and 275°C, and is 2.4mm x 4.791mm in area. The SPI controllers have a gate count of approximately 22950 (cells = 419).

The ROM is a custom mask design. The masking operation was completed separately from the other structures and was programmed using a combination of Matlab and Cadence SKILL language. A small monitor program, 68MON (Appendix 1), resides in the external 2Kbytes masked ROM. 68MON is a small monitor program for 68HC11 for downloading and debugging, was originally written by Keith Vasilakes. Modification to the code was customized for testing the OSU 68HC11, where additional code was added to the 68MON to allow the OSU 68HC11 microprocessor to load from either the SPI or the SCI. The 68HC11 monitor program, 68MON, is designed to allow the user to directly type commands to the program using a terminal emulator (PC running a terminal emulation program such as Windows Hyper Terminal), by connecting a terminal to the serial communication port of the microcontroller. 68MON monitor supports some standard monitor functions and an Intel upload (when defined) for those cases when an Intel hex is used (an s19 file is converted to Intel hex). The 68mon standard monitor functions include string IO character conversion, and serial port support. These functions can be called from a user assembly language program. The modified 68MON version does not support writing to the EEPROM, and auto detecting/changing the SCI/SPI baud rate, *the baud rate was fixed to 9.6K baud.*

1.1 Project Goals

This work was logically sub-divided into two major tasks: 1) megacell design, validation, fabrication and testing and integration and, 2) 68HC11fabrication (Appendix 2 through 8) and 2 peripheral memory chips (Appendix 9 and 10), a Data RAM and Mask ROM. The megacell design involved designing 23 megacells using OSU's high temperature SOI standard cells. These megacells are the basic functional building blocks of the 68HC11 microcomputer (see Figure 4) and the two peripheral RAM and ROM components. These megacell designs were implemented either using publicly available VHDL (Very

High Speed Integrated Circuit Hardware Description Language) or Verilog to the extent possible to speed up design and reduce debugging, codes generated by OSU or in combination. In addition to design/implement the essential digital blocks in Verilog and OSU's high temperature SOI standard cells, made extensive used of Ambit, Silicon Ensemble, and Virtuoso of Cadence design tools and extensive simulations were conducted to ensure the validity of the design and its layout in silicon. Figure 4 shows a diagram of the proposed OSU HC11 (including the basic functional blocks of the 68HC11 microcomputer composed of number of megacell blocks), and the association

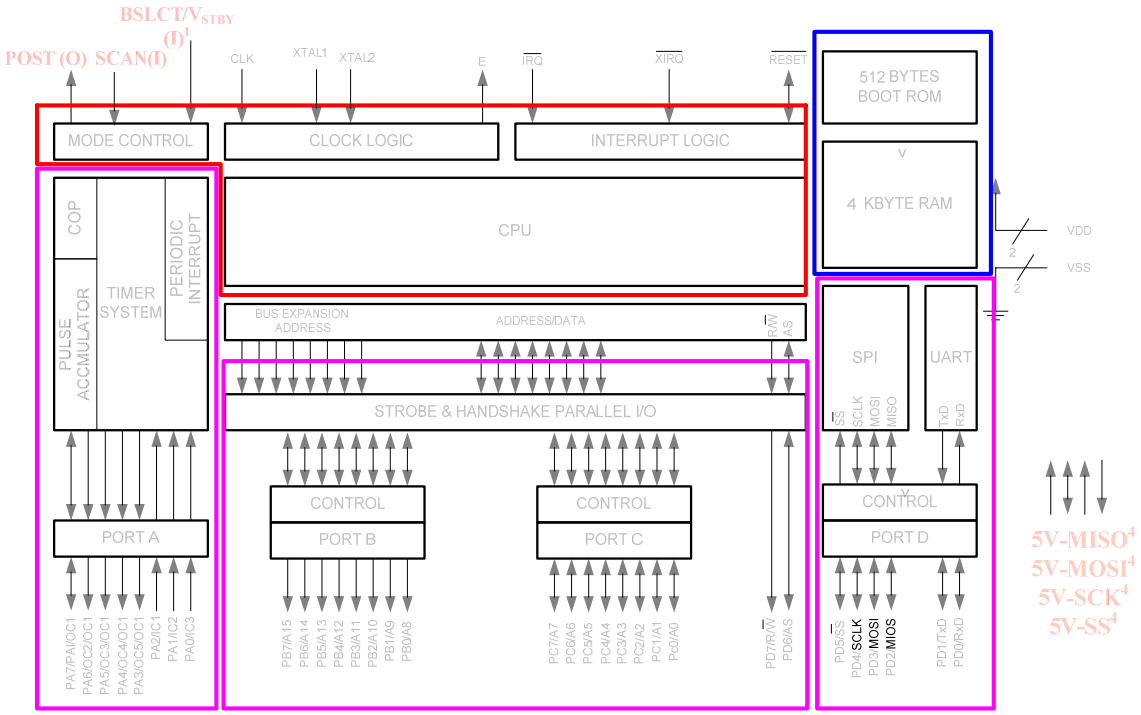


Figure 4. OSU-HC11 system block diagram.

between its major subsystems and I/O pin connections. For the OSU HC11 SPI system, there is a 5V port that is paralleled off the 3.3V function (Figure 4). The additional dedicated 5V SPI pins shadow the SPI function of the Port D 3.3V SPI pins.

The objective of this project was to fabricate wafers containing the fabricated 23 megacells, and 2 peripheral memory chips, a Data RAM and Mask ROM and merge them all into a functional high temperature HC11 microcontroller comprised of the OSU HC11 and both SPI memories. Each megacell and memory chip was tested for static and

dynamic operation. These included tests to verify correct static and dynamic operation at 27° C, 125° C, 175° C, 250° C and 275° C. This was to ensure timing and input/output compatibility of megacells when they were integrated into the full 68HC11 microcomputer and the peripheral components.

The deliverables for this project were the 1) design, 2) layout, 3) fabrication, 4) test, and 5) integration of three integrated circuits suitable for microcomputer based controllers and two peripheral memory chips (Data RAM and Mask ROM) operational at 275 °C operation. This included 50 die of each of the following parts:

1. 68HC11 single chip microcomputer integrated circuit (Figure 4)

   A 68HC11 microcomputer with 512 byte boot ROM, 4K by 8 bit static RAM, counter/timer unit, parallel input/output (PIO) units, and serial peripheral interface (SCI and SPI) units. Process yields prohibited the delivery of OSU HC11 die.
2. Data RAM integrated circuit

   A 4K by 8 static RAM with SPI communications circuit.
3. Mask ROM integrated circuit

   A 2K by 8 masked ROM with on board SPI communications circuit.

The Project resulted in 7 working HC11die and of which there were 3 good HC11 die remaining. “Good” being defined by those passing the start-up self test. A package 15 SPI SRAM die and 25 SPI ROM bare die were initially provided along with 35 and 25 bare die respectively later.

1.2 Project Scope

The project comprised three major tasks: 1) megacell design, validation, fabrication and testing and integration, 2) fabrication of 68HC11 and 2 peripheral memory chips, a data RAM and mask ROM suitable for 275 °C operation and 3) their integration into a high temperature microcontroller. The *first* major task encompassed: 1) design of building blocks constituting three integrated circuits using the MSVLSI group’s high temperature standard cells, and circuit designs for and VHDL code for others, 2) compilation and validation of layouts, 3) submission of designs for fabrication, 4) testing of fabricated circuits at high temperature and, 5) revision and resubmission for fabrication of designs as necessary. The *second* major task encompassed: 1) test of building blocks/megacells

at 275 °C operation and integration into an 68HC11 microcomputer IC, 2) compilation and validation of OSU HC11 and memory layouts, 3) submission of designs for fabrication, 4) testing of fabricated circuits at high temperature and, 5) finally testing of the integrated circuits at high temperature. The final or *third* task was the integration of the three die into a high temperature controller. The final task was not successful. Finally, a final report documenting the work done over the course of this project was written.

The HC11 circuit blocks (Figure 4), and SPI circuits of memory chips are implemented using the Verilog HDL and the OSU 275 °C library with the aid of Cadence design tools. The Cadence Ambit synthesizer takes a circuit description (register transfer level (RTL) coding) given in behavioral and/or structural hardware description language (VHDL or Verilog) and generates a netlist (pure structural description) of the circuit implementation from the logic gates contained and characterized in the high temperature cell libraries. The HC11 have a gate count of 50449 (cells = 19852). The SPI controllers have a gate count of approximately 22950 (cells = 419). The 512 byte boot ROM, 4K by 8 bit SRAM, and 2K by 8 bit ROM were designed using hand-layout, and then instantiated as cells into the final OSU HC11 and the SPI memories.

During the design, debug and revision stage, extensive used of Ambit, Silicon Ensemble, and Virtuoso of Cadence design tools and extensive simulations were conducted to ensure the validity of the design and its layout in silicon across process corners. Each circuit block was functionally validated *without the parasitic capacitance* using Cadence Verilog XL and Xilinx FPGA board. After place and route, each modules functionality and timing was again revalidated *with the parasitic capacitance* included using Cadence Verilog XL (back annotate), and Cadence Ultrasim. The timing and logic consistency between the ALU, interrupt handler circuitry, and the peripherals were verified and checked as well as the timing and handshake between SPI and ROM/SRAM and I/O ports. The final phase of chip set implementation was hardware testing.

1.3 Summary of Work Performed

The period of the work began on October 1, 2005 and was scheduled to end on March 31, 2007. An extension was requested to allow the second full HC11 design to piggy back

on the Switched-Mode Power Supply (SMPS) mask set. This extension ended August 31, 2008. The following tabulation summarizes the performed effort:

1. The initial place and route of the mega cells identified areas of improper function and excessive leakage problems within some of the original library cells. More significantly excessive cell areas forced a full redesign of the library.
2. HC11 building blocks were designed using the reworked library. Verilog code for each module was written. The 512 byte boot ROM, 4K by 8 bit static RAM was designed using hand-layout, and then instantiate the cell to achieve 512 byte boot ROM, and 4K by 8 bit static RAM.
3. Each megacell/building block functionality was validated without the parasitic capacitance using Cadence Verilog XL, and Xilinx FPGA board as was the HC11 *excluding* the scan chains. Each megacell functionality and timing was validated with the parasitic capacitance using Cadence Verilog XL, and Ultrasim.
4. Integrated the megacells into a 68HC11 microcomputer with 512 byte boot ROM, 4K by 8 bit static RAM, counter/timer unit, pulse accumulator unit, interrupt handler unit, reset logic, parallel input/output units, SPI unit and SCI unit. The peripherals were linked to the ALU and interrupt handler circuitry.
5. The timing and logic consistency (with the parasitic capacitance) between the ALU, interrupt handler circuitry, and the peripherals were verified using Cadence Verilog XL (back annotate the parasitic capacitance).
6. The memory cells of the 68HC11's on-chip 512 byte boot ROM, 4K by 8 bit SRAM, and 2K by 8 bit ROM were designed as part of the initial building HC11 building blocks. Following their initial test all memory cells were modified and instantiated as cells in all final memories. The SPI controller section was designed using Cadence tools, and coded in Verilog HDL.
7. Both the internal and external masked ROM used the same architecture. The masking operation was completed separately from the other structures and was programmed using a combination of Matlab and Cadence SKILL language.

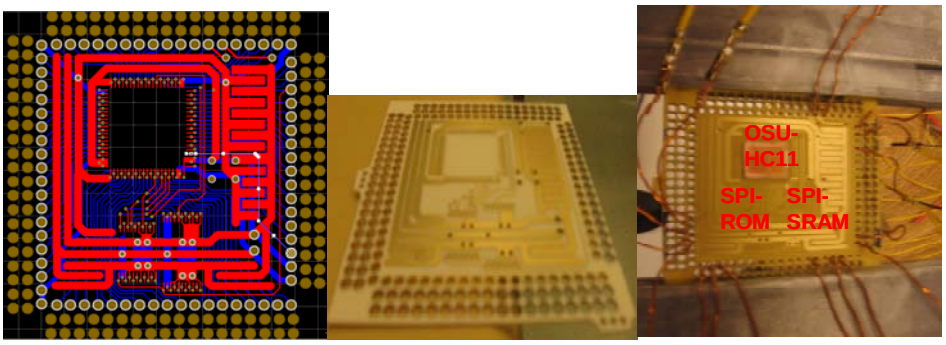


Figure 5. The 68HC11 and 2 peripheral circuits diced and mounted on a thin AlN substrate board for high temperature testing up to and including 275 °C.

8. The designs were delivered for fabrication on May 2006 (1st run), January 2007 (2nd) and February 2008 (3rd run). The 1st run was for verifying the functionality and timing of each module/design block on wafer. The 2nd and 3rd runs had integrated each module into the OSU 68HC11 chip set.
9. Wafer testing was conducted to verify the timing and logic consistency of the 68HC11 core and peripherals. Initial testing was completed on-wafer to verify operation, and then the 68HC11 and 2 peripheral circuits were diced and mounted on an AlN substrate for high temperature testing up to and including 275 ºC (Figure 5). The AlN board proved to be problematic during the test as the resulting AlN board was too flexible and fragile (Figure 5). Thin twin holes (hole pairs) were added to serve as a strain relief function. However, torque on the motherboard from the Teflon

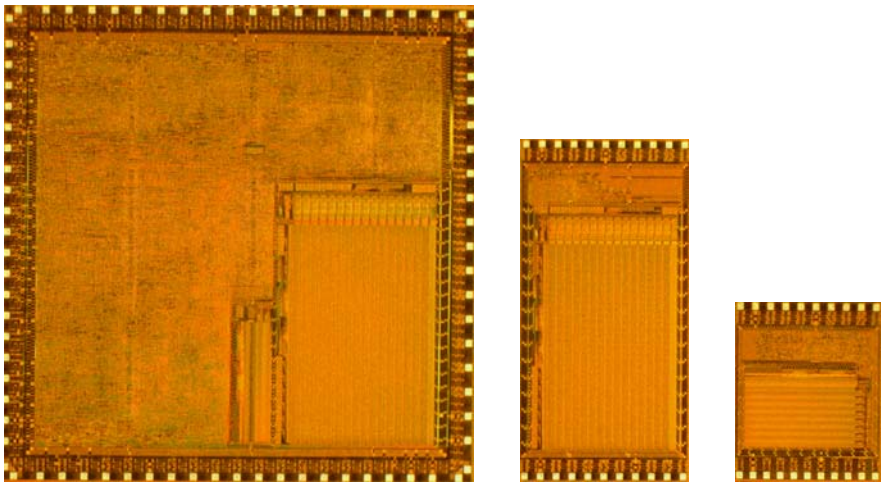


Figure 6. Screen Capture of DMC – 68HC11 (left), the 4K bytes SPI SRAM (middle) and 2K bytes ROM (right).

wires broke the holes from the board in addition to peeling off Au traces. Further research and tests needs to be conducted on a suitable chip on board interconnect system for 275 ºC applications. The only reasonable solution at this point was to lower temperatures to 240 ºC and use a polyamide chip on board solution. This was not attempted due to time limitations. Die testing on the probe station confirmed the basic operation of the OSU 68HC11 and the RAM and ROM peripherals. Die yields for the OSU HC11 are unacceptably low at 5 per cent. Minor timing problems remain with the OSU HC11 in spite of functional testing of the final VHDL on a Xilinx platform as a result of conflicts with the scan chains.

Figure 6 shows a micrograph of the die for the 275 °C 68HC11microcontroller (area of 6.5 mm x 6.5 mm), the SPI-SRAM (2.4mm x 4.791mm), and the SPI-ROM (2.4 mm x 2.13 mm). Note the internal ROM and RAM of the OSU 68HC11 are clearly visible.

2 68HC11 Wafer Testing Results

Wafer testing was conducted to verify the timing and logical consistency of the 68HC11 core and peripherals, SPI-ROM, and SPI-SRAM from room temperature to 300°C with the Cascade Alessi REI-6100 semi automatic probe station and a full complement of pattern generators and logic analyzers.

Figure 7 shows that five per cent of the 153 OSU HC11 die tested as good, functioning correctly, or more correctly passed the self-test and deemed able to load code externally (trigger by the on-chip boot ROM).

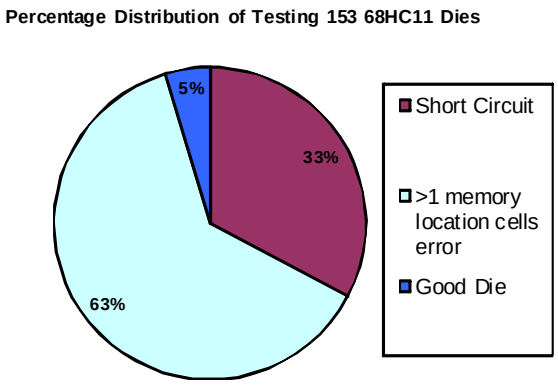


Figure 7. OSU 68HC11 testing results distribution.

Of the remaining 95%, 32% were shorted die as a result of unknown process issues presumably inadequately DRC rules. The m2-m2 spacing in the memory was suspected. The same problem was observed with three different die; on different runs a

LEON3, the 2nd OSU HC11 with SRAM. While 63% fail the self test due to either a memory read failure or improper execution of an opcode. The most likely failure is a bad ROM (op code fetch) or bad opcode timing fetch. Another potential problem is opcode fetch at start up. At start up the state machine waits (counts) 5 cycles to fetch the reset vector after power on reset. The scan chain was observed to have bad op code fetches in some die via the scan chain. Timing is synchronous and the start up op code is hardwired pointing to either timing skew or weak ROM drive. All failures can not be written off as ROM errors however startup inadequate decoupling (die and probe card) may also be the issue. *OSU has fabricated a LEON3 of approximately twice the area which had only a 50% failure rate with a very similar cell library.* In addition assuming that the all failures are due to RAM and ROM the combined yield rate of the RAM and ROM project a 15% HC11 yield verse the 5% observed. When self-test and boot-load sequence failed, test evidence pointed to op code conflict or failure, addressed in section 3 below. After successful power-on reset, into self test this followed by a series of memory and I/O peripherals self-test sequences (for code, refer to Appendix 1):

- tested 1st 256 bytes on-chip SRAM
- Port B, C and D (B and C handshake with port C)

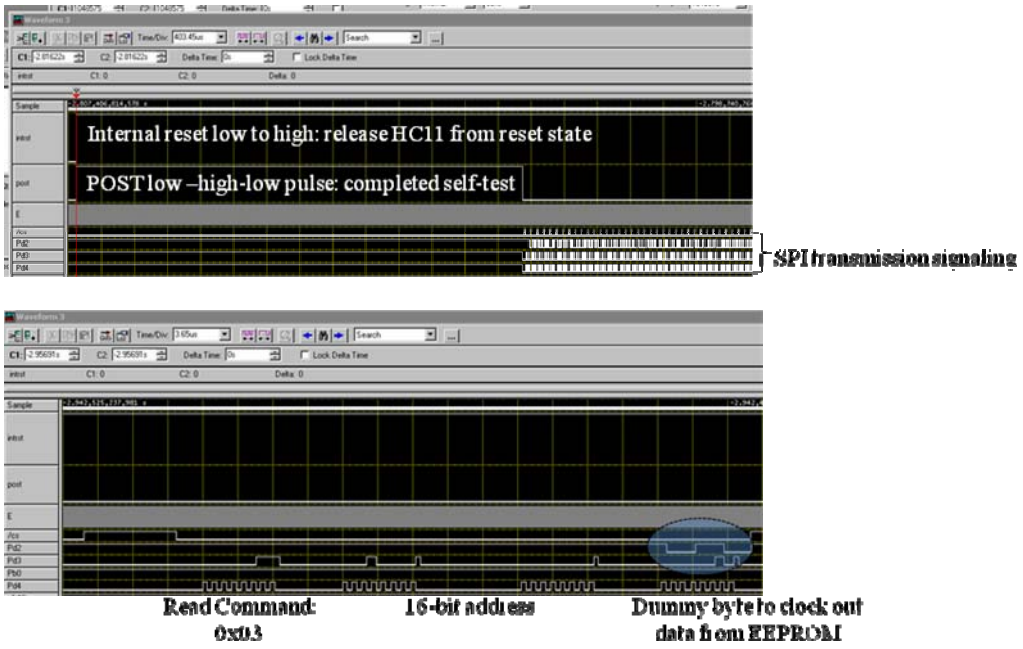


Figure 8. OSU 68HC11 communicates with external commercial EEPROM via SPI interface.

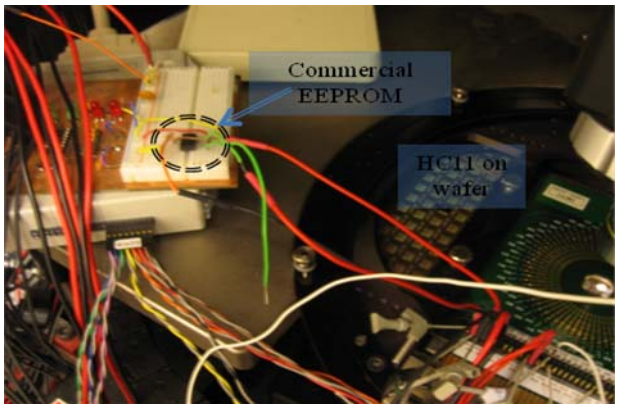


Figure 9. Test setup for communication between 68HC11 die and EEPROM.

- SPI loop back test
- SCI loop back test

The error count is recorded in register Y. 'Post' pin (see Figure 4) level high indicates self-test in process, while 'Post' pin level low indicates self-test is finished, see Appendix 4 and 11 for HC11 pad-out. After self-test is finished, the CPU boots from external memory device via SPI or SCI (depends on which module is externally available, if no SCI module is detected, the code will jump to SPI boot sequence). In this test setup, the HC11 successfully communicates and loads the user/boot code from commercial SPI-EEPROM (Atmel, AT25640A), shown in Figure 8 and 9 or the OSU SPI ROM.

3 Discussion and Challenge

There were several issues faced in the design/test of the OSU 68HC11 related to hardware or software. First is the low yield as a result of internally shorted die (33%) and self test failure (63%) (Bad ROM read or timing fetch errors.). See Section 5 High Temperature Memories for details. Improving the stability (required) of read/write of the SRAM will significantly increase HC11 yield. It is believed that self test failures are due to ROM read failure either code read error or timing. OSU has no means to discern the difference. OSU's success with the LEON3 confirms cell libraries are sound and suggests that the OSU HC11 code can be further improved to avoid potential timing issues. The HC11 CPU could not load data from the bus peripherals, external devices due to an interaction with the indexed addressing mode of the ldaa,x command and the data bus scan chain. The ldaa,x command loads data from the scan chain data bus rather from the

normal or internal data bus as it should. This scan chain enable signal interaction with the ldaa,x command results in a timing delay preventing the CPU from loading from the normal data bus with data ready. The CPU code needs to be modified and timing optimized for the synthesizer to correctly synthesize OSU HC11 CPU code. The behavior code simulates correctly further confirming a timing issue.

A final hardware problem was the failure to pad out OSU 68HC11 Port D pin to multiplex pin SCI's TX pin. Thus, there is/was a need to modify the top level Verilog file that connecting the modules pin out. On the current design, the SCI Tx pin can be and was bypassed in the user code by initializing an I/O port, e.g., Port B pin 0 to function as SCI Tx pin by writing a "bit bang" routine that performs/mimics the SCI transmission (see Appendix 1: TxPB.asm).

4 Resulted Publications

Three articles had been resulted from this funded project:

1. Chris Hutchens, Steven Morris, and Chia-Min Liu, "A proposed 68HC11 chip set for 275 degrees C," IMAPS International Conference on High Temperature Electronics (HiTEC 2006), Santa Fe, NM, May 15 - 18, 2006.
2. Chris Hutchens, Chia-Ming Liu and Hooi Miin Soo, "High temperature Down-hole Microcomputer System, Switched-Mode Power supply Component Development," GasTIPS, vol. 13, no. 1, 2007.
3. H. M Soo, Zhe Yuan, R.Sridharan, Vijay Madhuravasal, Dr. C.M. Liu, Srikanth Velore, Jiri Gaisler, Dave Hiscock, Mike Willett, and Dr. C. Hutchens, "Microcontrollers with Memory for Extreme Temperature Applications," HiTEC 2008, Santa Fe, NM, May 12 - 15, 2008.

5 High Temperature Memories

5.1 Memory Module Introduction

This section describes the work done by Oklahoma State University (OSU) to produce 275 °C operational at up 8MHz. The group demonstrated functional SPI (Serial Peripheral Interface) SRAM and ROM, and on-chip SRAM and ROM.

1. On-chip 4K X 8 SRAM

The architecture of the on chip SRAM is shown in Figure 10. The on-chip 4K SRAM is used to store the data and instructions executed by the OSU 68HC11 microcontroller and operates at the same clock frequency as the HC11 microcontroller; 2MHz, 4MHz and 8 MHz, respectively.

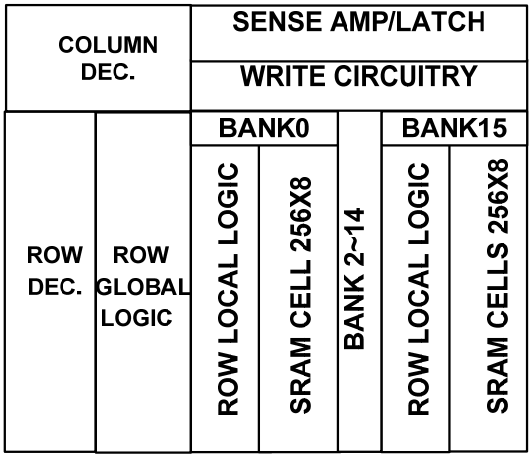


Figure 10. Block diagram of the 4Kx8 SRAM.

2. On-chip 512-byte ROM

The architecture of the on-chip 512-byte ROM is shown in Figure 11. The on-chip ROM is a masked ROM which contains the boot-up, self-test code and development/debugger software that both aids in debug and validation of the OSU 68HC11. The self-test code is followed by the bootstrap code, which boots the microprocessor from either SPI or SCI based on user choice, or boot from SPI if SCI connection fails in the debug mode test.

3. 4K X 8 SPI SRAM

The 4K X 8 SPI SRAM (Figure 12) is intended for off chip storage of data and software routines to be uploaded and executed by the 68HC11 microcontroller. The SPI SRAM clock is synchronized with the SPI port of the HC11 and operates at 2MHz, 4MHz and 8 MHz, respectively.

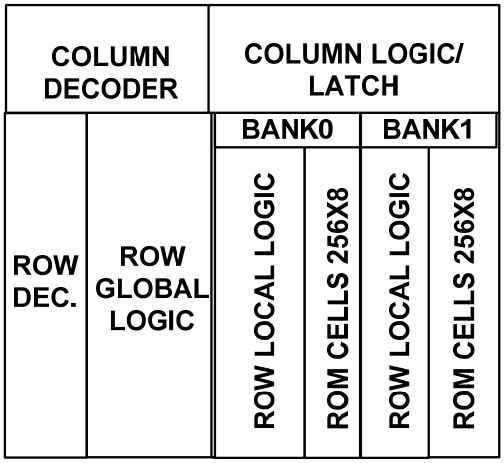


Figure 11. Block diagram of the 512-byte ROM.

The SPI SRAM has virtually the same structure as the on-chip SRAM. The exception is the communication interface and standby current circuitry. Test results have demonstrated SRAM capable of sustained operation at 8 MHz and at in excess of 275 °C.

4. 2K X 8 SPI ROM

The SPI ROM (Figure 13) is a masked ROM which has a small monitor program, 68MON. 68MON is a small monitor/debugger program for the 68HC11 for downloading and debugging of user code. Additional code was added to the 68MON to allow the 68HC11 microprocessor to load from SPI in addition to the SCI. The 68HC11 monitor program is designed to allow a person to directly type commands to the program using a terminal emulator (PC running a terminal emulation program such as Windows Hyper Terminal), by connecting a terminal to the serial communication port of the microcontroller.

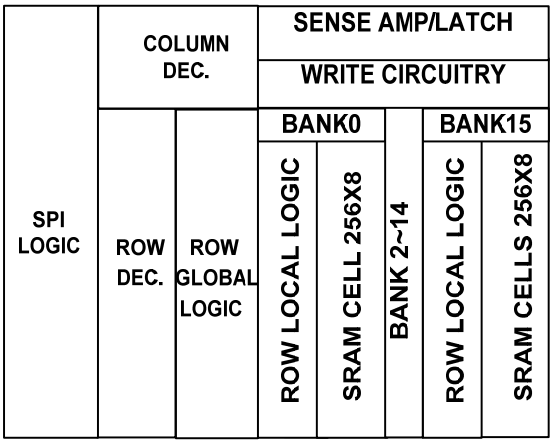


Figure 12. Block diagram of 4Kx8 SPI SRAM.

The SPI ROM again is virtually the same structure as the on-chip ROM except for a different communication interface and the size being 2K.

5.2 Memory Module Project scope

The memory deliverables for this project were the 1) design, 2) fabrication 3) test of the 4k on-chip SRAM, 512-bytes on-chip ROM, and two peripheral memory chips (4k SPI RAM and 2k mask ROM):

1. Design: the Cadence library Manager was used to generate schematic and layout. Cadence Design Environment was used for simulation. Verilog coding was used to generate the SPI logic. The combination of SKILL and Matlab coding was used to generate the ROM mask.

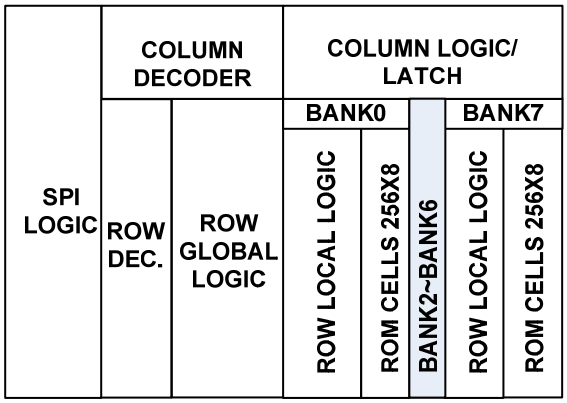


Figure 13. Block diagram of the 2Kx8 SPI ROM.

Derive 6 was used to calculate the SRAM cell size to ensure read stability and write stability.

2. Fabrication: the designs (gds layout files) were submitted to Peregrine Semiconductor to fabricate memories on the 0.5um SOS process.
3. Testing: A high temperature probe station (alessi rel 6100) was used to maintain the wafer at accurate elevated temperatures. The input patterns were given by a TLA 720 logic analyzer. Data outputs are serial in nature instead of the natural 8-bit parallel outputs and probed by the logic analyzer. The captured output data were input to Matlab program, converted to parallel data and compared by the program with the expected values.

The design requirements for the memory designs were to write/read over the temperature range from 27 °C to 275 °C and be synchronized with the communication timing provided by the microprocessor or the SPI port. Moreover lower power consumption, smaller area, lower delays and reduce design complexity were also considered. The testing methods used to define a successful read/write are outlined in section 5.5.

5.3 Memory Module Submission Timeline

1. May 2006 first run submission

In May 2006, a 256-byte SRAM was designed and fabricated with a SRAM cell area of 125 um^2. An NMOS cross-coupled sense amp was used and was padded out separately for test. The bit line precharge voltage for sense amp was set at VDD/2. Diva DRC and LVS was used on first runs with verification taking over 10 hours to LVS 256 bytes SRAM. Times for the full SRAM are now under 2 hours.

The 256-byte SRAM and the separate NMOS cross-coupled sense amp were tested functional. However, the SRAM cell was less than optimal with regard to read upset. Five die were fully tested with test patterns that did cover all error possibilities. However, sizing in the memory cell was further optimized and bank time sensitivity errors were uncovered.

2. January 2007 second run submission

The full 4kx8 on-chip SRAM and 4kx8 SPI SRAMs were designed and submitted for fabrication. The 256-byte SRAM submitted in 2006 was reorganized to improve performance in 2007 and extended to 16 banks. SRAM cell area was reduced to 115 um^2. Compared to 2006, cell area was reduced by 9.2%. Sense amp design and bias circuitry were also improved by using the test results of 2006. The sense amp was modified to reduce the impact of the kink effect and reduce the delay by separating the bit line and sense amp output. The precharge voltage remained at VDD/2 and a voltage regulator was used to achieve faster and more accurate settling compared to the 2006 design. This bit line voltage regulator/reference proved to power cost ineffectively.

SPI SRAM testing demonstrated that the design was functional but with only 15% yields. The 512-byte on-chip ROM and 2k-byte SPI ROM were again redesigned and fabricated. The SPI ROM testing demonstrated the design was functional working with 45% yields. Standby or leakage currents were found to be excessive, 6 mA due to a minor error in the I/O pads and later corrected.

3. February 2008 third run submission

The 4k on-chip SRAM and 4k SPI SRAM were redesigned and fabricated for the final time. Testing of the 4k SPI SRAM, showed a yield of 31%, an improvement of 73% over the 2007 run. The static or stand by current was 0.6 mA in 2008, reduced by a factor of 10 from 6 mA in 2007.

The 512-byte on-chip ROM and 2k-byte SPI ROM were also redesigned with improvements and fabricated. Testing of 2k SPI ROM, demonstrated a yield of 72%, an improvement of 60% over the 2007. The static current was also improved by a factor of 10 down from 2 mA in 2007 to 0.2 mA in 2008.

5.4 Memory Module Designs

1. SRAM Design

The SPI-SRAM, Figure 12, is constructed by using the SPI logic, decoders, and SRAM banks. A SRAM bank includes: RAM cells, sense amps, write circuitry and buffers, and which are arrayed in a 256x8 arrangement. The SPI-SRAM is designed for low power applications up to 275 °C with an 8 MHz clock frequency.

(a) Low power considerations

At elevated temperatures, Peregrine PMOS devices leak less than the NMOS affecting the Ion/ Ioff ratio [2]. To overcome the Ion/Ioff ratio and leakage problems and reduce power consumption, a larger than minimum channel length devices were used for both PMOS and NMOS devices in the SRAM design.

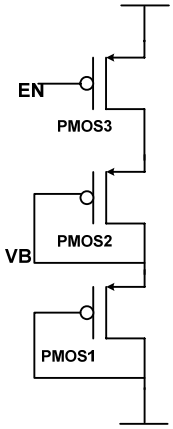


Figure 14. Two PMOS diode bias circuit.

Additional architectural features were included to enhance performance and reduce power consumption. These include [15]: Divided word line, predecoding techniques and a PMOS voltage divider for the pre-charge voltage reference, VB equals VDD/2. As shown in Figure 14, an enable signal, EN allows VB to be switched off to save power when not in use. The full dielectric properties of SOS allow this circuit to achieve divider accuracies in excess of 1 %. In addition, the switch in the reference divider of Figure 14 has the added advantage of have only a single bit line or pre-charge reference per byte or bank on at any one time conserving power. Such an approach dramatically saves on standby power.

The RC delay associated with word lines and bit lines grows proportionately with the greater number of cells along the columns and rows respectively. Word line loading by the SRAM cell's access/pass transistors along the row and is proportional to the number of columns or bit lines. The power dissipation on the word lines increases linearly with capacitance. The use of divided word line techniques reduces the associated power consumption. Power consumption is directly dependent on the required settling of the bit lines.

$$V_{final} = V_{settle} \bullet (1 - e^{-t \bullet gm / C_{COL}}). \qquad (1)$$

Equation (1), demonstrates the relationship of delay and transconductance (gm) of the PMOS1/PMOS2, in bit line settling

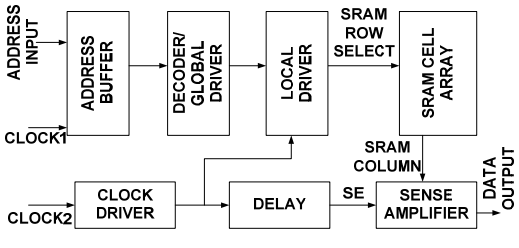


Figure 15. SRAM's read circuitry block diagram.

where V_{final} equals 90% VDD/2. and, V_{settle} equals VDD/2.

In the original design, a voltage regulator was used to bias the voltage at VDD/2 in 2007. The voltage regulator is a feedback system and has the ability to adjust the voltage to VDD/2. Compared to 2008, the two PMOS diode bias circuit reduced the pre-charge reference power by 90%, a significant improvement. In addition the SRAM cell design in the 2008 is more read robust.

(b) Robust sense amplifier design

The SRAM read circuitry is shown in Figure 15 and includes: the sense amp controlled by the sense signal (SE) and a timing delay to assure the sense amp settles with the correct data. In Figure 15, CLOCK 2 controls both SRAM's row select and sense amp sense enable signal, and DELAY ensures that SRAM conversion does not start prematurely across the temperature, process, and supply voltage corners [10]. The DELAY circuit is designed based on the analysis shown in Figure 15.

The read cycle starts with the precharging of COL and COL_BAR to VDD/2, while D and DBAR are at VDD (see Figure 16). Precharging the columns to VDD/2 decreases the sense amp delay. With COL and COL_BAR precharged to VDD/2, the sense amplifier is enabled by SE, the SRAM cell select delay of;

$$t_{cell} = C_{COL} \bullet \Delta V / I_{cell}, \qquad (2)$$

where C_{COL} is the SRAM COL line capacitance, I_{cell} is SRAM cell on current. The total read timing is approximated by equation (3):

$$t_{read} = C_{COL} \bullet \Delta V / I_{cell} + 2 \bullet C_{gdop} \bullet V_{thp} / I_{SA} + \frac{Cgsp}{gm_p + gm_n - go} \ln \frac{V_{final}}{\Delta V} \qquad (3)$$

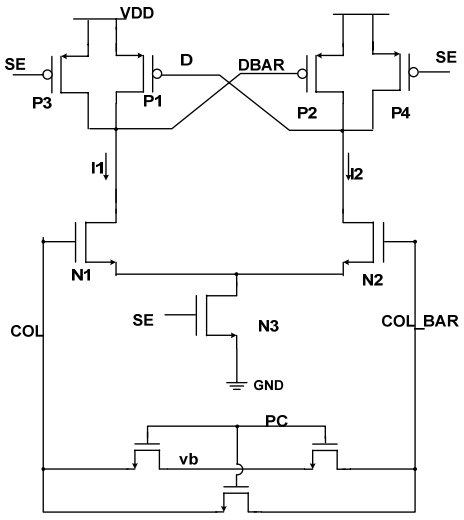


Figure 16. SRAM's read circuitry block diagram.

where C_{gsp} and C_{gdop} are the gate to source capacitance over overlap capacitance of P1 and P2. The transconductance, $gm_{x,}$ of the NMOS differential pair and PMOS cross coupled pair, I_{SA} is the tail current of sense amp, go is the output conductance at node D or DBAR, and V_{final} equal VDD-V_{thp}.

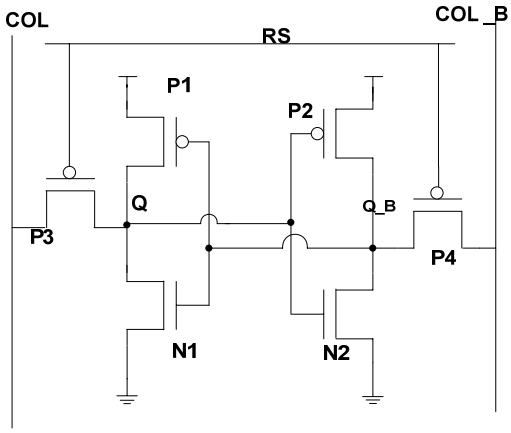


Figure 17. 6T SRAM cell schematic.

(c) SRAM cell

The 6T SRAM cell, Figure 17, uses PMOS access devices to reduce both area and leakage currents. Cell and pull-up ratios are calculated to assure proper read and write stability. Cell ratio is defined as the size ratio between pull down transistor (N1,N2) and pass transistor (P3, P4) and the pull-up ratio of the cell is defined as the size ratio between pull up transistor (P1, P2) and pass transistor (P3, P4) [14].

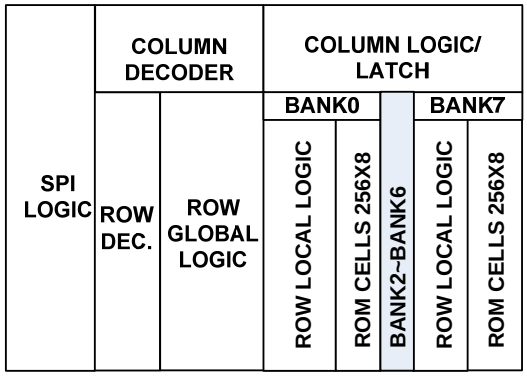


Figure 18. Block diagram of the 2k SPI ROM.

2. ROM Design

The SPI-ROM, Figure 18, has the similar structure to the 4k SPI-SRAM except the ROM cell and sense amp. A sense amp as such is not used in ROM due to the strong drive capability of ROM cell and lower column line capacitance.

As shown in Figure 19, the ROM cells are connected to either VDD or VSS depending on the value of the bit stored and are read from column lines. After the ROM layout without the 'data' connection to VDD/VSS, the resulting layout was completed with metal lines placed on the original layout by using SKILL code written to instantiate the desired logic bit. The PMOS ROM cell design is low leakage with reduced area but having a weak pull down behavior relative to its NMOS equivalent.

5.5 Testing results

5.5.1 SPI SRAM Testing

(a) SRAM testing conditions

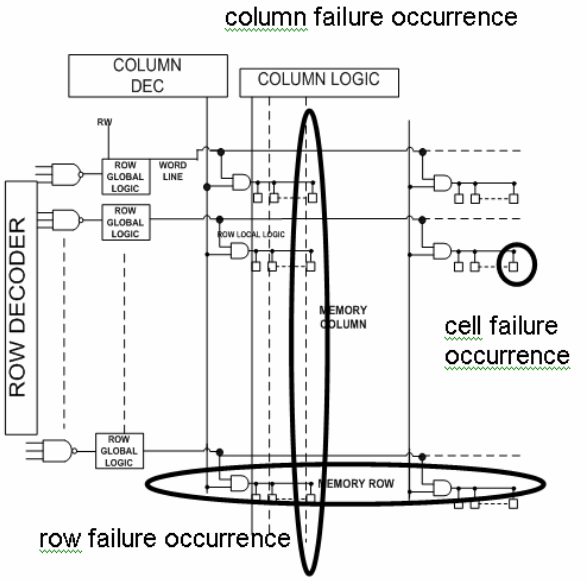


Figure 20. SRAM failure modes.

The definition of successful write/read for SRAM is a write followed by 2 successful reads of FF's, 00's, 55's, followed by 255-thur-0's of each byte in the die. The 4K SPI-SRAM must pass the frequency, temperature, and voltage corners of 2MHz, 4MHz and 8MHz, 27°C, 200°C, 275°C, and 295°C, and 2.5V, 3V, 3.3V, and 3.6V respectively on the alessi rel 6100 probe station. A 16 hour test at 300 °C along with a 1 week test of packaged SPI SRAM on Roger board at 200 °C were completed to assure SRAM long time viability. A 300°C probe station (alessi rel 6100) was used to control temperatures for wafer testing. Test patterns included FF's, 00's, 55's, 255-thur-0's written to every byte followed by two successful reads. The output was checked by automatic test bench code in Matlab. The Tektronix TLA 720 logic analyzer was used to provide the input pattern and observe the outputs, where the logic analyzer's switching threshold voltage was set to VDD/2.

(b) Testing analysis

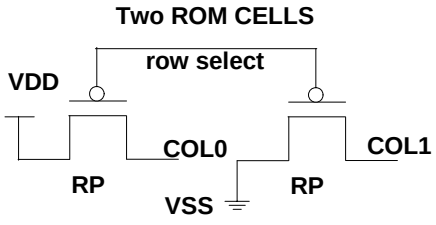


Figure 19. ROM schematic.

Figure 20 defines the 3 types of errors occurred in the SPI 4k SRAM: cell failure, row failure and column failure. Cell failures occurred in 1 to 4 cells in a single and across all banks and no more than 4 bits in a single byte. Row failures involved all rows of all 16 banks incorrectly reading but rarely more than one row. Column failures were usually 1 or more columns but never more than two. These results exclude those die deemed as being shorted.

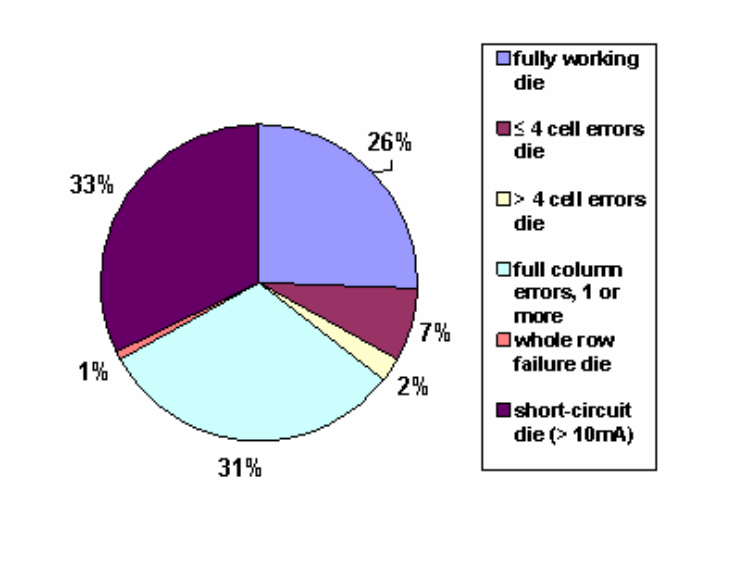


Figure 21. Percentage Distributions of Tested 121 4k SRAM Die

Figure 21 and Figure 22 show the comparison of the SPI SRAM testing results in 2008 and 2007 respectively as follows:

- 26% of the die read successfully in 2008 compared to 15% in 2007, a yield improvement of 73%.
- ≤4 cell errors were 11% and 7% for 2007 and 2008

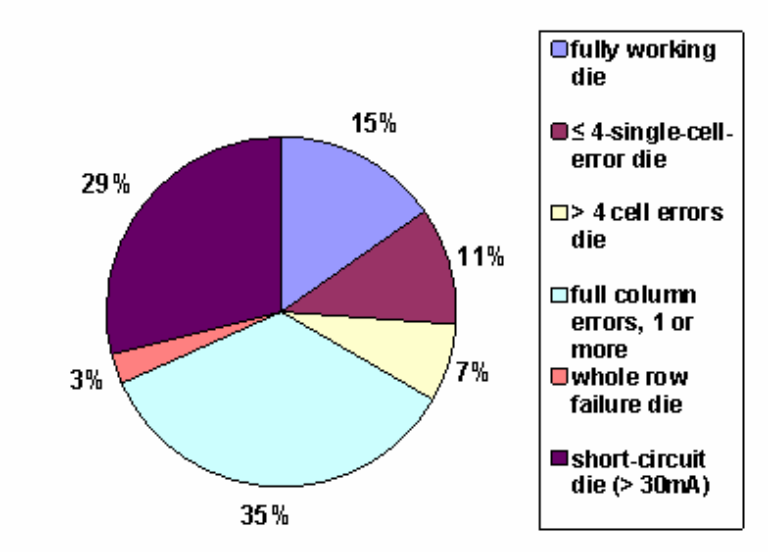


Figure 22. Percentage Distributions of Tested 111 4k SRAM Die (2007).

respectively.

- >4 cell errors were 7% and 2% for 2007 and 2008 respectively.
- 35% of the die has full column errors in 2007 compared to 31% in 2008.
- 3% of the die has row errors in 2007 verses 1% in 2008.
- 29% has short circuits in 2007 verses 33% in 2008.

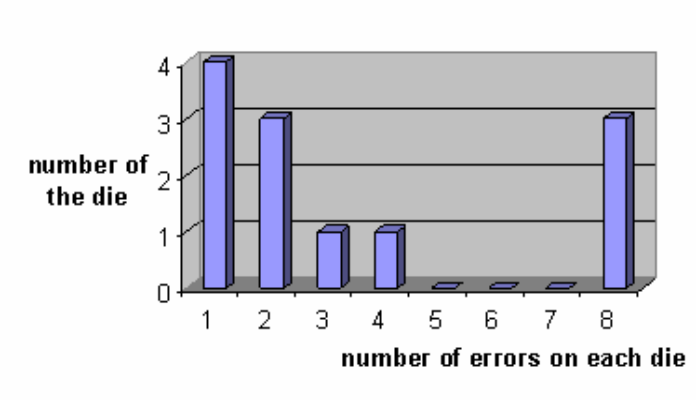


Figure 23. SRAM cell error distributions (2008).

Refer to Figures 21 and 22.

(c) Error diagnosis of 2007 to 2008

(1) Cell errors

7% of the die have less than 4 cell errors. Compared to cell errors in 2007, the cell errors were reduced. This is believed to be the benefit of adding more contacts and improving step edge coverage within the cell. The possible causes of a stuck 1 or 0 can be open via or contact, poorly formed transistor, i.e. open or short NMOS or PMOS.

Figure 23 shows the SRAM cell error distributions. Of the die with 1 or more cell errors per byte, 4 die have only 1 cell error out of 4k byte; 3 die have only 2 cell errors; 1 die has 3 cell errors; 1 die has 4 errors. The less-than-5 error cells are usually a stuck bit (1 or 0). No bytes have 5, 6 or 7 cell errors out of 50 error die tested in 2008 with cell errors. This trend is close to the expected probability distribution as random cell error is approximately 3.3% out of 121 SRAM tested as having a single cell error. The probability of a 2 cell error die is 1%. Test data show that 2.5% of the memory die have 2 cell errors. The 8 error cells are usually and were determined to be the result of read out flipping patterns and were observe on multiple occasions when the test setup was poorly setup. Total byte

flips were observed as a result of a bad test setup of the probes but was not noted until 2008.

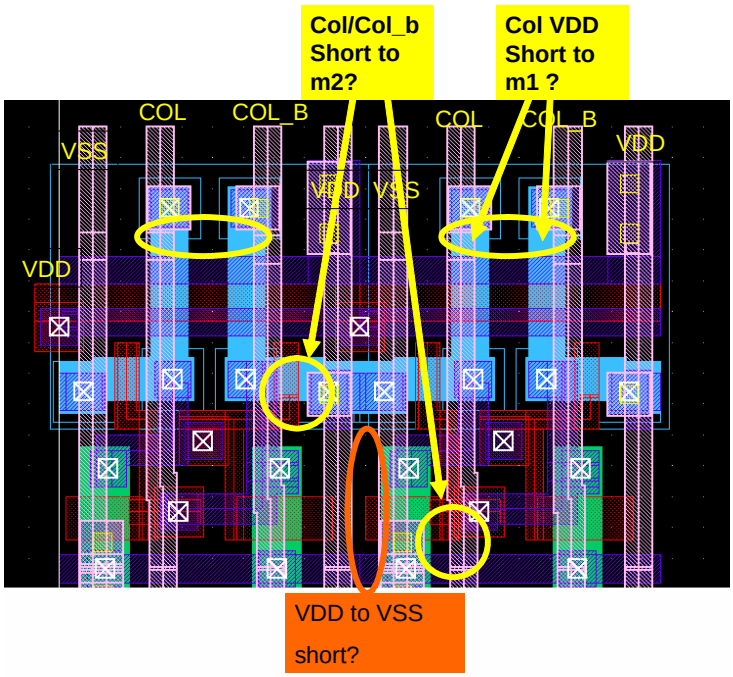


Figure 24. 6T-SRAM cell layout potential shorting failures locations.

(2) Column errors

For the column errors, a power line short or ill formed sense amp is a reasonable cause. In the SRAM cell (Figure 24) VDD and or VSS are minimum separation from COL_BAR and COL respectively at the vias. Two possible causes of column failures are a possibility 1) Col/Col_b to m2 VDD or VSS or 2) Col/Col_b to m1 VDD. An ill formed sense amp and D latch can be another cause of the column failures. Revaluating the sense amp and D latch design and layout may reduce the column failure.

(3) Row errors

One percent of the 2007 SRAM die have row errors. Compared to other errors in 2007, the row errors were very small and were further reduced by 3X in 2008. This is believed to be the benefit of adding more contacts in the decoding logic.

(4) Shorted die

33% of the SRAM die is Shorted die. This is assumed to be due to the process design rule limitations and there was no change over 2007. *Process design rules as set by the manufacturer were always adhered to.* It should be noted that the only *long*s in m2-m2 runs with minimal spacing in both memories are the vertical VDD and VSS runs, Figure 24, in the memory array and column arrays. These were not present in the LEON3 run but were present in the all OSU HC11 runs.

This may account for the large differences in the two die yields when die areas are considered.

(5) Static current

Static current is 0.6 mA at 27 °C and improved from 6 mA. This is a benefit of fixing of I/O pads errors, improving the bias generator and fixing internal floating nodes.

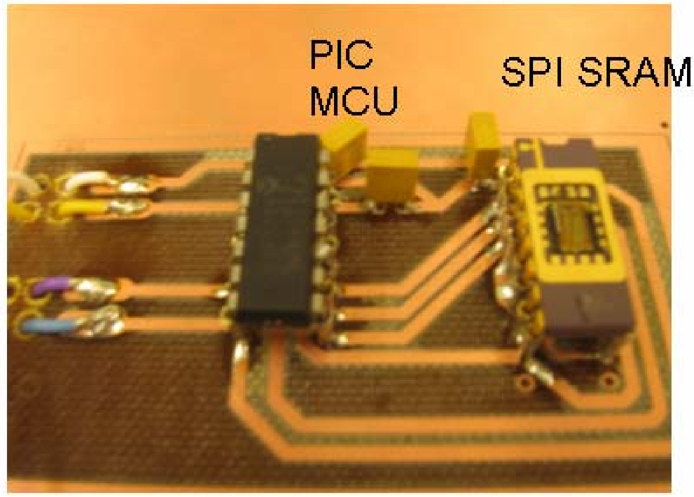


Figure 26. PIC & SRAM chips on polyamide PCB.

5.5.2 SPI-SRAM Testing on PCB

Figure 25 shows the test set up of the PIC and SRAM chips on a PCB. As shown in Figure 26, the testing is writing a byte and reading back a byte to/from SPI SRAM. The output byte from SPI SRAM is read by PIC using SPI standard and echo to HyperTerminal window. Testing was carried out at 200 °C due to PIC and PCB temperature limitations. This is was more of a demonstration and no errors were observed. The four die used were previously screened as know good die to 275 °C.

5.5.3 SPI ROM Testing

(a) ROM testing conditions

The 2K SPI-ROM passed the frequency, temperature, and voltage corners of 2MHz, 4MHz and 8MHz, 27°C, 200°C, 275°C, and 295°C, and

Figure 25. HyperTerminal output.

2.5V, 3V, 3.3V, and 3.6V respectively test on the Alessi Rel 6100 probe station. The test consisted of confirming OSU's modified version of 68MON monitor code by using the Tektronix TLA 720 logic analyzer to capture the output data, and then compared it with the original memory file used to form the ROM contents by running an automatic Matlab testbench. The setup of the Tektronix TLA 720 instrument is same as SPI-SRAM testing.

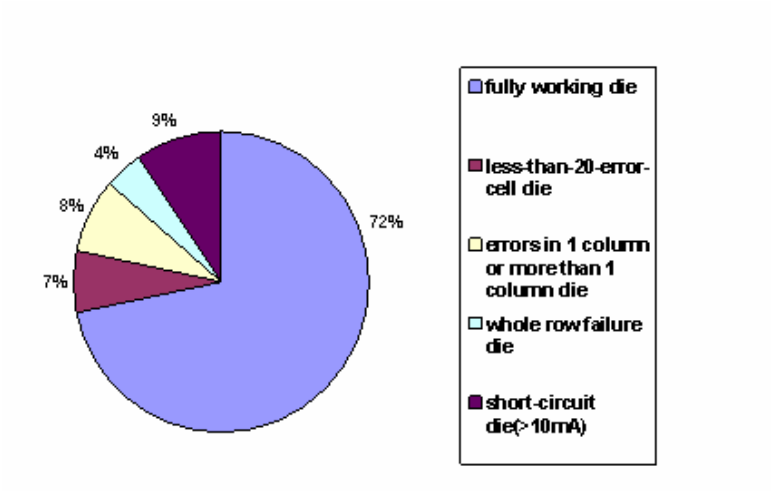


Figure 27. Percentage distributions of tested 74 SPI 2K-ROM die by failure mode (2008)

(b) Testing analysis

Figure 27 and Figure 28 show the comparison of the SPI ROM testing results in 2008 with the results in 2007 and are summarized as follows:

1. 72% of the die read out successfully in 2008 and compared to 45% in 2007, the yield improved by 60%.
2. ≤20 cell errors were 7% in 2008 and 16% in 2007. Totally 5 die have the cell errors, 1 die has 1 cell error, 1 die has 2 cell errors, 3 die has more than 8 cell errors and that are attributed to a testing setup.. ROM cell errors are consistent with SRAM cell errors.
3. 8% of the die has full column errors in 2008 versus 19% in 2007.
4. 4% of the die has row errors in 2008 versus 6% in 2007.
5. 9% are short die in 2008 versus 14% in 2007.

(c) Error diagnosis of 2007 to 2008

(1) Cell errors

7% of the SPI 2k-ROM have the multi-cell errors and was reduced from 2007. The 2008 ROM cell has greater drive and 1 more contact on the PMOS gate.

(2) Column failure

8% of the SPI 2k-ROM have column failures and was reduced from 2007. Column logic was redesigned and more contacts added in the layout.

(3) Row failure

4% of the SPI 2k-ROM have row failures and was also reduced from 2007. This may be the benefit of adding more contacts for the decoding logic or a process shift.

(4) Shorted die

9% of the ROM die are shorted. This is due to an unknown process limitation. All die submission passed DRC and LVS per process PDK requirements.

(5) Static current

Static current was 0.2 mA at 27 °C in 2008, down by two orders of magnitude 2 mA at 27 °C in 2007. This is benefit of fixing an error in the I/O pads and floating nodes in ROM dynamic read logic.

6 Conclusion

Summary of tasks accomplished

The deliverables for this project were the design, layout, fabrication, test, and integration of three integrated circuits suitable for microcomputer based controllers for 275 °C operation. This included 50 die of each of the following parts.

1. **68HC11**

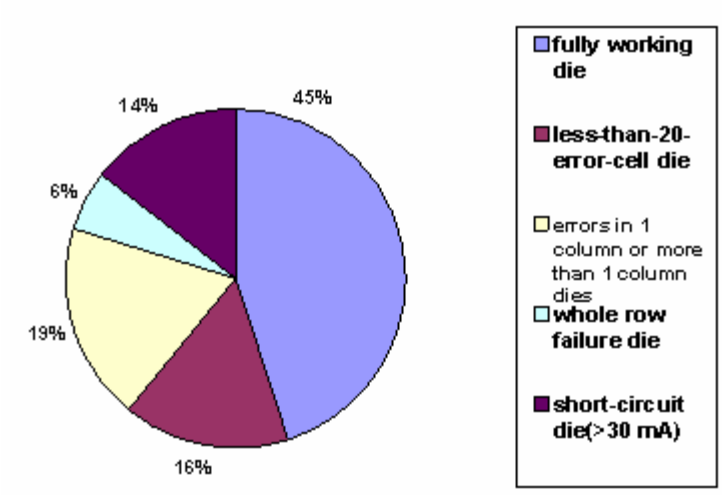


Figure 28. Percentage Distributions of Tested 69 SPI 2K-ROM Die (2007).

single chip microcomputer integrated circuit. A 68HC11 microcomputer with boot ROM, 4K by 8 bit static RAM, counter/timer unit, parallel input/output (PIO) unit, and serial peripheral interface (SPI) unit. Yields were inadequate to allow the delivery of 50 die.

2. **Data RAM integrated circuit**. A 4K by 8 bits static RAM with SPI communications circuit. Sufficient die are available to deliver both SRAM and ROM die.
3. **Mask ROM integrated circuit**. A 2K by 8 bits masked ROM with SPI communications circuit. Sufficient die are available to deliver both SRAM and ROM die.
4. **Documentation**. All design documentation and data was organized and provided to the sponsor upon completion of work. This final report discusses and documents the final design, test results and problems encountered, and suggested further work/potential fixes.

This report has demonstrated the successful development of a 275 °C 4K-SRAM and 275 °C 2K-ROM silicon designs with an SPI interface. The designs are suitable for, aerospace, well logging, solar controllers, automobile, and other extreme temperature environment applications. The memory devices demonstrated proper performance across the frequency, temperature, and voltage corners of 2MHz, 4MHz and 8MHz, 27°C, 200°C, 275°C, and 295°C, and 2.5V, 3V, 3.3V, and 3.6V respectively, making the memory devices suitable for the HC11 chip and other processors, for use as additional memory storage and/or external system boot memory devices. The SPI SRAM and SPI ROM ‘s 2008 yields are low, but yield and standby power consumption have been significantly improved from the 2007 designs. Further yields can be expected with continued efforts.

In conclusion, the project was scheduled to be executed in six subtasks, and each subtask was completed except for the integration of the HC11 chip into a functioning microcontroller. The HC11 was not integrated due to timing limitations the selection of to thin of a AlN substrate. The resulting brittleness of the substrate resulted in the combined mass of the wires and die breaking the “holes” used in wire mounting breaking the board.

At this point there was inadequate time to develop a polyamide board (240 °C) The corresponding documentation where relevant is provided in the Appendices:

1. Research management plan.
2. Technology status assessment.
3. Formation of industry advisory committee.
4. Megacell design, validation, fabrication, and test.
5. Integration and fabrication of 68HC11.
    - 68HC11 Core and Peripherals Implementation (Appendix 2 -8)
    - Verify 68HC11 and I/O "opencore" code
    - Memory Megacells Integration (Appendix 7 – 10)
    - Communication Megacells - /UART/SPI/Comparators Integration (Appendix 6)
    - Counter/timer/QPT Megacells Integration (Appendix 3)
    - 68HC11 Core and Peripherals, pre-submission review-test, layout, timing verification ALL FUNCTIONs (Testing and Simulation section of each appendixes)
    - Formalize/Simulate 68HC11 test plan (Testing and Simulation section of Appendix 2)
    - Final Timing and Functionality Verification on Wafer (Section 2 of this report)
6. Documentation.

Subtasks on research management plan, technology status assessment, and formation of industry advisory committee. Subtasks 1 to 3 were 100% completed, and were reported previously in the intermediate report.

Megacell design, validation, fabrication and test. Subtask 4 involved design, validation, fabrication and test of megacells. These megacells are circuits corresponding to functional blocks in the final 68HC11. Design, fabrication, and testing of megacells was completed. The improved and final version of the designs were delivered on

February 29th 2008 for wafer fabrication, and test results were reported in this final report plus appendixes.

Integration and fabrication of the 68HC11. Subtask 5 involved integrating the macrocells, designed and tested in Subtask 4, into a complete 68HC11 for fabrication. This integration consisted of 11 steps and all are in 100% completion:

- **68HC11 core and peripherals timing verification, Subtask 5.1., 100% complete**. Verilog code for modules that link the peripherals to the ALU and interrupt handler circuitry were designed and written. This Subtask consisted of verifying the timing and logic consistency between the ALU, interrupt handler circuitry, and the peripherals.
- **Verify 68HC11 and I/O code, Subtask 5.2. 100% complete.** OSU substituted their own code for the 68HC11 ALU and interrupt handler circuitry, and did not using opencore code but compiled their own primarily due inadequate documentation of opencore codes. The generated code was verified by comparing the results, in every detail, with an HC11 emulator to ensure the code validity.
- **68HC11 core and peripheral megacell integration, Subtask 5.3. 100% complete.** This Subtask was closely related to the timing verification, Subtask 5.1. There was a modest degree of difficulty simulating this when this code was integrated with the code from the interrupt handling, power on rest (POR) functions, and counter timer module. This interface was verified at the behavioral level as well as at post place and route with interconnect parasitic.
- **Memory megacells integration, Subtask 5.4. 100% complete**. This integration involved verification of hardware interfaces built on top of the ROM and RAM memory cells. This included both SPI ROM and RAM and on chip ROM and RAM in the HC11.
- **Communications megacells - /UART/SPI, Subtask 5.5. 100% complete.** This Subtask involved 3 SPI ports and a single UART. The SPI ports on the SPI memories support slave function only. The HC11 SPI port supports both master and slave functions in addition to supporting 3.3 and 5 V logic levels. There was a modest degree of risk with 5V logic level as a result of the Peregrine process

being a 4V process. Special efforts were taken, at the circuit level, to mitigate these risks. However, the 5.5 V corners are unachievable resulting in potential life time issues primarily associated with low temperature operation arising from the 'kink" and avalanche effects and excessive leakage at high temperatures for the fast process models.

- **Counter/timer megacell integration, Subtask 5.6., 100% complete.** This Subtask involved completing full design of the counter/timer, pulse accumulator, COP (computer operating properly) timer, and parallel Ports.
- **Pre-submission Review - Test, layout, timing ALL FUNCTIONs, Subtask 5.7., 100% complete** This Subtask involved confirming the timing of all sub-blocks of the OSU 68HC11, review of the final floor plans and test plans, and timing optimizing to compensate interconnect parasitic including; SPI, SCI, Port blocks etc.
- **Formalize/Simulate 68HC11 test plan, Subtask 5.8., 100% complete.** This Subtask involved the writing and documenting of the test vectors to be used in testing the OSU 68HC11.
- **Revise layout as required, Subtask 5.9, 100% complete.** The project team modified and checked the layouts of the full 68HC11 circuit and the 2 peripheral memories in response to the pre-submission reviews uncovered. The key Subtask here included confirmation of an adequate number of power pins, adequate on-chip decoupling and signal integrity issues in general. The Subtasks summarized here are interrelated, requiring some iteration to find a reasonable optimization of design parameters.
- **Final Timing Verification, Subtask 5.10, 100% complete.** This Subtask involved the verification of the OSU 68HC11 timing at post integration with parasitic. The test vectors developed in Subtask 5.8 were verified in this Subtask.
- **Fabrication Submission, Subtask 5.11**, **100% complete.** The OSU 68HC11 and peripheral components (Figure 6) were submitted to the foundry. The HC11 was 6.5 x 6.5 mm^2 or 42.25 mm^2 in area and comprised of 50449 gates (cells = 19852). The SPI-ROM was 2.4 mm x 2.13 mm in area. The SPI-SRAM was 2.4mm x 4.791mm in area. The SPI controllers had a gates count of

approximately 22950 gates (cells = 419). Completion of Subtask 5.11 is major milestone 3.

Documentation. **100% complete.** This Subtask provided the sponsor all design documentation and data upon completion of work. This document fulfills the final documentation task.

References

[1] V. Jeyaraman, "Design, characterization, and automation of a high temperature (200 °C) standard cell library," in *Electrical and Computer Engineering*. Stillwater, Oklahoma: Oklahoma State University, 2004.

[2] U. Badam, S. Viswantathan, V. Jeyaraman, C. Hutchens, C. Liu, and R. Schultz, "High temperature SOS cell library,"presented at International Conference on High Temperature Electronics (HITEC), Santa Fe, New Mexico, 2006.

[3] W.Agaststein, K. McFaul, P.Themins, "Validating an ASIC Standard Cell Library", Intel Corporation, 1990.

[4] Chris Hutchens, Steven Moris, and Chia-min Liu, "A proposed 68HC11 chip set for 275 degrees C," IMAPS International Conference on High Temperature Electronics (HiTEC 2006), Santa Fe, NM, May 15 - 18, 2006.

[5] Chris Hutchens, Chia-Ming Liu and Hooi Miin Soo, "High temperature Down-hole Microcomputer System, Switched-Mode Power supply Component Development," *GasTIPS*, vol. 13, no. 1, 2007.

[6] J. Tao, N. Cheung, and C. Ho, "An Electromigration Failure Model for Interconnects Under Pulsed and Bidirectional Current Stressing," *IEEE Transactions on Electron Devices*, vol. 41, pp. 539, 1994.

[7] J. Tao, N. Cheung, and C. Ho, "Metal Electromigration Damage Healing Under Bidirectional Current Stress," *IEEE Electron Device Letters*, vol. 14, pp. 554, 1993.

[8] *Peregrine Semiconductor, Foundry Services. [cited Mar. 8, 2006]; Available from: http://www.peregrine-semi.com/content/foundry/foundry.html.*

[9] *68HC11 Reference Manual, Document 68HC11RM/D, Rev 6 4/200,. Freescale Semiconductor Inc, 6501 William Cannon Drive West, Austin, Texas, U.S.A., 2002.*

[10] Nambu, H.; Kanetani, K.; Yamasaki, K.; Higeta, K.; Usami, M.; Fujimura, Y.; Ando, K.; Kusunoki, T.; Yamaguchi, K.; Homma, N., "A 1.8-ns access, 550-MHz, 4.5-Mb CMOS SRAM," *IEEE Journal of Solid-State Circuits*, vol.33, no.11, pp.1650-1658, Nov 1998.

[11] Kobayashi, T.; Nogami, K.; Shirotori, T.; Fujimoto, Y., "A current-controlled latch sense amplifier and a static power-saving input buffer for low-power architecture," *IEEE Journal of Solid-State Circuits*, vol.28, no.4, pp.523-527, Apr 1993.

[12] Seki, T.; Itoh, E.; Furukawa, C.; Maeno, I.; Ozawa, T.; Sano, H.; Suzuki, N., "A 6-ns 1-Mb CMOS SRAM with latched sense amplifier," *IEEE Journal of Solid-State Circuits*, vol.28, no.4, pp.478-483, Apr 1993.

[13] Takeda, K.; Hagihara, Y.; Aimoto, Y.; Nomura, M.; Nakazawa, Y.; Ishii, T.; Kobatake, H., "A read-static-noise-margin-free SRAM cell for low-VDD and high-speed applications," *IEEE Journal of Solid-State Circuits*, vol.41, no.1, pp. 113-121, Jan. 2006.

[14] Jan M.Rabaey, ”Digital Integrated circuit- A design perspective”,p.658-p.661.

[15] Amrutur, B.S.; Horowitz, M.A., "Fast low-power decoders for RAMs," *IEEE Journal of Solid-State Circuits*, vol.36, no.10, pp.1506-1515, Oct 2001.

[16] Lovett, S.J.; Gibbs, G.A.; Pancholy, A., "Yield and matching implications for static RAM memory array sense-amplifier design," *IEEE Journal of Solid-State Circuits*, vol.35, no.8, pp.1200-1204, Aug 2000.

List of Appendix

APPENDIX 1

BOOTLOADER AND MONITOR CODES DOCUMENTS

TABLE OF CONTENTS

DOCUMENT **PAGE**

1 DMS 68HC11 Self-Test and Boot-load

The internal 512 bytes ROM consists of boot-up self-test code that both help debug the chip and validate the chip before executing the first user code. It consists of the following processes:

- Internal SRAM write/read AA&55.
- All Ports – Loop Port B to C and C to D.
- Loop-Back test a byte over SPI (Internal).
- Loop-Back test a byte over SCI (Internal).
- SELFTEST "pass" – BIT1 of $1001 is set.
- Load code over SCI (Check Null)/SPI.
- Set BOOTSET ($1001[0]) to indicate boot done.

The self-test code followed by the bootstrap code, which allows the microprocessor to boot from SPI or SCI based on user choice, or boot from SPI if SCI connection fails.

SCI Bootstrap: Initial SCI registers setup, refresh SRAM, and download PC program into SRAM.

Pseudo code:

1. Initiate related registers (STACK, SPSR, BAUD)
2. Refresh SRAM.
3. Send BREAK to PC.
4. Wait until START bit detected.
5. Receive data.
6. Download 256 bytes program from PC to SRAM.

SPI Bootstrap: Initial SPI registers setup, refresh SRAM, and download a small monitor from external SPI_ROM into HC11 internal RAM.

Pseudo code:

1. Initiate related registers (DDRD, SPCR)
2. Refresh RAM.
3. Send Address to external SPI-ROM (write to SPDR).

4. Wait until SPIF bit set (check SPIF bit in SPSR).
5. Receive data (load from SPDR).
6. Repeat step 3 to 5 until download 256 bytes program from SPI-ROM to RAM.

A small monitor program, 68MON, resides in the external masked ROM. 68MON is a small monitor program for 68HC11 for downloading and debugging written by Keith Vasilakes. Additional code was added to the 68MON to allows the OSU 68HC11 microprocessor to load from SPI in addition to the SCI. The 68HC11 monitor program is designed to allow a person to directly type commands to the program using a terminal emulator (PC running a terminal emulation program such as Windows Hyper Terminal), by connecting a terminal to the serial communication port of the microcontroller. 68MON monitor supports some standard monitor functions and an Intel upload (when defined) for those cases when an Intel hex is used (an s19 file is converted to Intel hex). The 68mon standard monitor functions include string IO character conversion, and serial port support. These functions can be called from a user assembly language program. This modified 68MON version did not support writing to the EEPROM, and changing the baud rate.

2 Self-test and Bootstrap Codes

```
* Initialize
  org $bf00

PORTD   equ     $08
DDRD    equ     $09
SPCR    equ     $28
SPSR    equ     $29
BAUD    equ     $2B
SCCR1   equ     $2C
SCCR2   equ     $2D
SCSR    equ     $2E
SCDR    equ     $2F
SPDR    equ     $2A
PIOC    equ     $02
PORTB   equ     $04
PORTCL  equ     $05

START       clr $0101
          ldy #$00
* Test SRAM
        ldx #$003f
            ldab #$AA
Testaa:             stab $0,X
                    ldaa $0,X
                    sba
                    beq loop1
                    inc $0101
                    iny
loop1:              dex
                    bne Testaa

                    ldx #$003f
            ldab #$55
Test55:             stab $0,X
                    ldaa $0,X
                    sba
                    beq loop2
                    inc $0101
                    iny
loop2:              dex
                    bne Test55

* End of memory test

* REGISTER TEST  If ROM > 512 bytes
*       ldab #$FF
*       ldx  #$1000
********************************** PORTB
*       stab        $4,x
*       cmpb $4,x
*       beq         no_err6
*       inc         $0101
```

```
*       iny
*       brano_err6
* no_err6
*       ldaa #$00
*       staa $4,x
*       clr $4,x
***********************************
DDRC
*       stab            $7,x
*       cmpb    $7,x
*       beq             no_err7
*       inc             $0101
*       iny
*       brano_err7
*no_err7
*       ldaa #$00
*       staa $7,x
*       clr     $7,x
***********************************
DDRD
*       stab            $9,x
*       ldaa    #$3F
*       cmpa    $9,x
*      beq no_err8
*       inc             $0101
*       iny
*       brano_err8
*no_err8
*       clr     $9,x
***********************************
BAUD
*       ldx     #$1000
*       ldaa    #$03
*       staa            $2B,x
*       cmpb    #$03
*       beq             no_err13
*       inc             $0101
*       iny
* no_err13

  ldx  #$1000
* port selftest
    ldaa  #$03
    staa  PIOC,x
    ldaa  #$55
    staa  PORTB,x
    ldab     PORTB,x
    cmpb     #$55
    beq  no_err14
    inc   $0101
    iny
no_err14
* full input mode
    ldaa  #$17
    staa  PIOC,x
* get portb data $55
* stra pulse 1

*Disable SPI and SCI
    ldaa  #$00
    staa  SPCR,x
    staa  SCCR2,x

    ldaa  #$ff
    staa  DDRD,x
    ldab  #$00
    stab  PORTD,x
    ldab  #$20
    stab  PORTD,x
    ldaa  PORTCL,x
    cmpa  #$55
    beq  no_err15
    inc   $0101
    iny
no_err15
*full output mode normal
*stra==0
    ldab  #$00
    stab  PORTD,x
    ldaa  #$1f
    staa  PIOC,x
    ldaa  #$55
    staa  PORTCL,x
* portd receive add
    ldab #$f0
    stab     DDRD,x
    ldaa     PORTD,x
    anda     #$05
    cmpa     #$05
    beq  no_err16
    inc   $0101
    iny
no_err16

********************************** Hooi
Miin testing
                ldx    #$1000
                ldaa      #$38
                staa    DDRD,x
                ldaa    #$d0
                staa    SPCR,x

                ldab   #$0f
                ldaa      #$55
        staa    SPDR,x

LOOPwt                  decb
                beq       NEXT
                ldaa      SPSR,x
                bpl    LOOPwt
```

```
NEXT   ldaa      SPDR,x
       cmpa  #$55
    inc   $0101
    iny
no_err18
       ldaa      #$80
       staa      SPSR,x

********************************** Lisa
testing

     ldaa          #$03
     staaBAUD,x
     ldaa          #$00
     staaSPCR,x

*The following  code transmits a character and
waits for it to finish tranmission:
     staaSCCR1,x
     staaSCCR2,x
     ldaa          #$0c
     staaSCCR2,x

         ldx    #$1000
         ldx    SCSR,x
         ldx    #$1000
     ldaa          #$0f
     staa   SCDR,x


     ldab   #$ff
wloop2
     ldaa   SCSR,x
     cmpa   #$80
     beq    rbegin
         decb
         cmpb   #$00
         bne    wloop2
* The following code receives a character from
the serial port:

rbegin
* RCLoop
     ldab   #$ff
rloop2
    ldaa   SCSR,x
     cmpa   #$D0
     beq    cerror
         decb
         cmpb   #$00
         bne    rloop2

cerror
* Disable SELFTEST
     ldaa   #$02
     staa   $1001
     beq  no_err18



      ldx   #$1000
      ldaa   SCDR,x
      cmpa  #$0f
    beq   no_err19
     inc   $0101
     iny
* If SCI fails, go to SPIboot
         jmp    spiboot
no_err19


* BOOTLOADER FIRMWARE FOR 68HC11
* equATES FOR USE WITH INdex OFFSET =
$1000
* PORTD   equ    $08
* DDRD    equ    $09
* SPCR    equ    $28    *(FOR DWOM BIT)
* BAUD    equ    $2B
* SCCR1   equ    $2C
* SCCR2   equ    $2D
* SCSR    equ    $2E
* SCDAT   equ    $2F
* PPROG   equ    $3B
* TEST1   equ    $3E
* CONFIG  equ    $3F
* THIS BOOTSTRAP PROGRAM ALLOWS
THE USER TO
* DOWNLOAD A PROGRAM OF EXACTLY
256 BYTES.
* EACH BYTE OF THE PROGRAM IS
RECEIVED BY THE
* SCI, STARTING WITH THE $0000 BYTE
AND WORKING
* UP TO THE $00FF BYTE.
*      ORG    $C000
* INIT STACK
      lds    #$01FF
      jmp      JUP

    org $C000
JUP

         ldaa   #$FF
      staa   SCDR,X



* WRITE $FF TO ENTIRE RAM (EXCEPT
LAST TWO BYTES
* WHICH ARE USED BY THE STACK)
      PSHX
      ldx    #$FF02
```

```
LOP1
      staa   $FE,X
      inx
      bne    LOP1
      PULX

* SEND BREAK TO SIGNAL START OF
DOWNLOAD
          ldaa   #$0D
          staa   SCCR2,x
* CLEAR BREAK AS SOON AS START BIT
IS DETECTED
          pshy
          ldy    #$03
pdloop    ldaa   #$ff
pdloop2 ldab   PORTD,x
          andb   #$01
      cmpb   #$00
      beq    srnext
          deca
          cmpa   #$00
      bne    pdloop2
      dey
      cpy    #$00
      bne    pdloop

      jmp    spiboot

srnext
      puly
      ldaa   #$0C
             staa   SCCR2,x
* WAIT FOR FIRST CHARACTER (USERS
SEND $FF)

          ldab   #$ff
srloop    ldaa   SCSR,x
          anda   #$20
      cmpa   #$20
      beq    ldnext
          decb
          cmpb   #$00
      bne    srloop
* srloop  brclr   SCSR,X #$20 srloop   * WAIT
FOR RDRF
ldnext  ldaa   SCDR,X

* THEN DOWNLOAD 256 BYTE PROGRAM

* READ IN PROGRAM AND PUT INTO RAM

          pshy
          ldy    #$00
BK2
      ldab   SCSR,x
          andb   #$20
      cmpb   #$20
      bne    BK2
      ldaa   SCDR,X
      staa   $00,Y
tdwait  ldab   SCSR,x
           andb   #$80
           cmpb   #$80
           bne    tdwait

      staa   SCDR,X
loopw    ldaa SCSR,x
           anda #$20
           cmpa #$00
           bne  loopw



      iny
* UNTIL THE END IS REACHED
      cpy    #$0100
      bne    BK2
      puly
      jmp    NEXT6
********end of sci boot

* PORTD   equ $1008
* DDRD    equ $1009
* SPCR    equ $1028
* SPSR    equ $1029
* SPDR    equ $102A

spiboot  ldx    #$1000
                 ldaa    #$38
                 staa    DDRD,x
                 ldaa    #$d0
                 staa    SPCR,x
                 ldab      #$20
                 stab      PORTD,x


                 LDY     #$0000
                 sty       $0,Y

REPEAT                   ldaa      #$03
                 bclr   PORTD,x #$20
                 staa      SPDR,x
                 ldab      #$0F
WAIT1          decb
                 cmpb  #$00
                 beq       NEXT1
                 ldaa      SPSR,x
                 bpl       WAIT1
NEXT1
                 bset   SPSR,x #$80

                 ldaa      #$10
                 staa      SPDR,x
```

```
                ldab    #$0F
WAIT2       decb
                cmpb  #$00
                beq     NEXT2
                ldaa    SPSR,x
                bpl     WAIT2
NEXT2
                bset    SPSR,x #$80

                sty    $0110
                ldaa   $0111
                staa    SPDR,x
                ldab    #$0F
WAIT3       decb
                cmpb  #$00
                beq     NEXT3
                ldaa    SPSR,x
                bpl     WAIT3
NEXT3
                bset    SPSR,x #$80

                ldaa    #$02
                staa    SPDR,x
                ldab    #$0F
WAIT4       decb
*               cmpb  #$00 , if equal
                beq     NEXT4
                ldaa    SPSR,x
                bpl     WAIT4
                bset    PORTD,x #$20
NEXT4           ldaa    SPDR,X
                bclr    PORTD,x #$20
*           store data in A reg to location pointed
by Y
                staa    $00,Y
                bset    SPSR,x #$80

                INY
                sty    $0110
                ldaa   $0110
                cmpa    #$01
                bne     REPEAT
NEXT5
                bset    SPSR,x #$80

NEXT6
* set BOOTSET
                bset    $01,x, #$03
                jmp     $0000


   org $bffc
   fdb no_err19
   org    $bffe
   fdb    START
```

3 Monitor Code

Note: By KEITH VASILAKES, and modified by OSU HC11 research team.

```
*
*             ' 6811 ML monitor'
*          (c) MARCH 1992 KEITH VASILAKES
*
*   This is a small ml monitor that I origonally wrote on my Commodore 64
*  using a symbolic crossassembler I wrote in 6502 assembly. The assembler
*  was nice if nonstandard and lacking features such as conditional assembly
*  includes, etc. Shortly after finishing 68mon I broke down and
*  bought an Amiga 2000HD, this allowed me to use AS11 and the Buffalo
*  monitor. As it turns out Buffalo is a huuuuge, designed to run on EVB
*  boards and dosn't like other systems. So I reserected 68mon quickly
*  ported it to as11 and here is the result. Its not much but then again
*  its not supposed to be.
*
*    68mon neither requires nor expects expansion ram and uses only five
*  bytes of zero page ram for variables,unless INTERRUPTS is defined which
*  uses another 48 bytes. 68mon keeps track of two stacks,
*  one monitor stack and one user stack. If the INTERRUPTS variable is
*  defined 68mon allows the use of all of the 68HC11 interrupts via a
*  pseudovector system ala Barfalo mon, 68mon however uses different memory
*  locations so be carefull. ( I implemented my vectors before noticeing
*  Buffalo's pattern )
*    68mon supports some standard monitor functions that are
*  listed below including an intell upload ( if defined that is ) for those
*  cases when intell hex makes more sense such as when an s19 file has been
*  converted to intel hex ( such as for my EPROM blaster )and the s19 code
*  doesn't exist.
*
*    Note that 68mon has some usefull functions that can   be called from your
*  assembly language program. these functions include string IO character
*  conversion, and serial port support. See 68mon.h for a complete listing
*
*    Not supported is writing to the eeprom, changing the baud rate  There may
*  be other functions missing, oh well , feel free to add them, and your name
*   to the list at the top. just remember 68mon is supposed to be small and
*  light, make it possible to undefine unnessary code like INTERRUPTS for
*  those who need lots of room.
*
*
*LEGAL STUFF:
*  This program is hereby released into the public domain. It my not be sold
*   in any form for any price. If included with hardware offered for sale,
*   the words "Pubic Domain Monitor 68Mon V1.2" must be clearly visable on all
*   sales literature.
*
*  Usage:
*    Assemble using AS11 or compatable assembler. 68Mon is setup to reside
*    at $E000 but isn"t too picky about where it's at. Programs written to
*    run under 68Mon must end in an SWI or the results are undefined ( crash )
*    note that as soon as an illegal opcode is encountered controll is
*    returned to 68Mon. Be carefull of page zero especially the stack
*    pointers at $00F8 and $00FA
*
*
*
*
* V1.1      ADD S19 UPLOAD  **********DONE******************
```

```
* V1.2      ADD HELP ( LIST COMMANDS ) ***DONE*****************
* V1.?      ADD XON / XOFF ($13 = XOFF, $11 = XON )  YUCK !!!

* allows the use of intell hex uploads
INTELL:

INTERRUPTS:
* define 'INTERRUPTS' to enable the use of the
* pseudo interrupts. comment this out to
* free up 48 bytes of valuable chip RAM
PORTD      EQU  $1008
DDRD       EQU  $1009
SPCR       EQU  $1028
SPSR       EQU  $1029
SPDR       EQU  $102A
BAUD       EQU  $102B
SCCR1      EQU  $102C
SCCR2      EQU  $102D
SCSR       EQU  $102E
SCDAT      EQU  $102F
*
 org $1040     ;org $0000

START

spiboot   ldaa    #$38
                  staa    DDRD
                  ldaa    #$d0
                  staa    SPCR
                  ldab      #$20
                  stab      PORTD


                  LDY       #$1100
                  sty       $0,Y
                  bclr    PORTD,#$20

REPEAT            ldaa      #$03
                  staa      SPDR
                  ldab      #$0F
WAIT1        decb
                  cmpb  #$00
                  beq       NEXT1
                  ldaa      SPSR
                  bpl       WAIT1
NEXT1
                  ldaa      #$80
                  staa      SPSR

                  sty    $0110
                  ldaa   $0110
                  staa      SPDR
                  ldab      #$0F
WAIT2        decb
                  cmpb  #$00
                  beq       NEXT5
                  ldaa      SPSR
                  bpl       WAIT2
NEXT2
                  ldaa      #$80
                  staa      SPSR
```

```
            sty    $0110                                    nop
                   ldaa   $0111                             nop
                   staa       SPDR                         nop
                   ldab       #$0F                         nop
WAIT3          decb                                        nop
                   cmpb  #$00                               nop
                   beq        NEXT5                        nop
                   ldaa       SPSR                         nop
                   bpl        WAIT3                        nop
NEXT3                                                       nop
                   ldaa       #$80                         nop
                   staa       SPSR                         nop
                                                           nop
                   ldaa       #$02                         nop
                   staa       SPDR                         nop
                   ldab       #$0F                         nop
WAIT4          decb                                        nop
                   cmpb  #$00                               nop
                   beq        NEXT5                        nop
                   ldaa       SPSR                         nop
                   bpl        WAIT4                        nop
                                                           nop
NEXT4              ldaa       SPDR                         nop
*           store data in A reg to location pointed by Y    nop
                   staa       $00,Y                        nop
                   ldaa       #$80                         nop
                   staa       SPSR                         nop
                                                           nop
                   INY                                     nop
                   sty    $0110                            nop
                   ldaa   $0110                            nop
                   cmpa       #$17                         nop
                   bne        REPEAT                       nop
                   sty    $0110                            nop
                   ldaa   $0111                            nop
                   cmpa       #$28                         nop
                   bne        REPEAT                       nop
NEXT5                                                      nop
                   ldab       #$20                         nop
                   stab       PORTD                        nop
                   ldaa       #$80                         nop
                   staa       SPSR                         nop
                                                           nop
NEXT6                                                      nop
* set BOOTSET  and disable SELFTEST                        nop
            ldaa   $03                                     nop
            staa   $1001                                   nop
                   nop                                     nop
                   nop                                     nop
                   nop                                     nop
                   nop                                     nop
                   nop                                     nop
                   nop                                     nop
                   nop                                     nop
                   nop                                     nop
                   nop                                     nop
                   nop                                     nop
                   nop                                     nop
                   nop                                     nop
                   nop                                     nop
                   nop                                     nop
                   nop                                     nop
                   nop                                     nop
```

```
                nop
                nop
                nop
                nop
                nop
                nop
                nop
                nop
                nop
                nop
                jmp $2000 *$0100

        END



        ORG  $2000     ;$0100  * pseudo interrupts
similar to Barfalomon but
* notice the different adresses and order !
    IFD INTERRUPTS
PSEUDOVECT  EQU  $1040     ;$0000
PSCI       EQU  0          * serial comunication
interface (RS232)
PSPI       EQU  0003       * sync serial port
PPAC       EQU  0006       * pulse accumulator input
edge
PPACOV     EQU  0009       *  "      "      overflow
PTOF       EQU  $000C      * timer overflow
PTIOC45    EQU  $000F      * in capt 4 / out comp 5
PTOC4      EQU  $0012      * output compare 4
PTOC3      EQU  $0015      *  "     "     3
PTOC2      EQU  $0018      *  "     "     2
PTOC1      EQU  $001B      *  "     "     1
PTIC3      EQU  $001E      * input capture  3
PTIC2      EQU  $0021      *  "     "     2
PTIC1      EQU  $0024      *  "     "     1
PRTI       EQU  $0027      * real time interrupt
PIRQ       EQU  $002A      * irq
PXIRQ      EQU  $002D      * xirq
    ENDIF
;       ORG  $E000      * start of eprom

SP         EQU  $20
CR         EQU  $0D
LF         EQU  $0A
ESC        EQU  $1B
*XON       EQU  $11       cant quite figure this out
yet ????
*XOFF      EQU  $13
************************ZERO PAGE
USAGE**********
TEMPX      EQU  $01FE       ;$00FE
FLAG       EQU  $01FD       ;$00FD
CHECK      EQU  $01FC           ;$00FC   * USED
FOR SUMCHECK
S19FLG     EQU  $01FB           ;$00FB          *
CURRENT LINE IS S9 RECORD
S0FLAG     EQU  $01FA           ;$00FA          *
CURRENT LINE IS S0 RECORD

MONSTACK   EQU  $10F9        ;$00F9
USRSTACK   EQU  $10F7        ;$00F7

****************************************
******
STACK      EQU  $10F6        ;$00F6
USTACK     EQU  $10D0        ;$00D0

COLD       LDS  #STACK
        LDAA #$20
        ORAA SPCR
        STAA SPCR
        LDAA #$30
        STAA BAUD
        LDAA #$0C
        STAA SCCR2
        CLR  FLAG      *CLR ECCO (BIT 7 ) =
ECCO CHARS

   IFD INTERRUPTS

        LDAA #$7E
        LDX  #PSEUDOVECT
        STX  PSCI+1
        STAA PSCI
        STX  PSPI+1
        STAA PSPI
        STX  PPAC+1
        STAA PPAC
        STX  PPACOV+1
        STAA PPACOV
        STX  PTOF+1
        STAA PTOF
        STX  PTIOC45+1
        STAA PTIOC45
        STX  PTOC4+1
        STAA PTOC4
        STX  PTOC3+1
        STAA PTOC3
        STX  PTOC2+1
        STAA PTOC2
        STX  PTOC1+1
        STAA PTOC1
        STX  PTIC3+1
        STAA PTIC3
        STX  PTIC2+1
        STAA PTIC2
        STX  PTIC1+1
        STAA PTIC1
        STX  PRTI+1
        STAA PRTI
        STX  PIRQ+1
        STAA PIRQ
        STX  PXIRQ+1
        STAA PXIRQ

    ENDIF

        CLI

        BRA  MON68

MONSWI     STS  USRSTACK
        LDS  #STACK
        BRA  MON68
```

```
MONTRAP    STS  USRSTACK
           LDS  #STACK
           LDX  #TRAPP
           JSR  PRINT
MON68      LDX  #PROMPT    *SWI CALLS
MONITOR
           JSR  PRINT
           JSR  REG1
MAIN       LDAA #'>'
           JSR  CHROUT
           LDAB #3
           LDX  #CMDTAB
           JSR  CHRIN
           JSR  TOUPPER
           CMPA #CR
           BNE  LOOP
           LDAA #LF
           JSR  CHROUT
           BRA  MAIN
LOOP       CMPA 0,X
           BEQ  CALL
           TST  0,X
           BEQ  NOTFOUND   *END OF TABLE
           ABX
           BRA  LOOP
NOTFOUND   BSR  ERROR
           BRA  MAIN

ERROR      LDAA #'?'
           JSR  CHROUT
           LDAA #CR
           JSR  CHROUT
           LDAA #LF
           JSR  CHROUT
           SEC
           RTS

CALL       LDX  1,X
           JSR  0,X
           BRA  MAIN

PRINT      LDAA 0,X
           BEQ  PREND
           JSR  CHROUT
           INX
           BRA  PRINT
PREND      RTS

INWORD     PSHB
           PSHA
           JSR  INBYTE
           BCS  WRDERR
           TAB
           JSR  INHEX
           BCS  WRDERR
           PSHA
           PSHB
           PULX
           PULA
           PULB
           RTS

WRDERR     PULA
           PULB
           BRA  ERROR

OUTWORD    PSHA
           PSHX
           PULA
           JSR  OUTHEX     *PRINT 2 HEX CHRS
           PULA
           JSR  OUTBYTE    *PRINT 2 HEX CHRS +
SPACE
           PULA
           RTS

INBYTE     BSR  INHEX      *ALLOW LEADING
SPACES
           BCC  INB1
           CMPA #SP
           BEQ  INBYTE
INB1       RTS          *RETURNS W CARRY SET
IF NOT SP OR HEX

INHEX      PSHB
INH1       BSR  CHRIN
           BSR  FASCII
           BCS  INHERR
           TAB
           BSR  CHRIN
           BSR  FASCII
           BCS  INHERR
           ASLB
           ASLB
           ASLB
           ASLB
           ABA
           CLC
INHERR     PULB
           RTS          *RETURNS WITH ERROR
CHAR IN ACCA

OUTHEX     PSHA
           PSHA
           ANDA #$F0
           LSRA
           LSRA
           LSRA
           LSRA
           JSR  TOASCII
           JSR  CHROUT
           PULA
           ANDA #$0F
           JSR  TOASCII
           JSR  CHROUT
           PULA
           RTS

OUTBYTE    BSR  OUTHEX
           PSHA
           LDAA #SP
           JSR  CHROUT
           PULA
           RTS
```

```
FASCII    BSR  TOUPPER
          CMPA #$30
          BLO  GETEND
          CMPA #$39
          BLS  FASC1
          CMPA #$41
          BLO  GETEND
          CMPA #$46
          BHI  GETEND
          SUBA #$07
FASC1     SUBA #$30
          CLC
          RTS

TOASCII   CLC
          ADDA #$90
          DAA
          ADCA #$40
          DAA
          RTS

TOUPPER   CMPA #$61
          BLO  END
          CMPA #$7A
          BHI  END
          SUBA #$20
END       RTS

TOLOWER   CMPA #$41
          BLO  END1
          CMPA #$5A
          BHI  END1
          ADDA #$20
END1      RTS

CHRIN     BSR  GETIN
          BCS  CHRIN
          TST  FLAG
          BMI  END1
          BRA  CHROUT

GETIN     LDAA SCSR
          ANDA #$20
          BEQ  GETEND
          LDAA SCDAT
          CLC
          RTS
GETEND    SEC
          RTS

CHROUT    BSR  PUTOUT
          BCS  CHROUT
          RTS

PUTOUT    PSHB
PUTOUT1   LDAB SCSR
          ANDB #$80
          BEQ  PUTOUT2
          STAA SCDAT
          PULB
          CLC
          RTS
PUTOUT2   PULB
          SEC
          RTS

CMDTAB    FCB 'M'
          FDB  MEMEX
          FCB 'G'
          FDB  GO
        IFD INTELL
          FCB 'U'
          FDB UPLOAD
        ENDIF
          FCB 'F'
          FDB FILL
          FCB 'R'
          FDB REGISTER
          FCB 'C'
          FDB CONTINUE
          FCB 'S'
          FDB S19UPLOAD
          FCB '?'
          FDB HELP
          FCB 0

MEMEX     JSR  INWORD
          BCS  ERR
          STX  TEMPX
MEMX      JSR  INBYTE    *CHANGE MEMORY
          BCS  READ0
          STAA 0,X
          INX
          BRA  MEMX

READ0     CMPA #ESC
          BEQ  ERR
          LDX  TEMPX
          LDAA #CR
          JSR  CHROUT
          LDAA #LF
          JSR CHROUT
READ      JSR  OUTWORD   *PRINT ADDRESS
          LDAB #$10      *NUMBER OF BYTES
PER LINE
READ1     LDAA 0,X
          JSR  OUTBYTE
          INX
          DECB
          BNE  READ1
          LDX  TEMPX
          LDAB #$10
READ2     LDAA 0,X
          CMPA #SP
          BLO  READ4
          CMPA #$80
          BLS  READ3
READ4     LDAA #'.'
READ3     JSR  CHROUT
          INX
          DECB
          BNE  READ2
          STX  TEMPX
```

```
        JSR  GETIN     *PRINT DATA UNTIL
KEYPRESS
        BCS  READ0
READ5   LDAA #CR
        JSR  CHROUT
        LDAA #LF
        JSR  CHROUT
        RTS


FILLERR PULX
ERR     SEC
        RTS

FILL    JSR  INWORD    *GET FROM ADDRESS
        BCS  ERR
        PSHX
        JSR  INWORD    *GET TO ADDRESS
        BCS  FILLERR
        STX  TEMPX
        JSR  INBYTE    *GET FILL VALUE
        BCC  FILLX
        LDAA #$FF      *IF NO FILL VALUE,FILL
WITH NOP'S
FILLX   PULX
FILL1   STAA 0,X
        INX
        CPX  TEMPX
        BLS  FILL1
        LDAA #CR
        JSR  CHROUT
        LDAA #LF
        JMP  CHROUT

    IFD INTELL

UPLOAD  LDAA FLAG
        PSHA
        ORAA #%10000000  *BIT 7 SET = NO
ECCO
        STAA FLAG
        LDAA #CR
        JSR  CHROUT
        LDAA #LF
        JSR  CHROUT
        LDAA #'.'
        JSR  CHROUT
UPSTART JSR  CHRIN
        BCS  ERROR1
        CMPA #':'
        BNE  UPEND
        CLR  CHECK
        JSR  INBYTE    *GET NUM OF BYTES
        BCS  ERROR1
        PSHA           *SAVE NUMBER OF BYTES
(WILL BE USED IN ACCB LATER)
        ADDA CHECK
        STAA CHECK
        JSR  INWORD    *GET ADDRESS
        BCC  UPOK
        PULA
        BRA  ERROR1

UPOK    PSHX
        XGDX
        ADDA CHECK
        STAA CHECK
        ADDB CHECK
        STAB CHECK
        PULX
        PULB           *GET BACK NUMBER OF
FCBS
        JSR  INBYTE    *GET NULL (?) FCB
        ADDA CHECK
        STAA CHECK
        INCB
UPLOAD1 JSR  INBYTE
        BCS  ERROR1
        DECB
        BEQ  END2
        STAA 0,X
        ADDA CHECK
        STAA CHECK
        INX
        BRA  UPLOAD1
END2    NEG  CHECK
        CMPA CHECK
        BNE  ERROR1
        JSR  CHRIN     *GETCR AT END OF LINE
        BRA  UPSTART
UPEND   PULA
        STAA FLAG
        LDAA #1
        ORAA FLAG
        STAA FLAG
        RTS
ERROR1  PULA
        STAA FLAG
        JMP  ERROR

    ENDIF

REGISTER JSR  CHRIN
        CMPA #CR
        BEQ  REG1
        BRA  GOERR
REG1    LDAA #LF
        JSR  CHROUT
        LDX  #REGNAME
        JSR  PRINT
        LDAA #SP
        JSR  CHROUT
        JSR  CHROUT
        LDX  USRSTACK
        INX
        LDAB 0,X
        LDAA #8
        STAA TEMPX
REG     CLRA
        ASLB
        ADCA #'0'
        JSR  CHROUT
        DEC  TEMPX
        BNE  REG
        LDAA #SP
```

```
        JSR  CHROUT
        JSR  CHROUT
        LDAA 1,X
        JSR  OUTBYTE   *ACCB
        LDAA 2,X
        JSR  OUTBYTE   *ACCA
        LDD  3,X
        XGDX
        JSR  OUTWORD   *INX
        XGDX
        LDD  5,X
        XGDX
        JSR  OUTWORD   *INY
        XGDX
        LDD  7,X
        XGDX
        JSR  OUTWORD   *PC
        LDX  USRSTACK
        JSR  OUTWORD   *SP
        LDAA #CR
        JSR  CHROUT
        LDAA #LF
        JSR  CHROUT
        RTS

GO      JSR  INWORD
        BCS  GOERR
        JSR  CHRIN
        CMPA #CR
        BNE  GOERR
        LDS  #USTACK
        JMP  0,X       * CALL USER PROG,X

GOERR   RTS

CONTINUE   JSR  CHRIN
        CMPA #CR
        BNE  GOERR
        LDS  USRSTACK  * CONTINUE FROM
LAST SWI
        RTI

HELP    LDX  #HELPLIST
        JSR  PRINT
        RTS
HELPLIST   FCB
CR,LF,CR,LF,CR,LF,CR,LF,CR,LF
        FCC  'COMMANDS:'
        FCB  CR,LF,CR,LF
        FCB  CR,LF
        FCB  CR,LF
        FCC  '?       DISPLAYS THIS SCREEN',CR
        FCB  CR,LF,CR,LF
        FCC  'F [xxxx] [yyyy] [zz] FILL MEMORY
FROM xxxx to yyyy with zz '
        FCB  CR,LF,CR,LF
        FCC  'M [xxxx] MEMORY EXAMINE,
DISPLAYS DATA AT xxxx UNTIL ANY KEY IS
PRESSED'
        FCB  CR,LF,CR,LF
        FCC  'M [xxxx] [yy zz ...] MEMORY
CHANGE, WRITES yy TO xxxx AND zz TO xxxx
+1'
        FCB  CR,LF,CR,LF
        FCC  'G xxxx   TRANSFERS CONTROLL
TO PROGAM AT xxxx. PROG IS GIVEN ITS OWN'
        FCB  CR,LF,CR,LF
        FCC  '         STACK AND MUST END WITH
A SWI TO RETURN TO THE MONITOR'
        FCB  CR,LF,CR,LF
        FCC  'R       DISPLAYS USER REGISTERS'
        FCB  CR,LF,CR,LF
        FCC  'C       CONTINUES A USER
PROGRAM AFTER A SWI, LIKE A BREAKPOINT'
        FCB  CR,LF,CR,LF
        FCC  'S       UPLOADS A MOTO S19 HEX
FILE'
        FCB  CR,LF,CR,LF
    IFD INTELL
        FCC  'U       UPLOADS AN INTEL HEX
FILE'
        FCB  CR,LF,CR,LF
    ENDIF
        FCB  CR,LF
        FCB  0
;
TRAPP      FCB CR,LF
        FCB CR,LF
        FCC '   ******** ILLEGAL OPCODE TRAP
!!! ********'
        FCB CR,LF
        FCB 0
PROMPT     FCB CR,LF
        FCC '   68Mon V1.2 (C) 1992 Keith
Vasilakes'
        FCB CR,LF
        FCB 0
REGNAME    FCC ' SXHINZVC  AB AA  IX   IY
PC   SP'
*               12345678  12 12 1234 1234 1234 1234
        FCB CR,LF
        FCB 0
;
S19UPLOAD   LDAA FLAG
        PSHA
*         ORAA #%10000000 * BIT 7 SET = NO
ECCO
*          STAA FLAG
        LDAA #CR
        JSR  CHROUT
        LDAA #LF
        JSR  CHROUT
SUPSTART   CLR  S19FLG
                              CLR  S0FLAG
        JSR  CHRIN
```

5 Simple Test Code to Use Port B pin 0 to Perform RS232 TX Function

```
*****************************************************************************
*  TxPB.asm
*
*           Simple program for testing with the HC11 die 2008, where Port B pin 0 as SCI TX pin .
*           A text string is sent to the terminal using COM1.
*

***************
*   EQUATES   *
***************
REGBS   EQU  $1000

BAUD    EQU  REGBS+$2B
SCCR1   EQU  REGBS+$2C
SCCR2   EQU  REGBS+$2D
SCSR    EQU  REGBS+$2E
SCDAT   EQU  REGBS+$2F
COPRST  EQU  REGBS+$3A
PORTB   EQU  REGBS+$04
*PORTE   EQU  REGBS+$0A
PORTD   EQU  REGBS+$08
DDRD    EQU  REGBS+$09
*PORTA   EQU  REGBS+$00

IBUFSIZ EQU  35
SPEED   EQU  15
EOT     EQU  $04
CR      EQU  $0D
LF      EQU  $0A
high    EQU  $FF
loop    EQU  $5C

************************
* Program starts here *
************************
       ORG  $0000
START
       LDS  #$0FA0
                                        LDAA  #$3E
       STAA  DDRD

             JSR  ONSCI
             jsr  readyPB
       ldx  #hello
       jsr  outstrgB

             ldaa        #loop
eloop        deca
             bne         eloop

      bra START

       RTS


**********
*   ONSCI() - Initialize the SCI for 9600
```

```
*               baud at 8 MHz Extal.
**********
ONSCI   LDAA #$30
        STAA BAUD
        LDAA #$00
        STAA SCCR1
        LDAA #$0C
        STAA SCCR2
        RTS

**********
*   readyPB() - Initialize the Port B pin 0 for Tx 4800
*               baud at 8 MHz Extal.
**********
readyPB
              ldaa #$01
              staa PORTB
              jsr  bit_delay
              rts

**********
*  OUTSTRGB(x) - Output string of ASCII bytes
* starting at x until end of text ($04) via PB[0].
**********
outstrgB PSHA
out1 ldaa 0,X
        cmpa #EOT
        beq  outstrg2
              jsr  bit_bang
        inx
        bra  out1
outstrg2 PULA
        RTS

**********
*  OUTPUT() - Port B as Tx pin (bit bang).
**********

bit_bang
       pshy
       ldy  #$1000
start_bit
              ldab #$08
              bclr $04,y #$01
              jsr  bit_delay
shiftR
              lsra
              bcs  high_out
              bcc  low_out
low_out
              bclr $04,y #$01
              jsr  bit_delay
              bra  bottomlsr
high_out
              bset $04,y #$01
              jsr  bit_delay
              bra  bottomlsr
bottomlsr
              decb
              bne  shiftR
              bset $04,y #$01
              jsr  bit_delay
              puly
              rts
bit_delay
              pshx
              ldx          #SPEED
dly:          dex
              nop
              nop
              bne          dly
              nop
              nop
              pulx
              rts

*** TEXT TABLES ***
hello   FCB   CR,LF
              FCC   'Hello World, I am from portB[0]'
        FCB   EOT
*reset vector
        org     $BFFE
        fdb     START


         END
```

APPENDIX 2

CONTROLLER CORE
ALU
Multiplier
Standby Control

DOCUMENTS

TABLE OF CONTENTS

DOCUMENT **PAGE**

1 Description

The main controller implements the entire 68HC11 instruction set and controls the various peripherals and the memory. The Controller core implements the OSU 68HC11 instruction set using a generic ALU, an array multiplier and a clock control circuit to implement power control as explained in the sections below.

1.1 Controller Core

The M68HC11 Family of microcontrollers uses 8-bit opcodes. Each opcode identifies a particular instruction and associated addressing mode to the CPU. Several opcodes are required to provide each instruction with a range of addressing capabilities. There are a total of 311 68HC11 opcodes. With an 8-bit number spanning 256 values, it is clear that to implement 311 opcodes requires a scheme using a "special" opcode to indicate that the real opcode is in a different table (called a page). This special opcode is called a "prebyte" since it is a special number which is seen before the actual page-N opcode. It is interesting to observe that for the 68HC11 opcode mapping, instead of implementing a single prebyte that specifies to find the opcode on a second page, it implements 3 different prebytes which take us to 3 different auxiliary pages. As implemented, the primary page (page 0) includes 233 valid opcodes plus 3 prebyte codes, resulting in 20 undefined opcodes. If the processor encounters one of these undefined opcodes while running (for example, the number $42 -- in hex notation of course) it will throw an Illegal OpCode exception. Page 1 below includes 64 valid opcodes, page 2 has 7, and page 3 only has 4 opcodes defined. Refer to the **M68HC11 Reference Manual** for Instruction Set Details for more information.

Out of the entire instruction set the controller does not implement the ADC instructions for obvious reasons and the DIV instruction (division could be implemented in software through the vast majority of software library available). Below is a block diagram of the overall architecture of the controller.

Six addressing modes can be used to access memory: Immediate, Direct, Extended, Indexed, Inherent, and Relative. All modes except inherent mode use an effective address in the operand, which are 1 or more bytes following the opcode. The effective address is the memory address from which the argument is fetched or stored or the address from

which execution is to proceed. The effective address can be specified within an instruction, or it can be calculated. Depending on the exact instruction, the operand for the particular addressing mode will be determined.

For more information on CPU, please refer to section 6 of the M68HC11 reference manual from (Rev.6, 04/2002) Motorola (www.freescale.com)

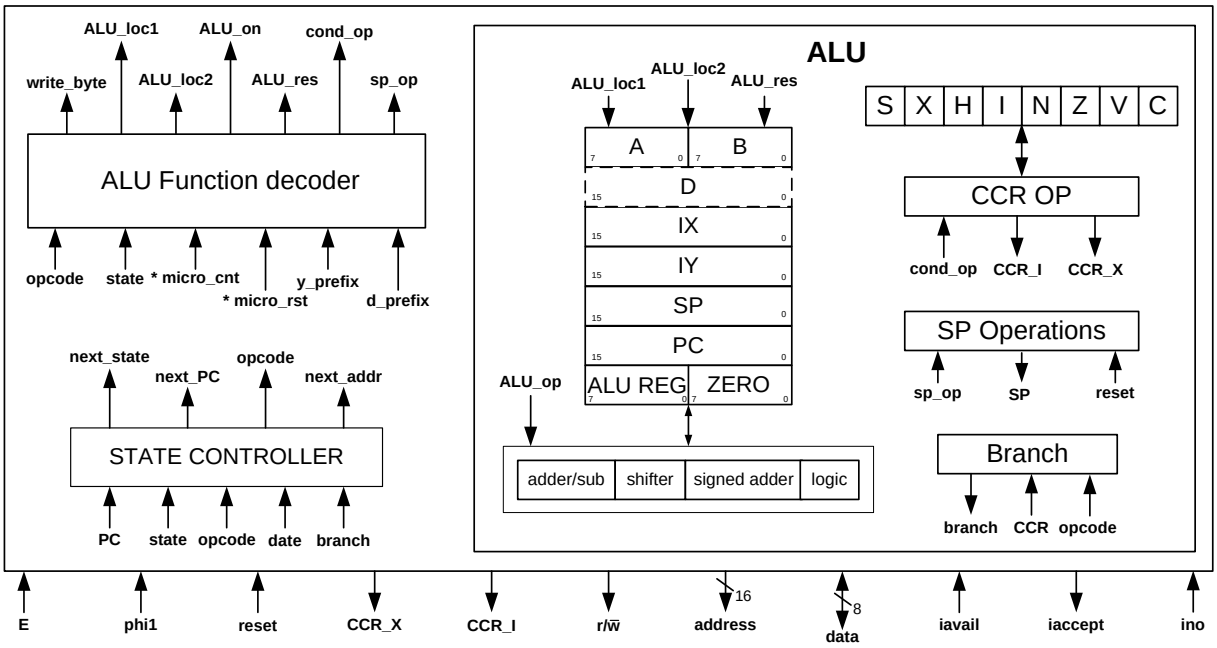


Figure 1 Controller Architecture Block Diagram

1.1.1 Legend for the Block Diagram:

S.No	All Possible states the CPU could be in	All ALU Operations	Locations from which the ALU can get its input data or write back the result to	Various Operations that could set the CCR	List of possible stack operations
	state/next_state	**ALU_op**	**ALU_loc1, ALU_loc2, ALU_res**	**cond_op**	**SP_op**
1	INIT	PASS	ZERO	PASS	PASS SP
2	FETCH-1	ADD	ALU_REG	ADD8	SET SP
3	FETCH-2	ADDC	ACCA	ADD16	INC SP
4	FETCH_AGAIN	AND	ACCB	LOGIC8	DEC SP
5	FETCH-3	LSL	ACCD	SHIFT-L8	
6	FETCH-4	LSR	IX	SHIFT-R8	
7	LOAD-1	ASR	IY	SHIFT-L16	
8	LOAD-2	CLR	SPC	SHIFT-R16	
9	IGNORE	OR	IMM8	SUB8	
10	IGNORE-2	SUB	IMM16	SUB16	
11	CALCADDR	PASS2	ANT_IMM8	CLR	
12	WRITE	XOR	SSP	SET	
13	WRITE-2	SADD16	ONE	NEG	
14	WAIT_INT	SUBC	BIT1	DA	
15	INT-1	ROL	BIT4	NZV	
16	INT-2	ROR	NEGONE	Z16	
17			DEC_ADJ	LOAD16	
18			IXH	ADDLO	
19			IYH	RESTORE	
20			BUS_DATA		
21			SCCR		

1.1.2 Test Structure Modifications:

The controller test structure has a lot of registers that have to be examined to verify correct operation and several of them have to given new values before one can proceed to the next cycle in case the previous operation did not perform properly.

As the number of the pads available is limited all observe only registers were connected to a Serial shift out/Parallel load register chain. Similarly all observe and

control registers have been connected through a scan chain and can be scanned out and new values scanned in serially. Both these structures are controlled by the same clock and the shift register' parallel load is controlled by a LOAD signal while the scan is enabled by the SCEN signal. While the scan is enabled the main E clock is stopped to prevent any change in the register's state affecting the controller.

The **Shift Register**'s **80-bit** Parallel Input is split as follows:

```
shiftdata[15:0]   = address;
shiftdata[19:16]  = decode_aluop_out;
shiftdata[24:20]  = decode_condop_out;
shiftdata[29:25]  = decode_aluin1_out;
shiftdata[34:30]  = decode_aluin2_out;
shiftdata[37:35]  = decode_alures_out[3:0];
shiftdata[38]     = decode_alures_out[4];
shiftdata[39]     = rw;
shiftdata[41:40]  = decode_spop_out;
shiftdata[57:42]  = PC;
shiftdata[65:58]  = write_data;
shiftdata[69:66]  = debug_micro;
shiftdata[70]     = control_start;
shiftdata[71]     = iaccept;
shiftdata[79:72]  = opcode;
```

The **Scan Chain** with **88 bits** is connected as follows:
SIctrl → A(8) → B(8) → X(16) → Y(16) → ALUREG(16) → CCR(8) → SP(16) → SOcrtl

During normal operation the scan chain is completely scanned out through SOctrl for verification of register contents. At the same time the Serial out SOctrl is looped back in through the Serial In SIctrl. Due to this at the end of 88 clock cycles all the registers will be restored to their original state so that the controller can proceed with the next instruction. If something is found to be problematic and the registers don't have the proper values then it will take another 88 clock cycles to feed in the desired values and then controller's clock is fed in to restart normal operation.

1.1.3 Interface with External Modules:

The controller mainly interfaces with two modules namely a) Register File b) Interrupt Controller c) Math Co-Processor.

a) Register File:

The controller reads all its input from either the data bus in case of a memory load/immediate data or from its internal register file which is used to store user variables as well as certain machine related registers such as Program Counter (PC) and Stack Pointer (SP). The register file also includes two special 16-bit index registers IX & IY which are used in the indexed addressing mode. Below is a block

diagram of the controller control input signals and how they are routed to the ALU/multiplier inputs.

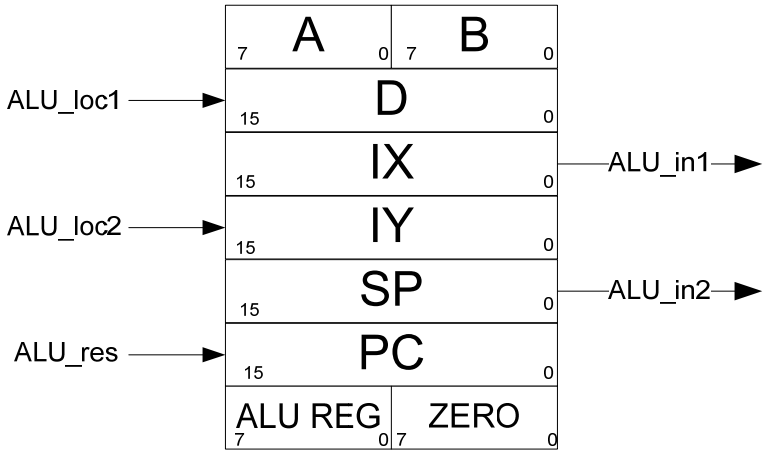


Figure 2 Register File ALU interface.

b) Interrupt Controller:

The interrupt controller receives all interrupts from various sources (Timer, COP, External, etc.), prioritizes them and interrupts the controller whenever an interrupt is available. The interrupt controller indicates that an interrupts is available by asserting the **iavail** line high and giving the interrupt number in **ino[3:0].** Once the controller receives this signal it processes the current instruction it is processing and then accepts the interrupt and goes on to process the ISR after asserting the **iaccept** line low to indicate to the interrupt controller that the interrupt has been **processed** and can be **cleared.** The interrupt controller is also provided with the **X** and **I** bits from the **CCR** to check with the status of these registers before issuing an **iavail** signal.

c) Math Co-Processor:

This mainly consists of the interface with the multiplier and the control signals used to control its input and output and are explained in the section on multiplier below.

1.2 ALU

The ALU implements all the arithmetic operations and controls the CCR (Condition Code Register) based on the results of the current operation. Due to the fact that multiple instructions

1.2.1 Internal Block Diagram of the ALU:

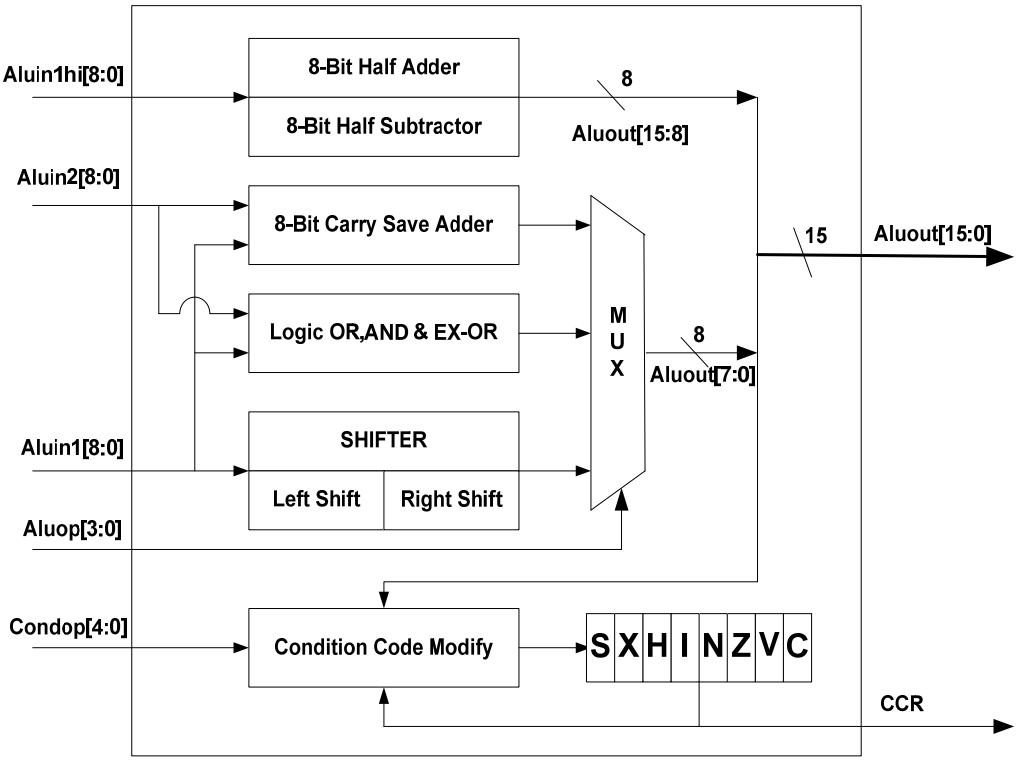


Figure 3 ALU internal block diagram.

1.2.2 Operations Performed by the ALU:

a. ALU OPERATIONS:

ALU Operation	OPCODE				Explanation
PASS1	0	0	0	0	Pass IN1 to output
SADD16	0	0	0	1	16-Bit Signed Addition (X +127/ X- 128) (Relative)
CLR	0	0	1	0	Sets Output Register to 00h
PASS2	0	0	1	1	Pass IN2 to Output
ASL	0	1	0	0	Arithmetic Shift Left IN1
ROL	0	1	0	1	Rotate Left IN1
ASR	0	1	1	0	Arithmetic Shift Right IN1
LSR	0	1	1	1	Logical Right Shift IN1
ROR	1	0	0	0	Rotate Right IN1
OR	1	0	0	1	Logical OR IN1 & IN2 (IN1+IN2)
AND	1	0	1	0	Logical And IN1 & IN2 (IN1.IN2)
XOR	1	0	1	1	Exclusive-OR IN1 & IN2
ADD	1	1	0	0	Arithmetic Add IN1 & IN2
ADDC	1	1	0	1	Arithmetic Add IN1 & IN2 with Carry from CCR
SUBC	1	1	1	0	Arithmetic Subtract IN1 & IN2 with Borrow from CCR
SUB	1	1	1	1	Arithmetic Subtract IN1 & IN2
0 C	Left shift Mux				
10 - b7 11 - 0 00 - C	Right Shift Mux				
	EX-OR for Subtract (to do One's Complement)				

b. Cin Bit for **Addition/Subtraction** is calculated as follows:

OP[1]	OP[0]	CBIT	CIN	ALU OPERATION
0	0	0	0	ADD
0	1	0	0	ADDC
1	0	0	1	SUBC
1	1	0	1	SUB
0	0	1	0	ADD
0	1	1	1	ADDC
1	0	1	0	SUBC
1	1	1	1	SUB
Switched SUB & SUBC for convenience of combining the minterms for the carry bit.				
Previously the CBIT was A~BC+~ABC+A~C.				
Now after Interchange reduced to A~C+BC. (A = OP[1], B = OP[2], C=CBIT)				

c. **Shift Operations Performed By the ALU**:

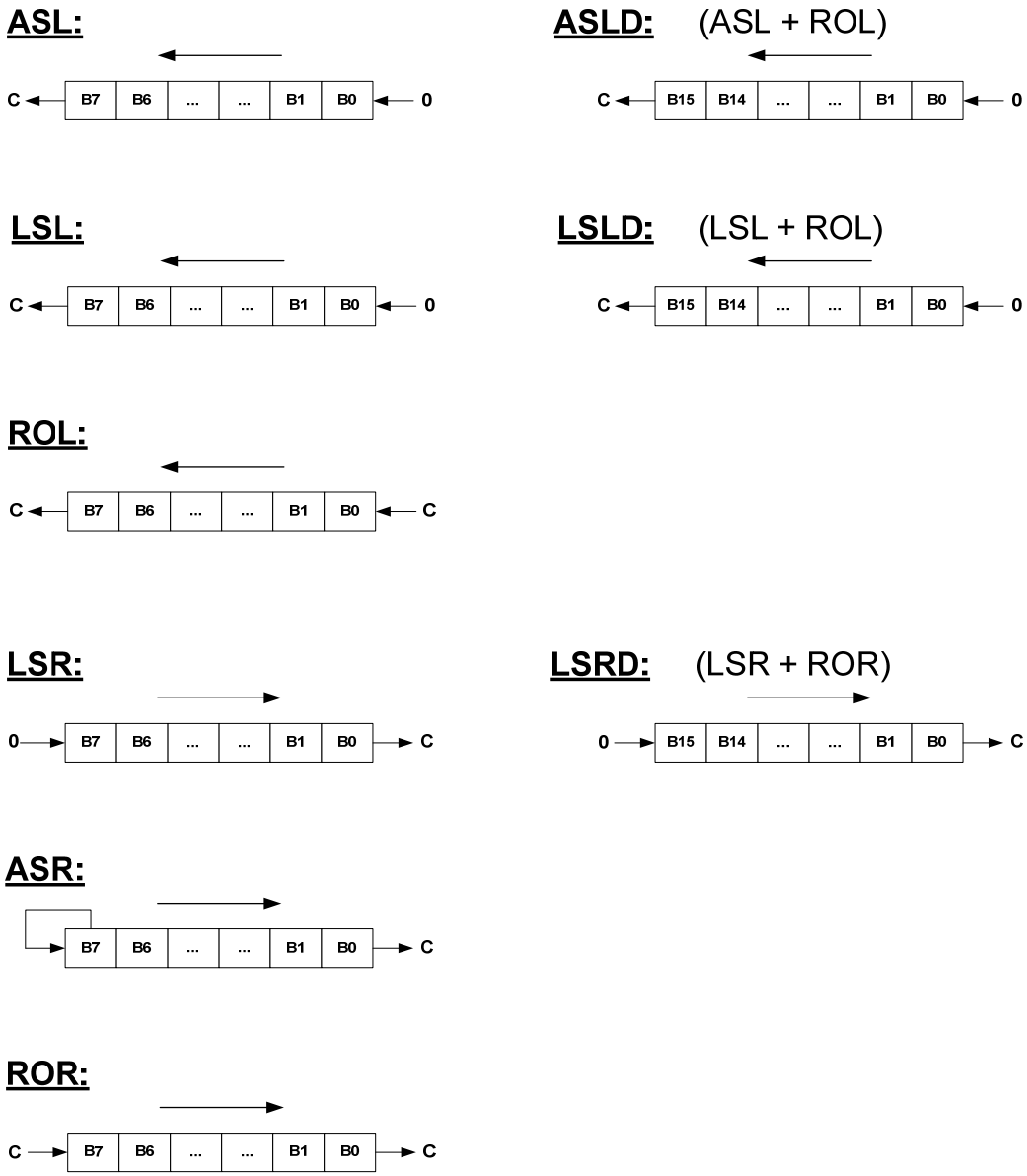


Figure 4 Shift Operations performed by ALU

d. **CCR modifying Operations Performed By the ALU**:

SHIFTR8: (LSR,ASR,ROR) N = R7 = 0 (Cleared) Z = $\overline{R7}.\overline{R6}.\overline{R5}.\overline{R4}.\overline{R3}.\overline{R2}.\overline{R1}.\overline{R0}$ V = N ⊕ C C = LSB of ACCX/M Before Shift (M0)	**SHIFTR16:** (LSRD) (referred to second byte) N = 0 (Cleared) Z = $\overline{R15}.\overline{R14}.\overline{R13}.....\overline{R3}.\overline{R2}.\overline{R1}.\overline{R0}$ V = N ⊕ C = C = D0 C = LSB of ACCD Before Shift (D0)
SHIFTL8: (LSL,ASL,ROL) N = R7 Z = $\overline{R7}.\overline{R6}.\overline{R5}.\overline{R4}.\overline{R3}.\overline{R2}.\overline{R1}.\overline{R0}$ V = N ⊕ C C = MSB of ACCX/M Before Shift (M7)	**SHIFTL16:** (LSLD) (referred to second byte) N = R7 Z = $\overline{R15}.\overline{R14}.\overline{R13}.....\overline{R3}.\overline{R2}.\overline{R1}.\overline{R0}$ V = N ⊕ C C = MSB of ACCX/M Before Shift (D15)
LOGIC8: (OR,AND,XOR) N = R7 Z = $\overline{R7}.\overline{R6}.\overline{R5}.\overline{R4}.\overline{R3}.\overline{R2}.\overline{R1}.\overline{R0}$ V = 0 (Cleared)	**ADD8:** (ADD) H = $X3.M3 + M3.\overline{R3} + \overline{R3}.X3$ N = R7 Z = $\overline{R7}.\overline{R6}.\overline{R5}.\overline{R4}.\overline{R3}.\overline{R2}.\overline{R1}.\overline{R0}$ V = $X7.M7.\overline{R7} + \overline{X7}.\overline{M7}.R7$ C = $X7.M7 + M7.\overline{R7} + \overline{R7}.X7$
ADD16: (referred to second byte) N = R7 Z = $\overline{R15}.\overline{R14}.\overline{R13}.....\overline{R3}.\overline{R2}.\overline{R1}.\overline{R0}$ V = $X7.M7.\overline{R7} + \overline{X7}.\overline{M7}.R7$ C = $X7.M7 + M7.\overline{R7} + \overline{R7}.X7$	**ADDLO:** N = R7 Z = $\overline{R7}.\overline{R6}.\overline{R5}.\overline{R4}.\overline{R3}.\overline{R2}.\overline{R1}.\overline{R0}$ V = $X7.M7.\overline{R7} + \overline{X7}.\overline{M7}.R7$ C = $X7.M7 + M7.\overline{R7} + \overline{R7}.X7$
SUB8: (SUB) N = R7 Z = $\overline{R7}.\overline{R6}.\overline{R5}.\overline{R4}.\overline{R3}.\overline{R2}.\overline{R1}.\overline{R0}$ V = $X7.\overline{M7}.\overline{R7} + \overline{X7}.M7.R7$ C = $\overline{X7}.M7 + M7.R7 + R7.\overline{X7}$	**SUB16:** (referred to second byte) N = R7 Z = $\overline{R15}.\overline{R14}.\overline{R13}.....\overline{R3}.\overline{R2}.\overline{R1}.\overline{R0}$ V = $X7.\overline{M7}.\overline{R7} + \overline{X7}.M7.R7$ C = $\overline{X7}.M7 + M7.R7 + R7.\overline{X7}$
NEG: N = R7 Z = $\overline{R7}.\overline{R6}.\overline{R5}.\overline{R4}.\overline{R3}.\overline{R2}.\overline{R1}.\overline{R0}$ V = 0 (Cleared) C = 1 (Set)	**NZV:** N = R7 Z = $\overline{R7}.\overline{R6}.\overline{R5}.\overline{R4}.\overline{R3}.\overline{R2}.\overline{R1}.\overline{R0}$ V = $X7.M7.\overline{R7} + \overline{X7}.\overline{M7}.R7$
Z16: Z = $\overline{R15}.\overline{R14}.\overline{R13}.....\overline{R3}.\overline{R2}.\overline{R1}.\overline{R0}$	**CLR:** CCR = CCR and $\overline{X}$
SET: CCR = CCR or X	**RESTORE:** CCR [XBIT] = CCR[XBIT] and X[XBIT] CCR[S,H,I,N,Z,V,C] = X[S,H,I,N,Z,V,C]
LOAD16:	**DA:** (not implemented in final)

Z = $\overline{R15}.\overline{R14}.\overline{R13}.....\overline{R3}.\overline{R2}.\overline{R1}.\overline{R0}$ N = R15 V = 0 (Cleared)	Z = $\overline{R7}.\overline{R6}.\overline{R5}.\overline{R4}.\overline{R3}.\overline{R2}.\overline{R1}.\overline{R0}$ N = R7 C = Alu_Carry_Out

1.3 Multiplier

The controller includes an 8-by-8 array multiplier that is used to perform multiplications at hardware speed and is much faster compared to the 10 cycle multiply implemented by the original M68HC11. The block diagram below shows the structure of the array multiplier that was built with full adders.

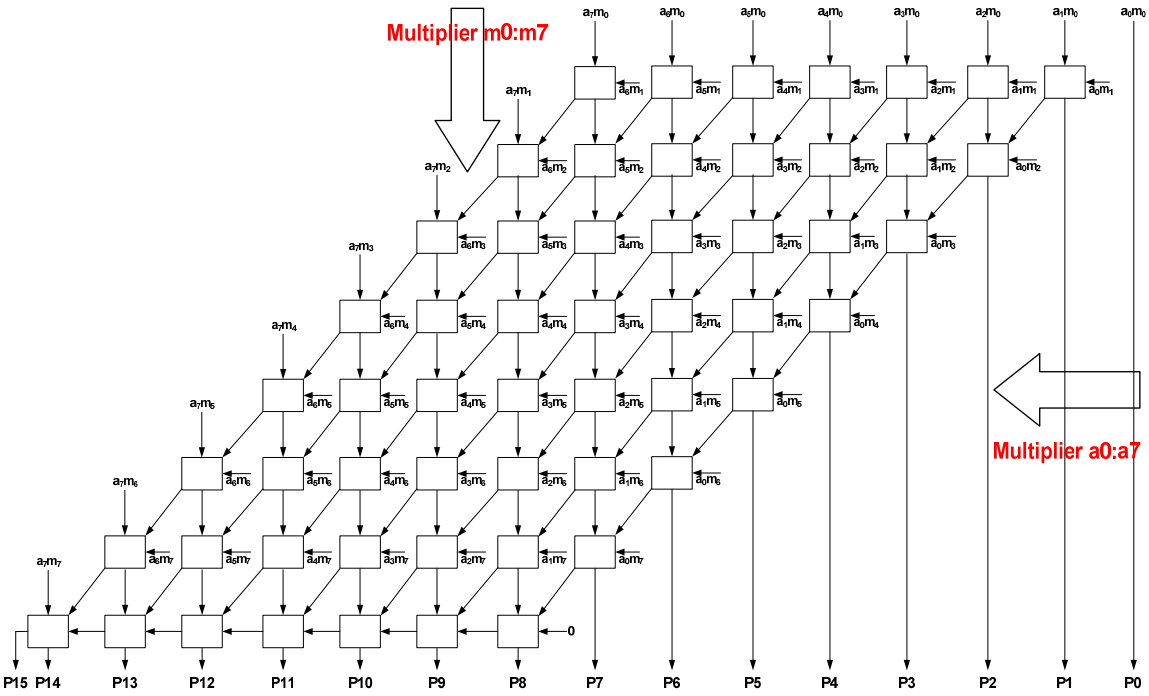


Figure 5 8-by-8 Array multiplier structure.

The multiplier does not have a ripple adder for the lower byte (8-bits) to reduce hardware duplication as the ALU already has a fast carry-select adder built inside. The two bytes of the LSB to be added are sent to the ALU and then the ALU adds them and writes the result back to the register. A small connection diagram showing the interconnection of the controller, ALU and multiplier is shown below.

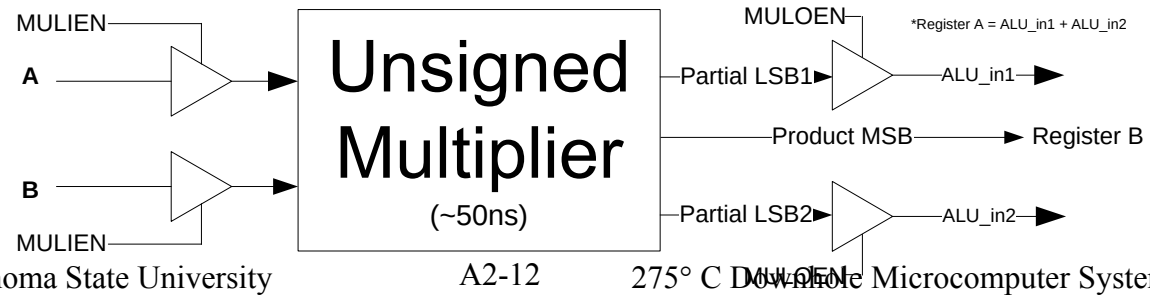


Figure 6 Multiplier, Controller & ALU interface

Original – 10 Clock Cycles. Enhanced version – 3 Clock Cycles.

Cycle Operation
1 - Fetch & Decode
2 - Input is Enabled (Multiply Starts)
3 - Output is Enabled (MSB goes to adder, LSB is written to LSB of D(B)), MSB after addition in ALU is written into the MSB of D(A) and CCR carry bit is modified

1.4 Standby Control

Power Saving mode is entered in HC11 by the execution of the STOP instructions. This instruction sets a special internal register which is read by the SRAM module to turn down its supply voltage and go to standby mode and also at the same time cut the clock input to the processor and all modules. The major power savers in the special low power mode implemented in the HC11 are two fold:

1) Low Static Power adopted where possible (This is done by decreasing the supply voltage of the SRAM block since this tends to be the leakiest block due to its cell capacity).
2) The Clock is cut-off to the entire chip until an interrupt occurs and then the interrupt controller (which is asynchronous and has no clock) enables back the clock to the core and peripherals and they start executing from where they left.

2 Pins

CPU and Module Interface Connections

Port	Width	Direction	Description
PORTAout	8	input	Data from the module, port A, to CPU
PORTDout	8	input	Data from the module, port D, to CPU
PORTBCout	8	input	Data from the module, port B and C, to CPU
RAMout	8	input	Data from the module, RAM, to CPU
ROMout	8	input	Data from the module, ROM, to CPU
Dout	8	output	Data to the module from CPU
Addressout	16	output	Core/interface address
IOSEL	1	output	IO select
Reset	1	input	Module reset
rstaddr	3	input	Reset address from interrupt controller
E	1	input	E-clock input
phi1	1	input	phi1-clock input
RWout	1	input	read/write control signal
iavail	1	input	Interrupt available flag from interrupt controller
iaccept	1	output	Interrupt accepted indicator
ino	5	input	Interrupt vector address from interrupt controller
CCRX	1	output	X-bit of CCR for interrupt handling
CCRI	1	output	I-bit of CCR for interrupt handling
XIRQ_ctrl	1	input	XIRQ/IRQ indicator from interrupt controller
STOP	1	output	STOP control signal to control system Clock and

			the SRAM standby control
SIctrl	1	Input	Serial scan input for the scan chain
SOctrl	1	Output	Serial scan output from the scan chain
SROctrl	1	Output	Shift Register output from second scan chain
SCKctrl	1	Input	Scan Clock input
SCENctrl	1	Input	Scan Enable
SI	1	Input	Serial scan input for the scan chain, bypass CPU core.
SO	1	Output	Serial scan output from the scan chain, bypass CPU core.
SCK	1	Input	Scan Clock input, bypass CPU core.
SCEN	1	Input	Scan Enable, bypass CPU core.

3 Testing and Simulation

Simulator: Verilog XL and AMS Ultrasim

The functional test is conducted using Verilog XL. The timing and functional test with parasitic capacitance for the RAM and ROM (using their layout views) was conducted in the AMS Ultrasim environment.

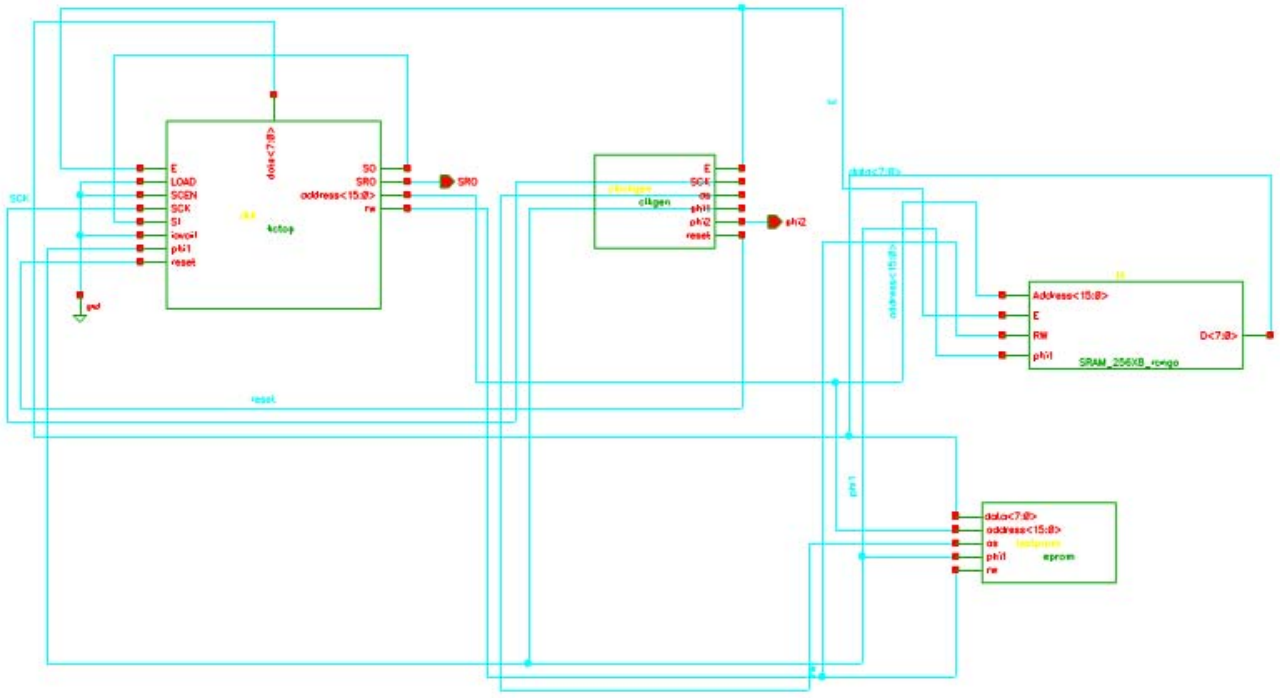


Figure 7 CPU's Ultrasim simulation setup.

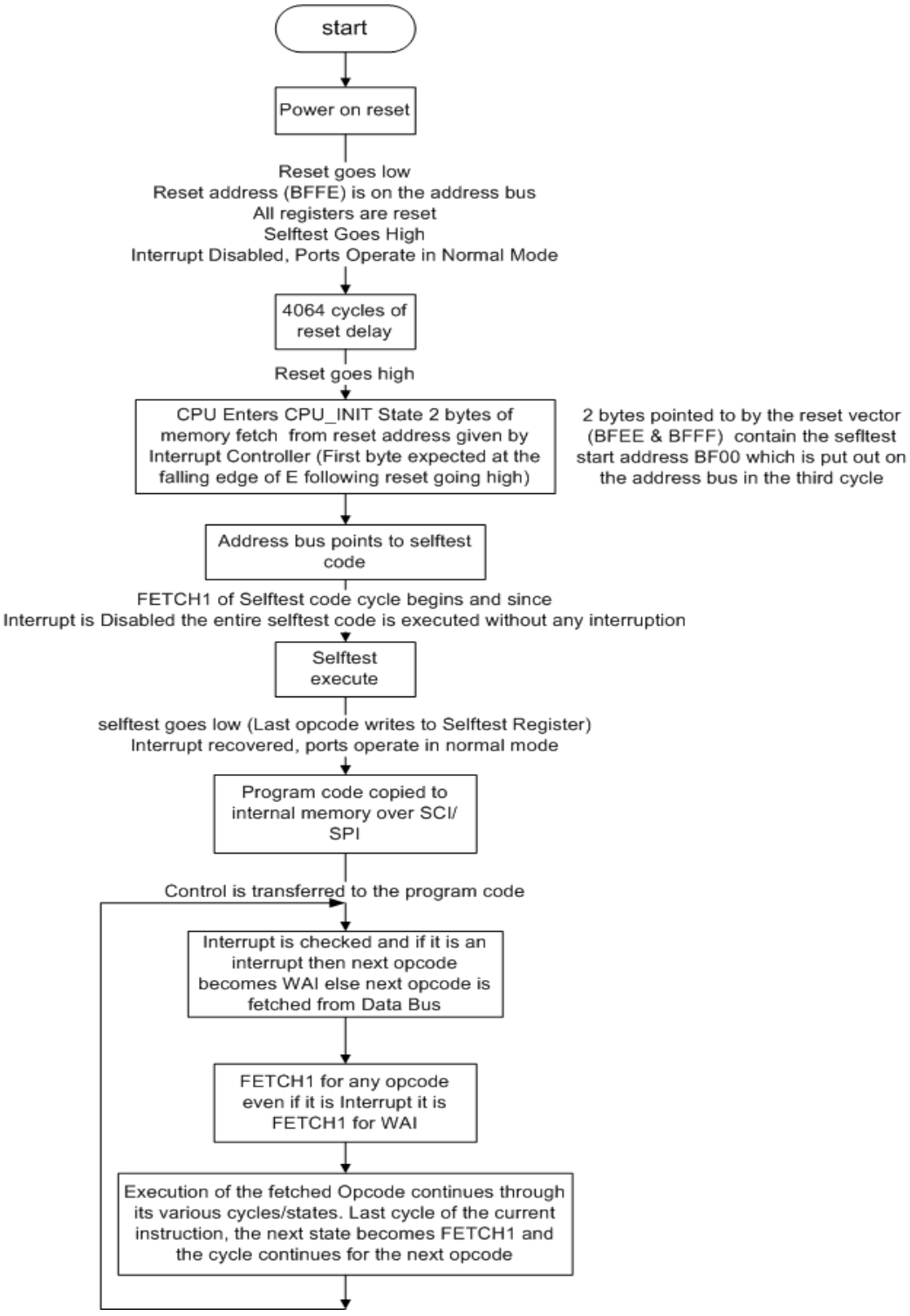


Figure 8 OSU 68HC11 test plan for wafer testing. The assembly code was written, and validate by testing on HC11 buffalo evaluation board.

4 Source Files

1. Module verilog: hctopscan.v, scanbus3.v, hctop.v, hcsyn.nodiv.v alu.v, multiplier.v

APPENDIX 3

PARALLEL INPUT/OUTPUT: PORT A, MAIN TIMER AND REAL-TIME INTERRUPT, PULSE ACCUMULATOR, RESET AND INTERRUPTS DOCUMENTS

TABLE OF CONTENTS

DOCUMENT
PAGE

1 System Description

The port A acts as real world interface for the timer system. It can also be used as an auxiliary general purpose input output port when not configured for timer related operation. The port A pins 0, 1 and 2 are input only pins, configured as input capture terminals and pins 3 through 6 are output only pins that can be used for output compare operations. The in-out Port A, pin 7 can be configured as either input capture or output compare interface based on the data direction bit in the port A control register.

A 16 bit counter forms the backbone of the timer unit. The counter's clock speed can be controlled by the pre scalar bits. Three 16 bit input capture units and five 16 bit output compare units are provided for real world timing related operations like pulse edge detection, PWM generation and delay. A standalone pulse accumulator unit is provided for counting input pulse either in normal mode or in gated clock mode.

The reset and interrupts are separately handled by different modules. The following operations cause a reset signal to be given to the CPU and other peripherals based on its cause.

1. Power on reset
2. External reset
3. Clock monitor fail reset
4. Watchdog timer reset

The power on reset is the initial reset provided to the entire system block when the microcontroller is powered up. The power on reset signal sets the register states to their corresponding predefined logic.

The external reset is given when the microcontroller gets / (want to give) a reset signal from / to the external chip that communicates with it.

The clock monitor fail reset is an optional reset which can be enabled by the clock monitor enable bit. The clock monitor fail circuitry is used to ensure that the system clock of the controller is above the minimum tolerance. In the current design the minimum clock speed is set to be around 50 KHz.

The watchdog timer in the timer module is a 8-bit counter which monitors the computers operations. Once enabled by clearing the NOCOP bit, this counter has to be periodically reset to avoid the reset signal being sent to the controller. The watchdog timer prevents the locking out of controller in a loop during execution.

The interrupt handler based on a prefixed priority encoder, resolves the interrupt requests from various modules and provides interrupt availability signal to the CPU. The priority and interrupt resolve flow charts are as consistent with 68HC11 manual.

The software interrupts (SWI), wait (WAI) and illegal opcode are directly handled by CPU and hence interrupt controller does not handle these exceptions.
The power on reset, clock monitor and clock generator modules are analog blocks that are manually laid out. The timer system, port A, interrupt handler and reset controller are all digital modules, placed and routed by encounter.

For more information on how to use the capture./compare functionalities of timer, interrupt priorities and flow, please refer to the M68HC11 reference manual from (Rev.6, 04/2002) Motorola (www.freescale.com)

2 Block diagram

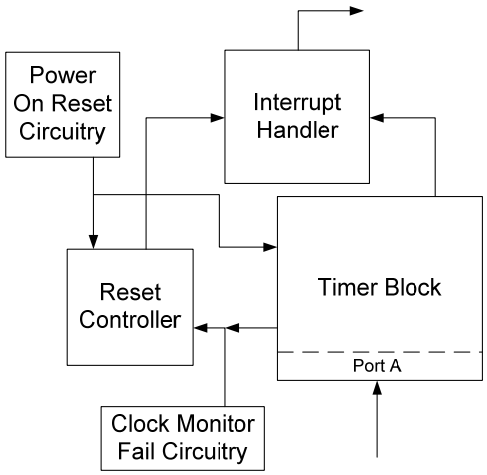


Figure 9. Functional block diagram power on reset, timer, interrupt handler, reset controller, clock monitor fail circuitries.

3 Schematic

Not applicable at high level

4 Pins Pad Out

Pads Out(External Connections)

Port	Width	Direction	Description
Pa7	1	Input/Output	Output when configured for compare (or) input when configured for pulse accumulator
Pa(3-6)	4	Output	Output compare controlled port A units
Pa(0-2)	3	Input	Input capture terminals of port A
intrst	1	Input	Internal reset to others blocks
reset	1	Input	External reset pin to reset block
xirq	1	Input	Active low XIRQ interrupt request pin
irq	1	Input	Active low IRQ interrupt request pin
xtal	1	Input	Oscillator input pin
extal	1	Output	Oscillator output pin to external peripheries
pwonrst	1	Input	Initial reset signal to HC11 blocks
	14		

CPU and Module Interface Connections

Port	Width	Direction	Description
DATA2CPU	8	Output	Data from timer to CPU
databus(7-0)	8	Input	Internal data communication channel between controller and timer
address(5-0)	6	Input	CPU signal for communicating address
iosel	1	Input	Control signal from CPU to identify the communicating block
rw	1	Input	CPU read/write control signal
ccr4	1	Input	CPU signal indicating I bit status
ccr6	1	Input	CPU signal indicating X bit status
rspc(1,2,14)	3	Output	Encoded reset address to CPU
rntr_addr(5-1)	5	Output	Encoded interrupt address to CPU
iaccept	1	Input	Status signal from CPU to acknowledge the acceptance of interrupt
Iavail	1	Input	Status bit to CPU for interrupt availability
Bootset	1	Output	Status of power on –boot up result
Slftst	1	Output	Status of power on self test result
ph2_clk	1	Output	Clock output to corresponding modules
ph1_clk	1	Output	Clock output to corresponding modules
e_clk	1	Output	Clock output to corresponding modules
pio	1	Input	Interrupt signal from port c
SPI_intr	1	Input	Interrupt signal from port D
SCI_intr	1	Input	Interrupt signal from port D
XIRQ_ctrl	1	Output	Status signal to CPU indicating a recovery by xirq interrupt
EI	1	Output	Clock stop information to CPU
	38		

5 Auxillary units

5.1 Power On Reset circuit

Releases an initial reset pin after the VDD pin reaches minimum operating voltage. This ensures the intended initial conditions on certain registers. This is an analog module, manually laid out. The circuit operates for VDD rise time lesser than or equal to 1mS.

5.1.1 Schematic

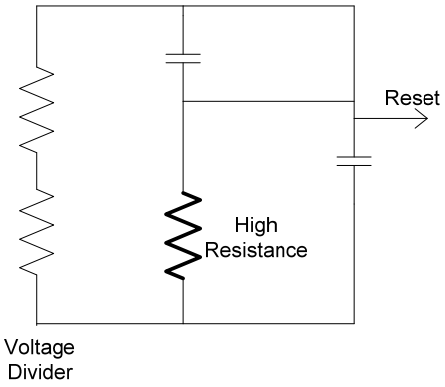


Figure 10. Functional schematic of Power on Reset circuit.

5.1.2 Layout

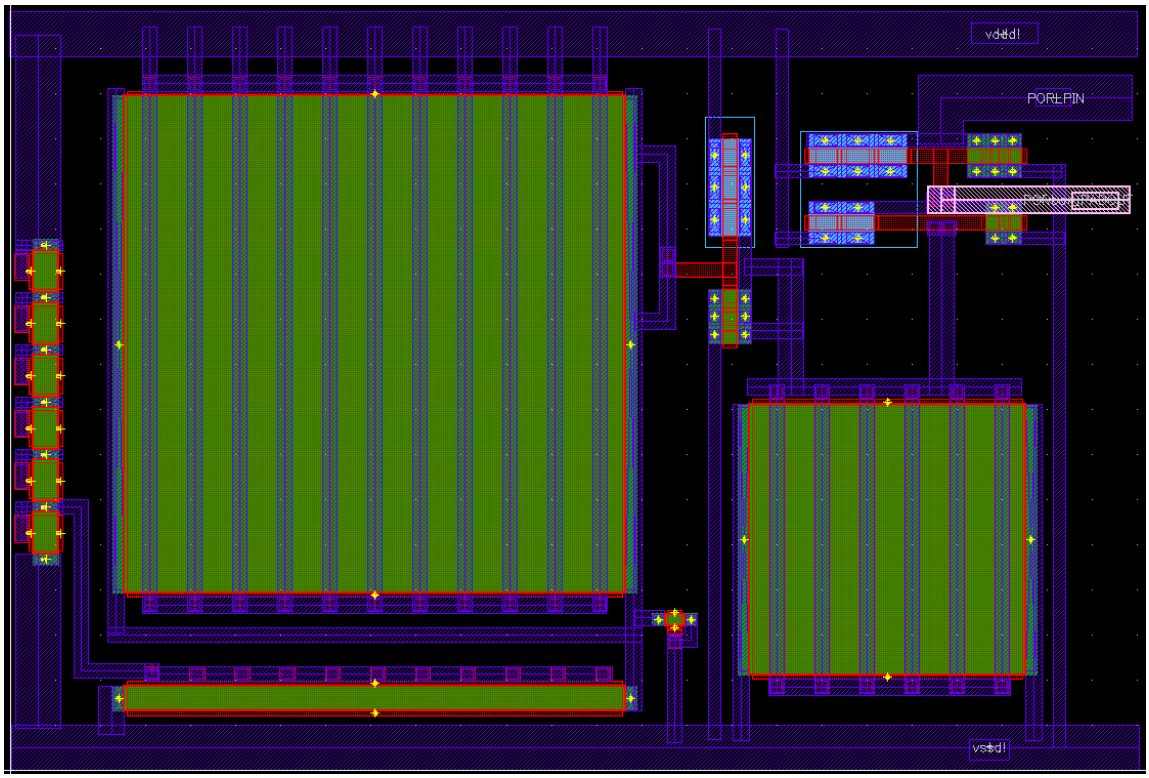


Figure 11. Layout (manual) of the power on reset circuit

5.2 Clock monitor fail circuit

Check the clocks rate and sends a fail signal when clock rate falls below 10KHz. This circuit is process sensitive and hence the failing point varies with corners. Enabling this circuit is from the clock monitor enable bit from timer.

5.2.1 Schematic

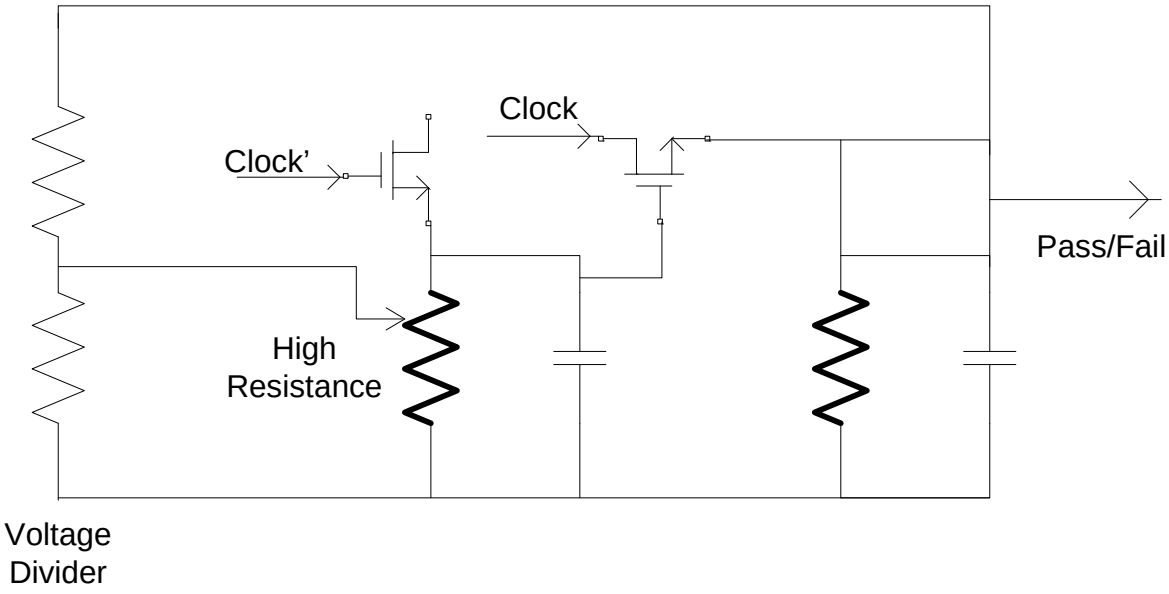


Figure 12. Functional schematic of Clock Monitor Fail circuit.

5.2.2 Layout

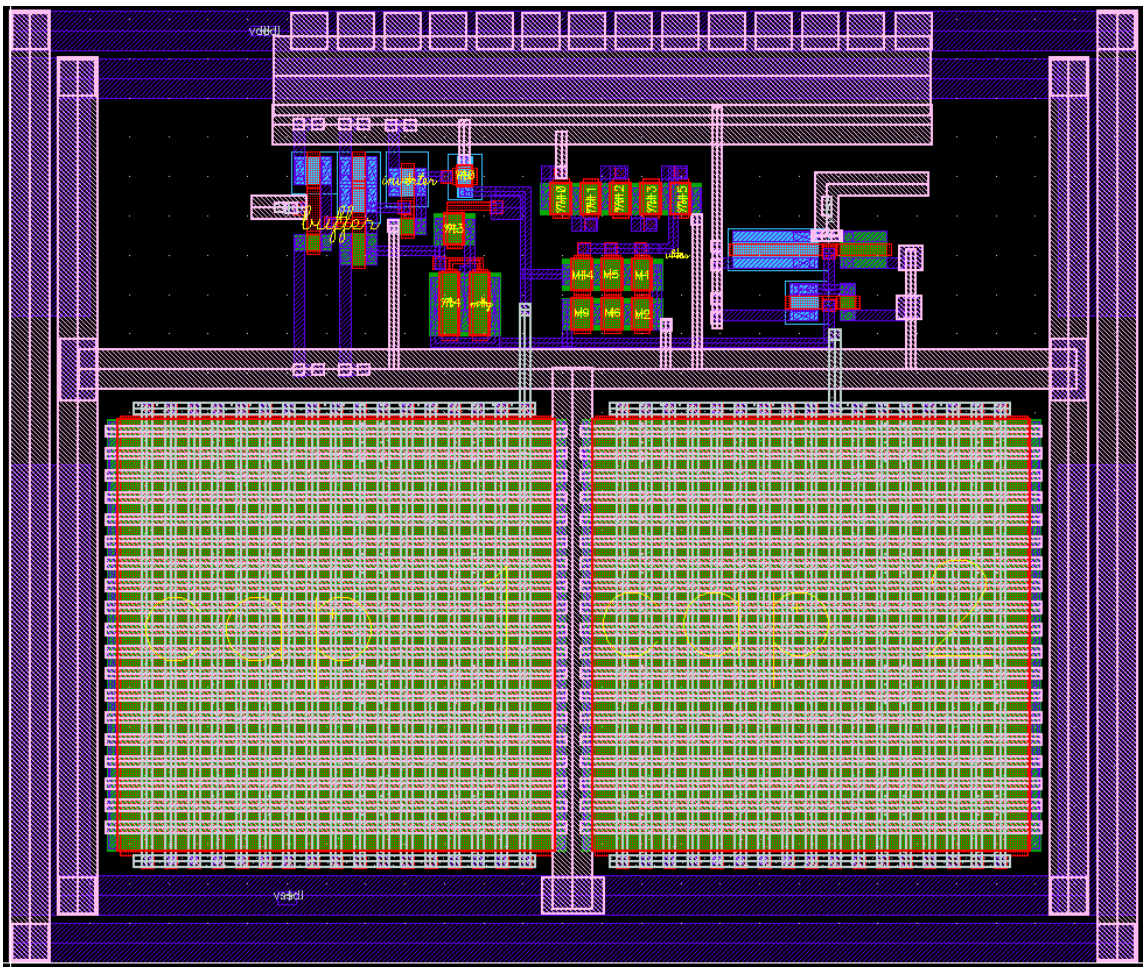
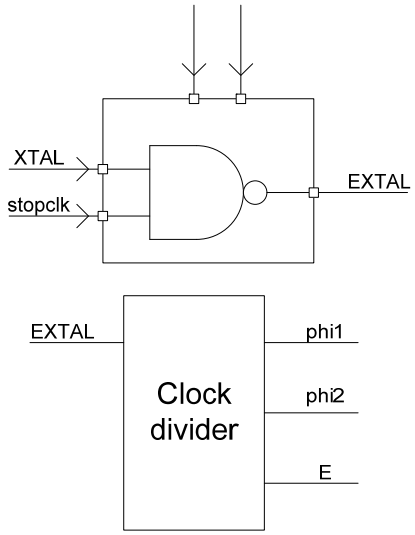


Figure 13. Layout (manual) of the clock monitor fail circuit.

5.3 Clock Generator Circuit

5.3.1 Schematic



Functional schematic of the clock generator circuit.

Figure 14. Functional schematic of Clock Generator circuit.

5.3.2 Layout

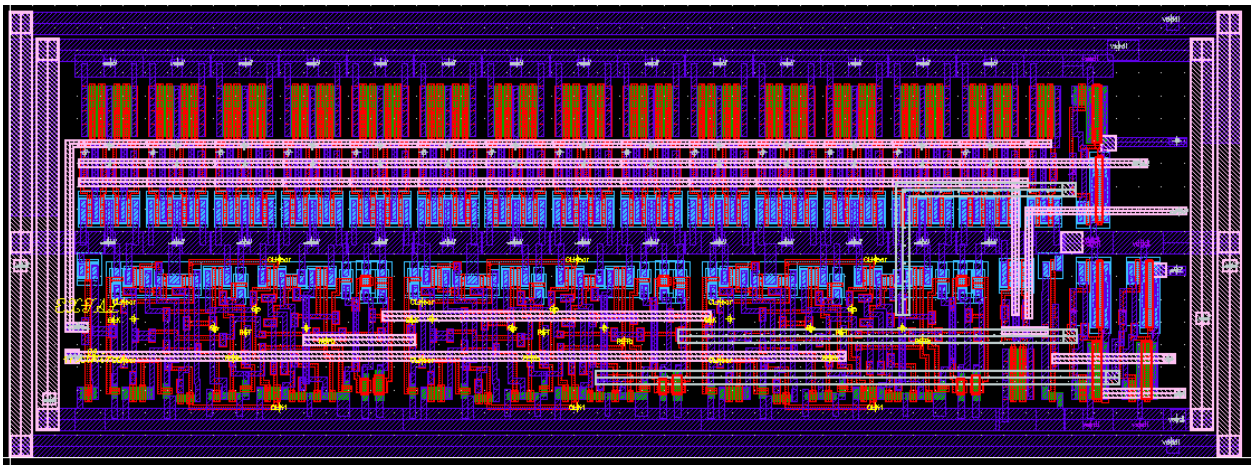


Figure 15. Layout of the clock generator circuit.

6 Testing Procedure

1.0 Timer

1.1 timer counter (16 bit)

1. inhibit count register update when reading high byte
2. timer registers (33)
3. 23 readable and writeable registers
4. 10 read only registers
5. 3 time critical writeable registers (write to some of the bits is valid only within 64 E clock cycles after reset)

1.2 input capture (16 bit, 3 modules)

1. configure TCTL2 register for positive, negative and/or either edge capture
2. inhibit updating capture register when reading high byte
3. interrupts shall be generated based on the TMSK1 register configuration.

1.3 output compare (16 bit, 4 modules)

1. inhibit compare when writing high byte of compare register
2. interrupts shall be generated based on the TMSK1 register configuration.
3. force compare corresponding to output compare force bit
4. no hardware generated interrupts for forced compares

1.4 pulse accumulator / output compare

1. verify working as pulse accumulator (input mode) or output compare (output mode) for the portA, based on PACTL register configuration.
2. verify the corresponding change in portA output based on TCTL1, OC1M and OC1D register

1.5 real time interrupt

1. interrupts requested at 4 different clock rates based on PACTL register configuration
2. configure TMSK2 register for hardware interrupts

1.6 computer operating properly (watchdog)

1. enabled based on CONFIG register setting.
2. interrupts requested at 4 different clock rates based on OPTION register configuration
3. write 55 and AA to COPRST register periodically to avoid watchdog to timeout.

2.0 Interrupts/Resets

- **2.1** change priority of interrupts through HPRIO register configuration
- **2.2** mask I interrupts based on CCR4 register
- **2.3** mask X interrupts based on CCR6 register

3.0 Clock monitor – Lower the clock speed to less than 50 KHz with and without setting the clock monitor enable bit and check for the corresponding clock monitor fail bit status.

4.0 Power on reset – Check for low signal at the diagnostic output from the power on reset unit to verify generation of reset signal at system power up.

Note:
Use open drain pull downs for RESET, IRQ and XIRQ pins.

7 Assembly Level Testing

7.1 Clock stopping and recovery testing:

Clock recovery conditional states:

1. CCR – D8; X & I Masked; XIRQ recovery – continues
2. CCR – D8; X & I Masked; IRQ recovery – continues
3. CCR – 98; I Masked; IRQ recovery – continues
4. CCR – 98; I Masked; XIRQ recovery – xirq interrupt
5. CCR – C8; X Masked; IRQ recovery – irq interrupt
6. CCR – C8; X Masked; XIRQ recovery – xirq interrupt
7. CCR – 88; No Mask; IRQ recovery – irq interrupt
8. CCR – 88; No Mask; XIRQ recovery – xirq interrupt

Assembly code:

```
    org $bf00
start
    LDAA #$ff
    STAA $1001
    ABA
    ldaa #$98
    TAP
    STOP
    ABA
    ABA
    NOP

    org $bffe
    fdb start
```

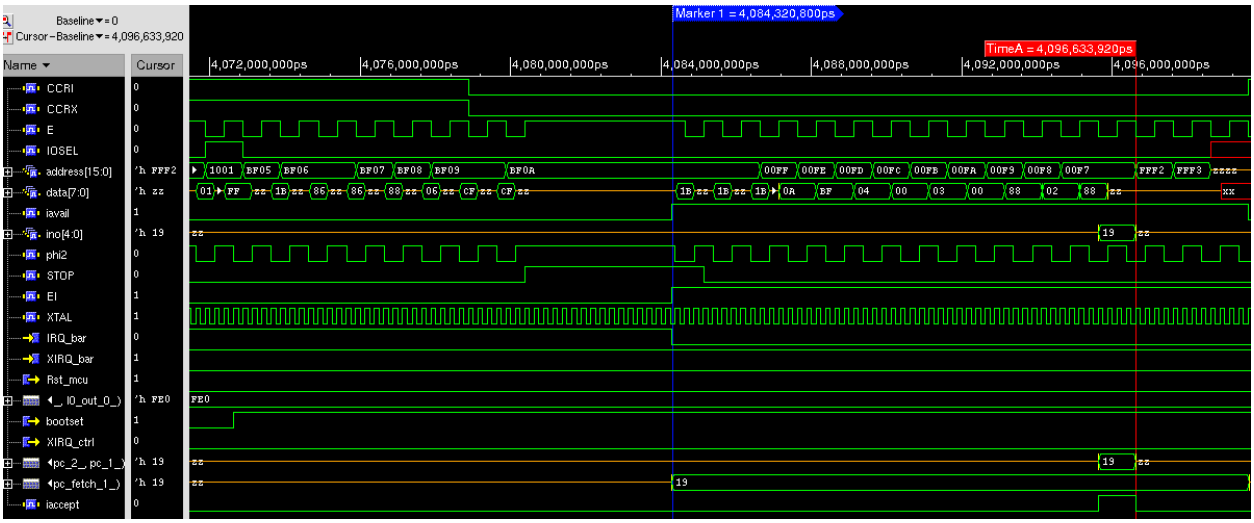


Figure 16. Clock recovery timing diagram

7.2 Capture/Compare testing:

Input capture:
Assembly code:

```
Bootset
   LDAA #$FF
   STAA $1001
Clear flags and intr. bits
   LDAA #$00
   STAA $1022
   STAA $1023
Set capture mode
   LDAA #$00
   STAA $1021
Give input
   NOP
   NOP
Check flag
   LDAA $1023
Give input
   NOP
   NOP
Check flag
   LDAA $1023
Continue process with
different capture mode
   LDAA #$15
   STAA $1021
   NOP
   NOP
   LDAA $1023
   NOP
   NOP
   LDAA $1023

   LDAA #$2A
   STAA $1021
   NOP
   NOP
   LDAA $1023
   NOP
   NOP
   LDAA $1023
   NOP
   NOP
   LDAA $1023
   NOP
Read capture reg.
   LDD $1010
```

Output Compare:
Assembly code:

```
   LDAA #$FF
   STAA $1001
   Clear Interrupt
   LDAA #$00
   STAA $1022
   Configure port
control
   LDAA #$55
   STAA $1020
Get current counter
   LDD  $100E
Set compare value
   ADDD #$001F
   STD  $1016
   STD  $1018
   STD  $101A
   STD  $101C
   STD  $101E
Wait till compare
   BRA *
END
```

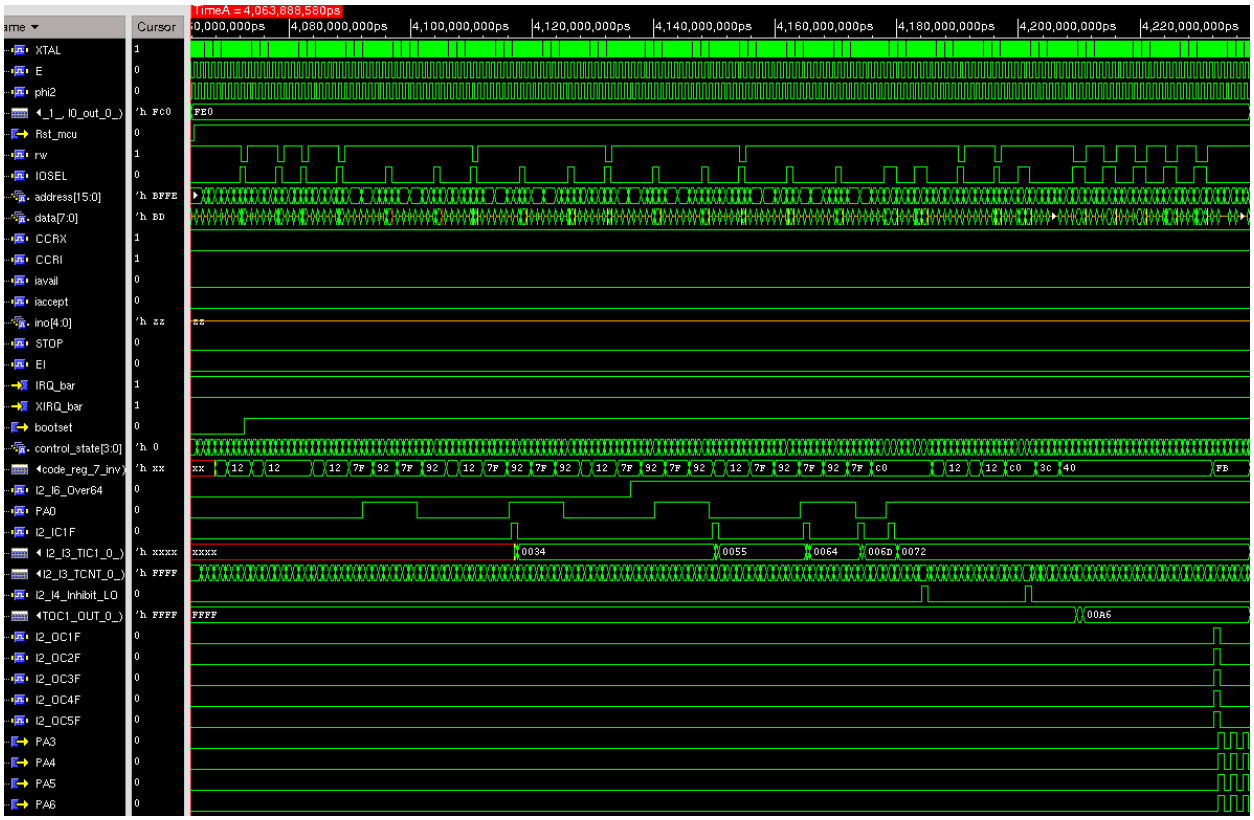


Figure 17. Power on reset, Input capture and Output compare timing diagram

8 File locations:

8.1 Timer/Interrupt

Msvlsi4 ./timepr/allinall/allinall
Encounter - allinall.enc
Library/Cell - not applicable.

8.2 Power on Reset

Msvlsi4 ./schtrig/
Library - schtrig
Cell - Initrst
Abstracttion
Library - porlib
Cell - Initrst

8.3 Clock Monitor

Msvlsi4 ./timer/
Library - timer

Cell - cmf
Abstracttion
Library - cmflib
Cell - cmf

8.4 Clock Generator

Msvlsi4 ./timer/
Library - timer
Cell - clkgen
Abstracttion
Library - clkgenlib
Cell - clkgen

APPENDIX 4

PARALLEL INPUT/OUTPUT: PORT B DOCUMENTS

TABLE OF CONTENTS

DOCUMENT **PAGE**

1 Description

Port B is implemented as eight output pins on the 68HC11. Port B operates in simple output mode. Extra functionality is added to Port B where they shared with other output pins during the diagnostics self test.

Port B pin logic is implemented based on Motorola M68HC11 reference manual. When the HNDS bit in the PIOC is zero, full handshake is disabled and Port B is used for simple strobe output. When performing write to Port B operation, the STRB signal is pulsed for 2 E clock cycles. Refer to Motorola M68HC11 reference manual for Port B registers in details. Additional debug pin, Post, self-test monitor pin, is added. The pin is initially low. *The pin output level high indicates the HC11 chip self-test is in progress. The pin output level low indicates the HC11 chip has existed from the self-test mode.* This Post pin is triggered by the CPU's internal self-test signal.

For more detail information on how to use the port B and implementation of port B, please refer to the M68HC11 reference manual from (Rev.6, 04/2002) Motorola (www.freescale.com).

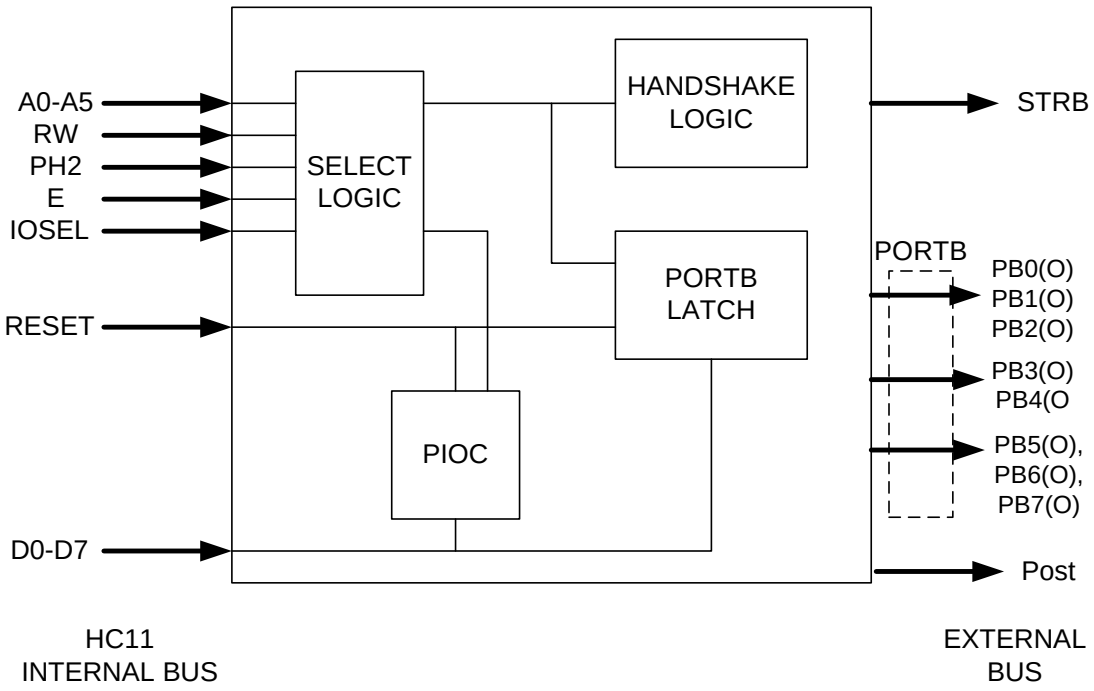


Figure 18 Block diagram of Port B system, showing Port B shared with diagnostics self test that the pins multiplexed Port B and register scan functionality.

Okl

2 Pins

Pads Out(External Connections)

Port	Width	Direction	Description
STRB	1	output	Handshake output
PB0	1	output	Port B bit 0 output
PB1	1	output	Port B bit 1 output
PB2	1	output	Port B bit 2 output
PB3	1	output	Port B bit 3 ouput
PB4	1	output	Port B bit 4 output
PB5	1	output	Port B bit 5 output
PB6	1	output	Port B bit 6 output
PB7	1	output	Port B bit 7 output
Post	1	output	Self-test monitor pin, the pin is initially low. High: indicates self-test in progress. Low: indicates exist from self-test mode.

CPU and Module Interface Connections

Port	Width	Direction	Description
datain	8	input	Data from the CPU to the module
PortBCdataout	8	output	Data from the module to CPU
addr (A0-A5)	6	input	Core/interface address
iosel	1	input	IO select
rst,	1	input	Module reset
e	1	input	E-clock input
ph2	1	input	Ph2-clock input
rw	1	input	read/write control signal

3 Testing and Simulation

Simulator: Xilinx, Verilog XL, and AMS Ultrasim

The functional test is conducted using Xilinx and Verilog XL. The timing and functional test with parasitic capacitance is tested on the AMS Ultrasim.

- Simulation is conducted to verify the Port B function: 1) general-purpose output and 2) simple strobe output. Both are in single-chip mode. The simulation setup and waveform are showed in figures below. The Port B function and timing (with parasitic) is fully simulated before integrated with CPU core.
- The code is compiled and burned in to Xilinx board to verify the code is synthesizable and implementable.

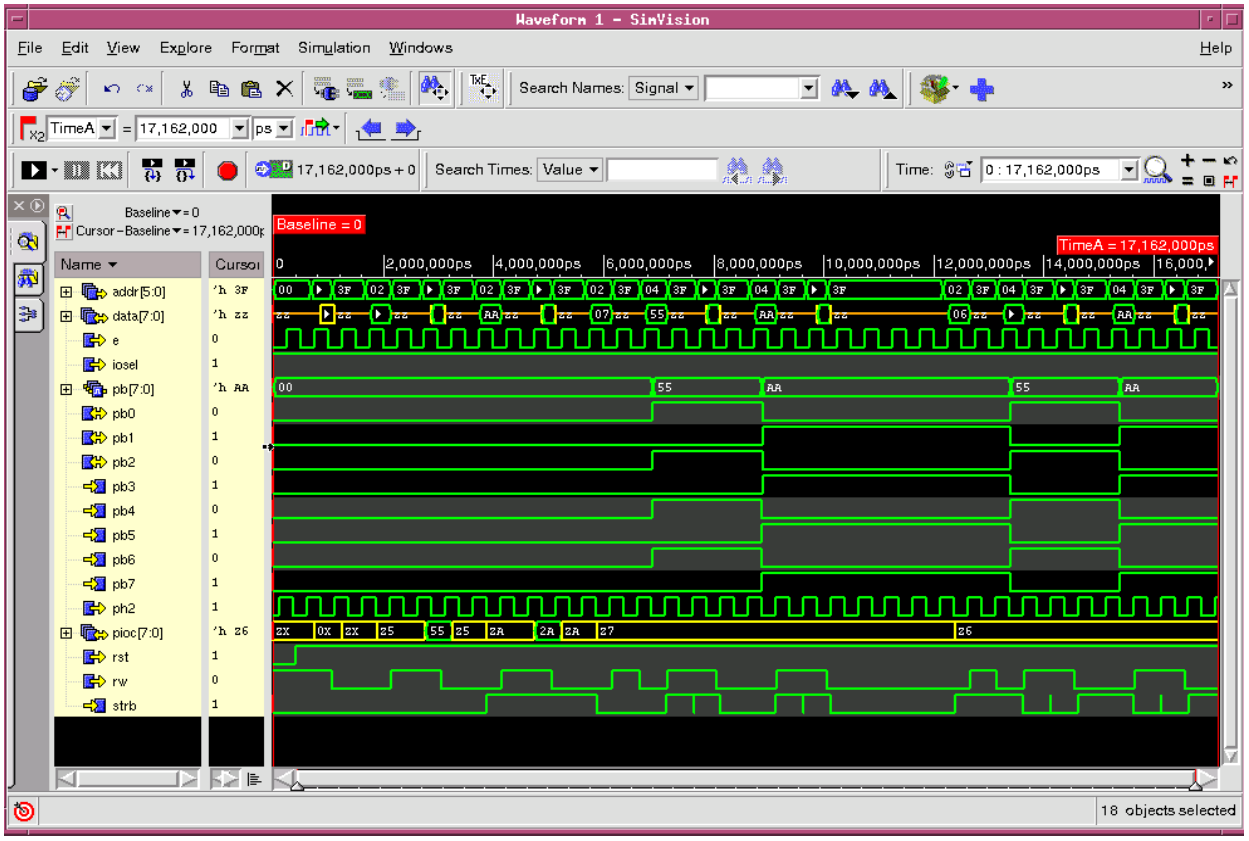


Figure 19 Simulation waveform of Port B. The test data 0x55 and 0xAA can be shifted out from Port B correctly

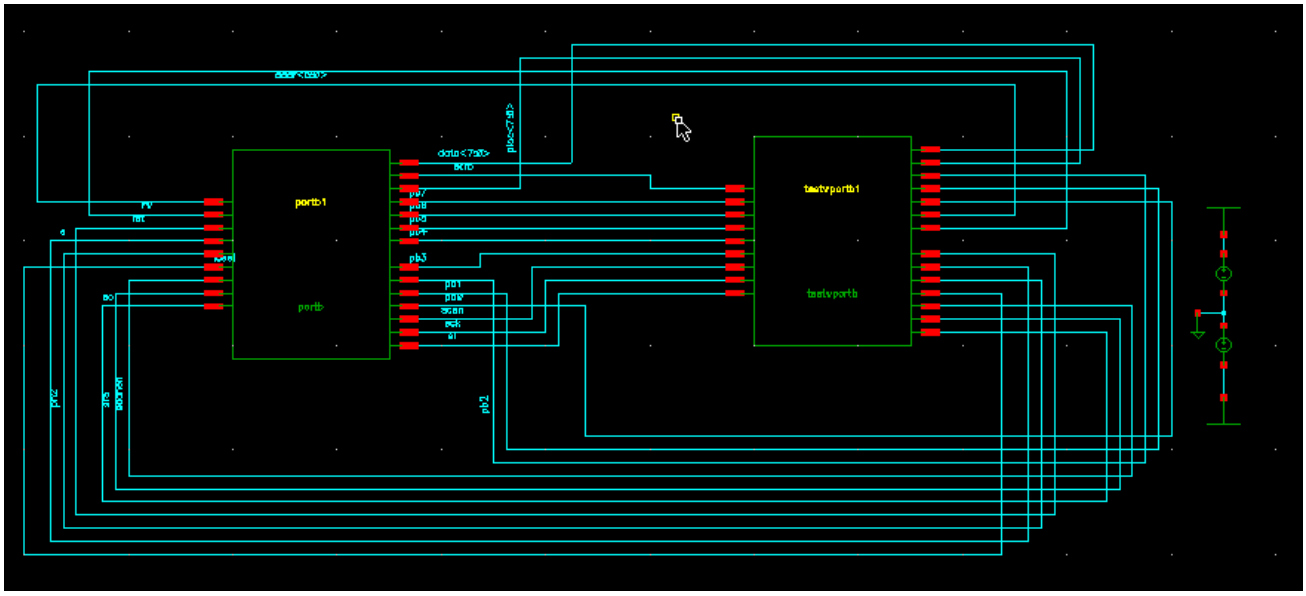


Figure 20 AMS Ultrasim simulation setup: test-bench supplies test vector to Port B.

4 Source Files

2. Module verilog: portb121707.v, twopls.v, combinepbc-new.v

APPENDIX 5

PARALLEL INPUT/OUTPUT: PORT C DOCUMENTS

TABLE OF CONTENTS

DOCUMENT **PAGE**

1 Description

Port C is implemented as eight input/output pins on the OSU 68HC11. Port C can either be in the simple input mode, full input mode and full output mode, when the HC11 is on single chip mode. Port C expanded mode is not being implemented.

Port C pin logic is implemented based on Motorola M68HC11 reference manual. Port C can be used for simple latching in the input mode, full input handshake mode, full output handshake mode (normal output handshake, 3-State variation of output handshake), or 8 general purpose input/output pins. Refer to Motorola M68HC11 reference manual for Port C registers in details.

As general I/O, each pin is bidirectional, and is configured individually by writing to the data direction register for Port C (DDRC) register, where 0 indicates input while 1 indicates output. Once the direction of the port is set up, the PORTC register can be easily read or write.

Port C can also be used for simple latching input mode, full input handshake mode, full output handshake in combination with the strobe A (STRA) and strobe B (STRB) signals. When the HNDS bit in the PIOC is zero, full handshake is disabled and Port C is used for simple latching input.

In the simple or full input handshake mode, data is latched into PORTCL when the STRA signal became asserted. In the simple strobe input mode, as the external peripheral put data in the port C pins that configures ad input, the peripheral will toggle STRA and the data will be latched on the PORTCL register of HC11. STRA signal polarity is set by the EGA bit in the PIOC register (0 for falling edge and 1 for rising edge). The STAF bit in the PIOC register is asserted as soon as the STRA active edge is sensed if interrupt of this source is enabled (STAI bit in the PIOC register is set and the I mask in the CCR is clear). To clear the STAF flag, the user must first read the PIOC and then read the PORTCL register. Figure 2 shows the full input handshake mode, the data is input to the port C, and the peripheral pulses the STRA signal, this will set the STAF

bit in the PIOC register. When STA flag is asserted, the HC11 code will want to read data from the PORTCL. The HC11 de-asserts the STRB signal informing the peripheral it is busy (notice STRB comes back to the peripheral). As soon as the STAF is cleared, HC11 asserts the STRB signal and to inform the peripheral that HC11 is ready to receive new data.

In the full output handshake mode, data is written to PORTCL outputs through port C pins, where the port C pins all force to the output mode when STRA is at its active level. Figure 3 shows the full output handshake mode operation, where the new data is inhibited from latching into PORTCL until the previous data is read from PORTCL where STRB is asserted. STRA is used for the peripheral to indicate when it is ready while STRB signal is used by the HC11 to inform the peripheral that the new data has been put on the output port. HC11 puts a byte at the port C (by latching data in to PORTCL register) and asserts the STRB signal. The external peripheral will take some time receive the data, and it will assert the STRA to indicate that the data has been received. As the STRA signal is asserted the STAF bit in the PIOC register will be set. STAF remains asserted until the code clears it, first by reading the PIOC and then writing data on the PORTCL register. As soon as the STAF is cleared, the HC11 can put new data and assert the STRB signal.

For more information on how to use port C, please refer to the M68HC11 reference manual from (Rev.6, 04/2002) Motorola (www.freescale.com).

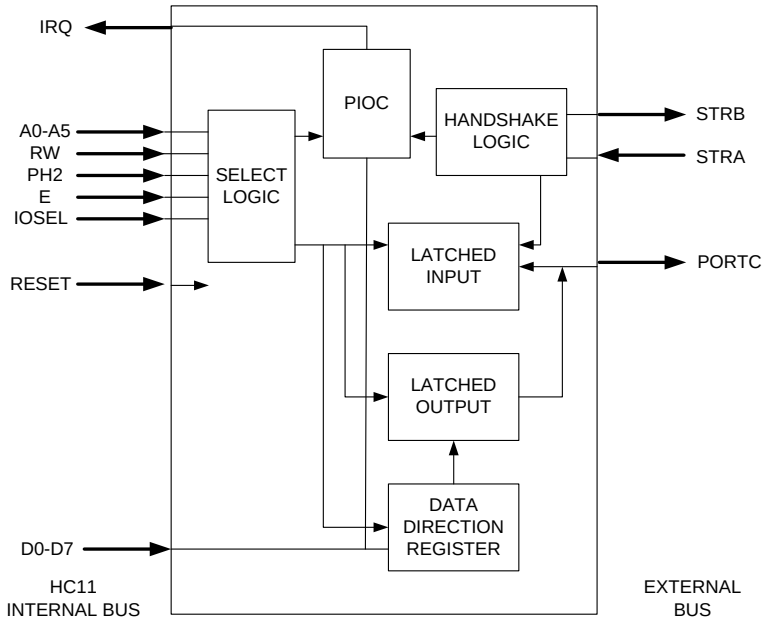


Figure 21Block diagram of Port C system, showing Port C shared with diagnostics self test that the pins multiplexed Port B and register scan functionality.

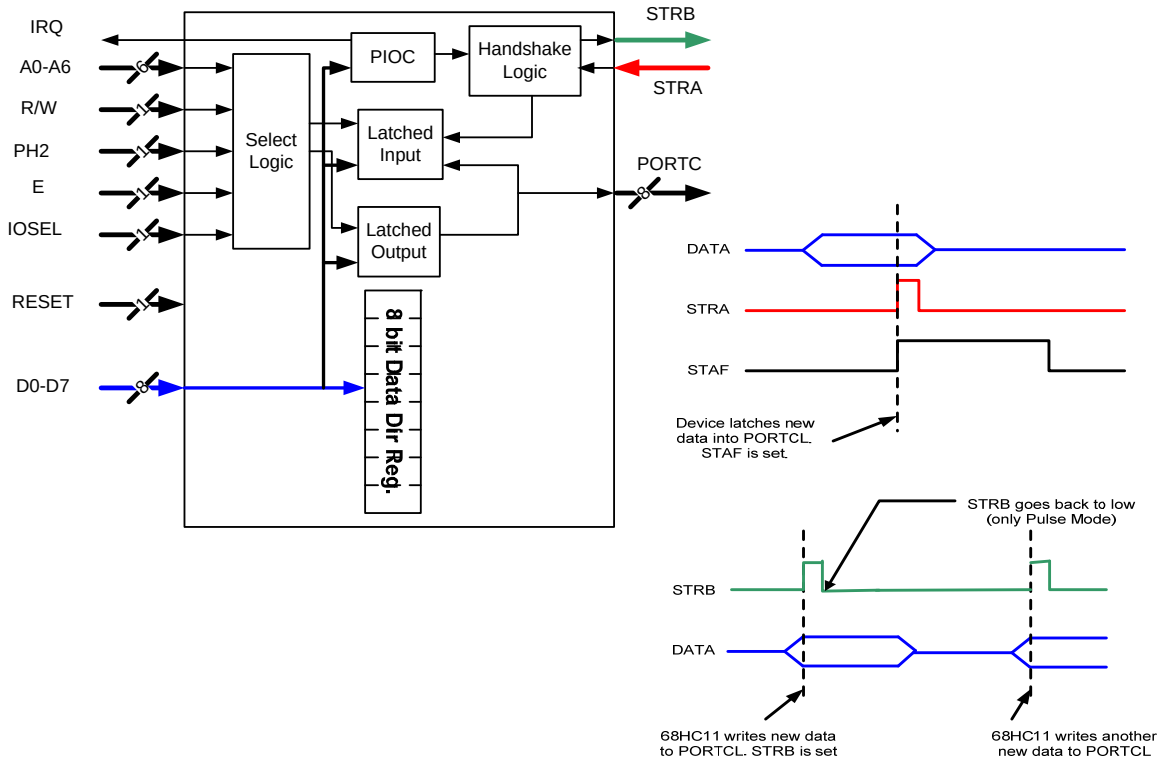


Figure 22 Port C simple input handshake mode operation.

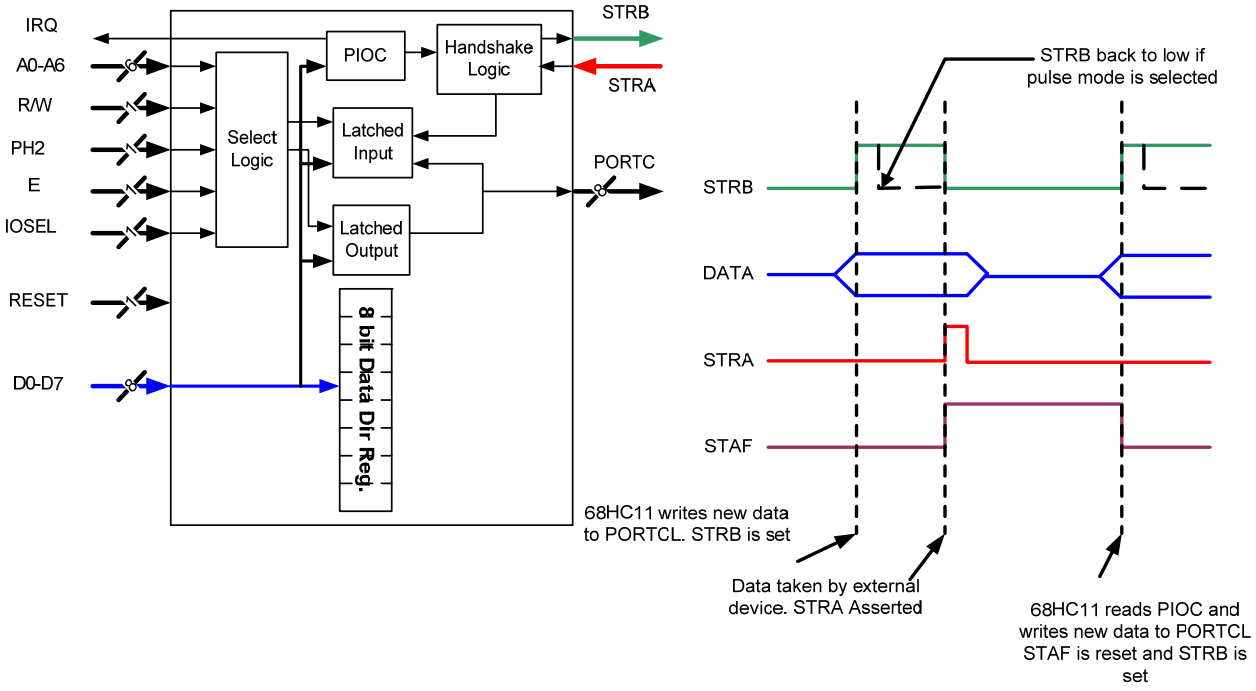


Figure 23 Port C output handshake mode operation.

2 Pins

Pads Out(External Connections)

Port	Width	Direction	Description
STRB	1	output	Handshake output
STRA	1	input	Handshake input
PC0	1	input/output	Port C bit 0 input/output
PC1	1	input/output	Port C bit 1 input/output
PC2	1	input/output	Port C bit 2 input/output
PC3	1	input/output	Port C bit 3 input/output
PC4	1	input/output	Port C bit 4 input/output
PC5	1	input/output	Port C bit 5 input/output
PC6	1	input/output	Port C bit 6 input/output
PC7	1	input/output	Port C bit 7 input/output

CPU and Module Interface Connections

Port	Width	Direction	Description
datain	8	input	Data from the CPU to the module
PortBCdataout	8	output	Data from the module to CPU
addr (A0-A5)	6	input	Core/interface address
iosel	1	input	IO select
rst,	1	input	Module reset
e	1	input	E-clock input
ph2	1	input	Ph2-clock input
rw	1	input	read/write control signal

3 Testing and Simulation

Simulator: Xilinx, Verilog XL, and AMS Ultrasim

The functional test is conducted using Xilinx and Verilog XL. The timing and functional test with parasitic capacitance is conducted on the AMS Ultrasim environment.

- Simulation is conducted to verify the Port C function in single-chip mode: 1) simple latching input mode, 2) full input handshake mode, 3)full output handshake in combination with the strobe A (STRA) and strobe B (STRB) signals. The simulation setup and waveform are showed in figures below. The Port C function and timing (with parasitic) is fully simulated before integrated with CPU core.
- The Port C verilog code is synthesized, compiled and burned in to Xilinx board to verify the code is synthesizable and implementable.

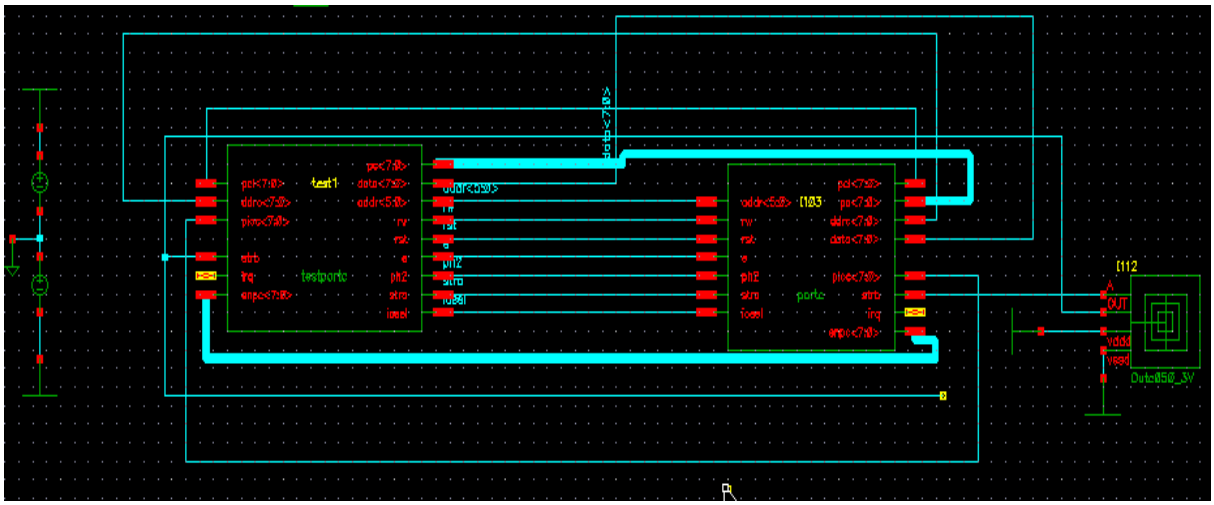


Figure 24 Port C's simulation setup.

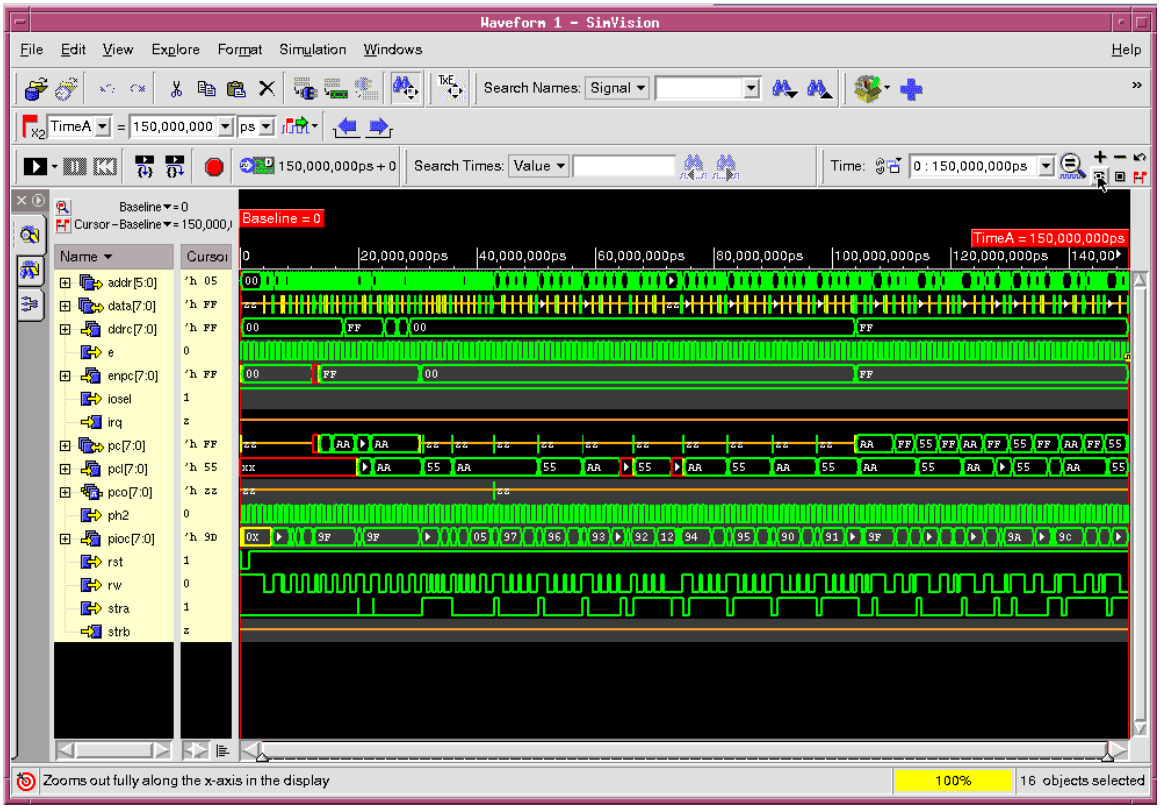


Figure 25 Port C testing and simulation output waveform.

4 Source Files

3. Module verilog: portc101007.v, twoplsc0203.v, straact0203.v, plszero.v

APPENDIX 6

PARALLEL INPUT/OUTPUT: PORT D, Synchronous Serial Peripheral Interface (SPI), and Asynchronous Serial Communications Interface (SCI)

DOCUMENTS

TABLE OF CONTENTS

DOCUMENT **PAGE**

1 Description

Port D is implemented as 6-bit bidirectional data port. Port D can be used as 6-bit Input/Output (PD0-PD5) general purpose I/O port, or two pins serve as asynchronous serial communication interface (SCI) system, and the other four pins serve as serial peripheral interface (SPI) system. See Figure 1.

1.1 Port D

Port D pins logic is implemented based on Motorola M68HC11 reference manual. Each Port D pin is bidirectional, and their direction can be set individually by writing to the data direction register for Port D (DDRD) register of the corresponding bit position (e.g., bit 0 is Port D's pin 0), where 0 for input and 1 for output. Once direction is configured, user can read or write to the PORTD register. Port D pin 0 and pin 1 are multiplexed with SCI receive and transmit function respectively. If SPI system is enabled, pin 2 to pin 5 are no longer general purpose I/O pins.

For more information on how to use Port D and its registers in details, please refer to the M68HC11 reference manual from (Rev.6, 04/2002) Motorola (www.freescale.com).

1.2 SPI Master/Slave

SPI Master/Slave General Features:

- Master or slave mode operations
- Mode fault error
- Write collision error
- Interrupt generation
- Bit rates generated 1/2, 1/4, 1/ 16, 1/32 of processor/system clock (E clock). Note: Slave does not generate SCK clock, only master provides SCK clock, and slave receives the SCK clock from master.
- Full duplex, synchronous, 8-bit serial data transfer. During an SPI transfer, data is simultaneously transmitted (shifted out serially) and received (shifted in serially) (Figure 3).

- Four transfer formats supported four modes, the combinations of the clock polarity (CPOL) and the clock phase (CPHA) (Figure 2):

CPOL	CPHA
0	0
0	1
1	0
1	1

When CPHA = 0, data is latched at the rising edge of the clock with CPOL = 0, and at the falling edge of the clock with CPOL = 1. If CPHA = 1, the polarities are reversed. CPOL = 0 means falling edge, CPOL = 1 rising edge.

During transmission, there is only one SPI master unit and all other units must be configures as slaves. All SPI units on the network must have the same type of clock polarity, phase and clock rate in order to communicate. All these configuration bits can be set in the serial peripheral control register (SPCR). The output SPI pin has to be configured in the DDRD register. The SPI master unit will write a byte of data to serial peripheral data register (SPDR), and flags in serial peripheral status register (SPSR) register will asserted when the transfer is complete and if any error occurred. The data arrives at the slave SPDR register, where the data is passed through a clocked shift register. Note that if a transfer is on going, a new byte written to the SPDR will overwrite the shift register. Data is simultaneously transmitted (shifted out serially) and received (shifted in serially) (Figure 3) at the SPDR whether is at the master side or slave side.

Refer to Motorola M68HC11 reference manual for SPI registers in details.

1.3 SCI Transmitter/Receiver

SCI Transmitter/Receiver Features:

- Full duplex, asynchronous, serial data transfer.
- 8 or 9 bit data transfer
- Integrated BAUD Rate generator, 32 different baud rate frequency.
- Enhanced receiver data sampling technique

- IDLE and BREAK characters generation
- Wake-up block
- SCI related interrupts
- Full-duplex UART-type asynchronous system, using standard non-return-to-zero (NRZ) format (one start bit, eight or nine data bits, and a stop bit).
- Baud rate generator derives standard baud-rate frequencies from the MCU oscillator (÷1, ÷2, ÷4, ..., ÷128).
- Both the transmitter and the receiver are double buffered; thus, back-to-back characters can be handled easily (Figure 4).
- SCI receiver's advanced features: ensure high-reliability data reception, and to assist development of efficient communications networks.
- Three logic-level samples are taken near the middle of each bit time, and majority logic decides the sense for the bit. Even if noise causes one of these samples to be incorrect, the bit will still be received correctly.
- Receiver wakeup mode: the receiver also has the ability to enter a temporary standby mode (called receiver wakeup).
- The SCI transmitter can produce queued characters of idle (whole characters of all logic 1) and break (whole characters of all logic 0).
- Transmit data register empty (TDRE) status flag, and a transmit complete (TC) indication (that can be used in applications with a modem).

The SCI port can be controlled through the serial communications control register 1 (SCCR1), serial communications control register 1 (SCCR2), serial communications status register (SCSR), serial communications data register (SCDR) and baud register (BAUD). Two signal lines: TXD (transmit) and RXD (receive) are used in SCI transmission. It uses either 8-bit or 9-bit data format, and data is sent as full-duplex UART-type asynchronous system, using NRZ format. There is one start bit and one stop bit. It ensures high-reliability data reception with advanced error detection. Both the transmitter and the receiver are double buffered. SCI also offers the features of sleep mode (idle), wake up mode and interrupt mode. Compare to SPI, SCI writing to the data register is more protected because there is an intermediate buffer between the data

register and the shift register. If a byte is written to the data register and the intermediate buffer is not emptied, the data in the intermediate buffer is lost but the shift register is unaffected (Figure 4).

Refer to Motorola M68HC11 reference manual for SCI registers in details.

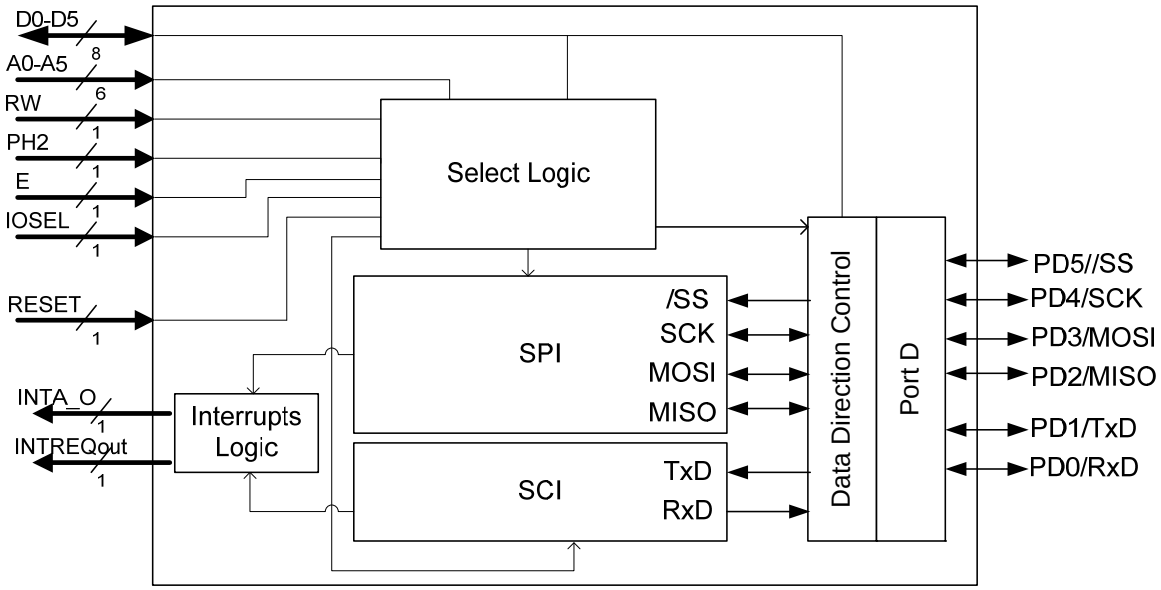


Figure 26 Block diagram of Port D system, showing Port D shared with SPI and SCI systems.

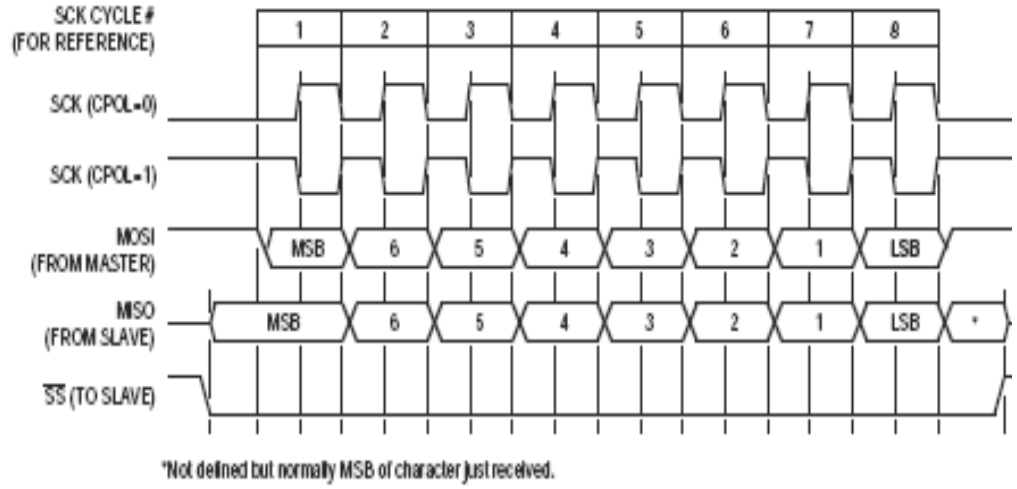


Figure 27 CPHA Equals Zero SPI Transfer Format (from M68HC11 reference manual Fig. 8-1).

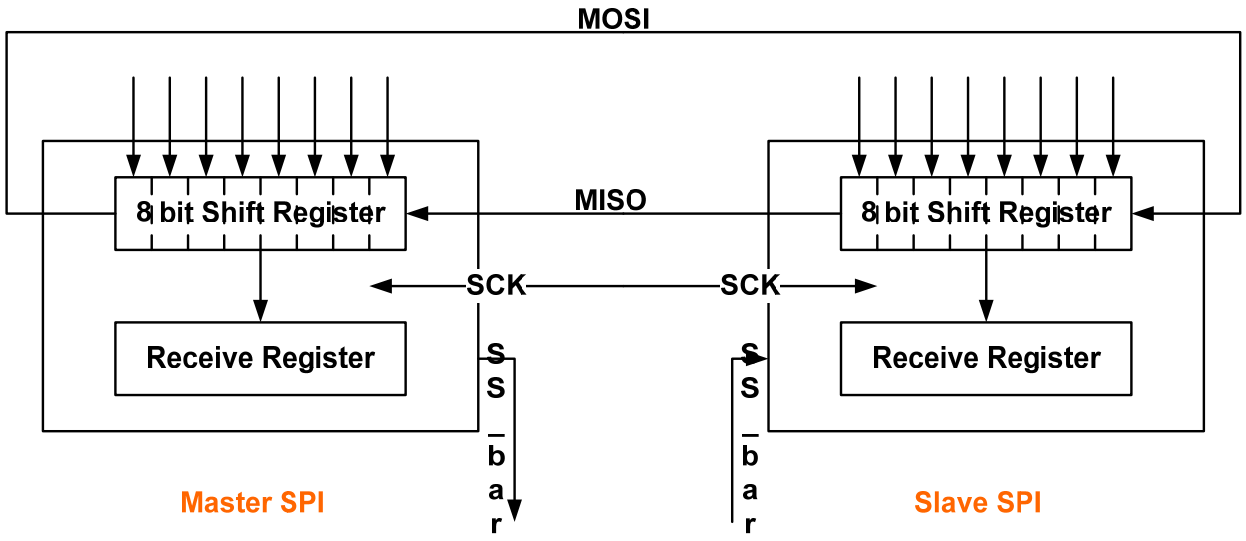


Figure 28 SPI shift register used in data transmission.

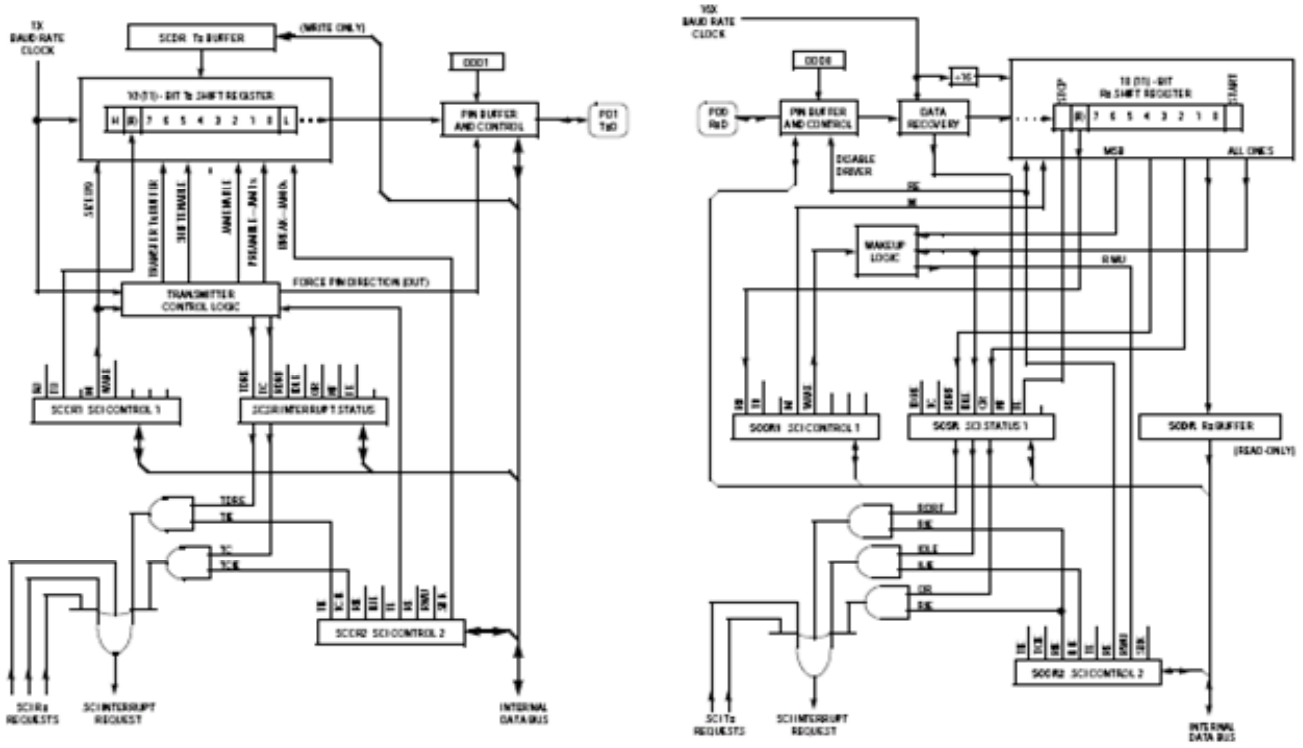


Figure 29 SCI transmit and receive logic.

For SPI system, there is a 5V port that is paralleled off the 3.3V function (Figure 5). The dedicated 5V SPI pins has the SPI function same as the Port D 3.3V SPI pins.

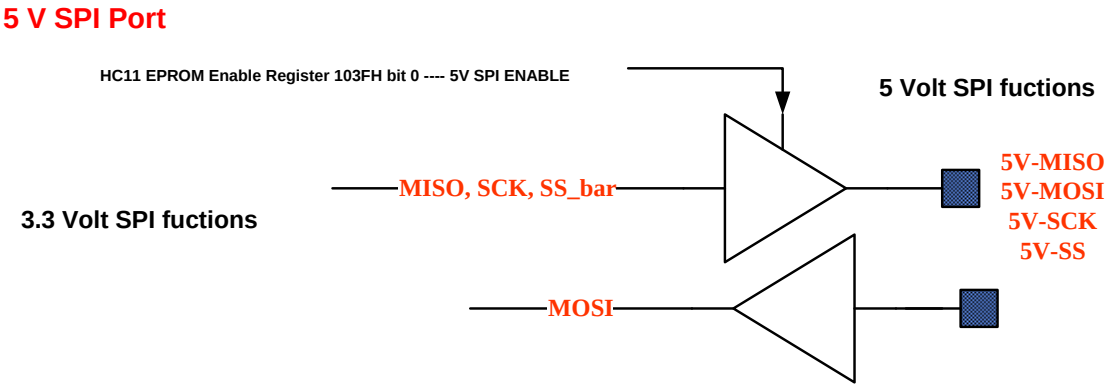


Figure 30 SPI 5V shadowed port logic.

2 Pins

Pads Out(External Connections)

Port	Width	Direction	Description
PD[0]/rxd	1	input/output	port input/output, and SCI receive pin (input)
PD[1]/txd	1	input/output	port input/output, and SCI transmit pin (output)
PD[2]/miso	1	input/output	port input/output, and SPI Master In Slave Out (MISO)
PD[3]/mosi	1	input/output	port input/output, and SPI Master Out Slave In (MOSI)
PD[4]/sck	1	input/output	port input/output, and SPI output clock sck (master) or input clock sck (slave)
PD[5]/ssn	1	input/output	port input/output, and SPI slave enable active low

CPU and Module Interface Connections

Port	Width	Direction	Description
PortDdataout	8	output	Data from the module to the CPU
dat_i(D0-D7)	8	input	Data from CPU to the module
addr (A0-A5)	6	input	Core/interface address
iosel	1	input	IO select
rst (reset)	1	input	Module reset
e	1	input	E-clock input
ph2	1	input	Ph2-clock input
rw	1	input	read/write control signal
inta_o	1	output	SPI tx/rx done interrupt flag
intreqout	1	output	SCI rx done interrupt flag

3 Testing and Simulation

Simulator: Xilinx, Verilog XL, and AMS Ultrasim

The functional test is conducted using Xilinx and Verilog XL. The timing and functional test with parasitic capacitance is conducted on the AMS Ultrasim environment.

- Simulation is conducted to verify the Port D functions: 1) general input/output port, 2) SCI communication signaling and its features, 3) SPI communication signaling and its features. The simulation setup and waveform are showed in figures below. The Port D function and timing (with parasitic) is fully simulated before integrated with CPU core.
- The Port D verilog code is synthesized, compiled and burned in to Xilinx board to verify the code is synthesizable and implementable.

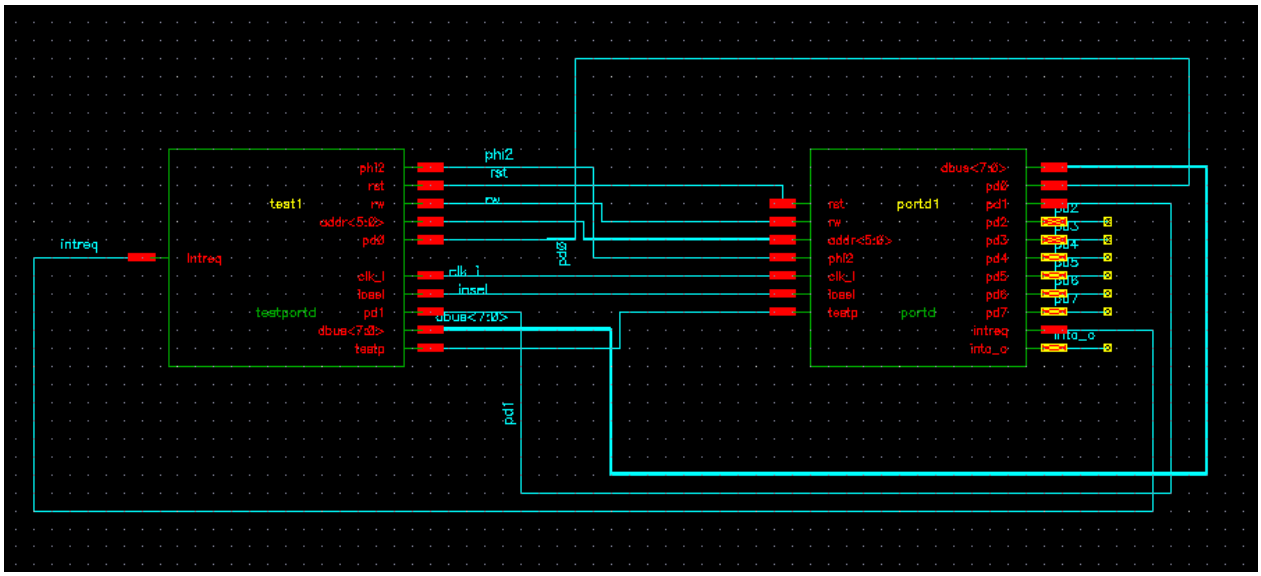


Figure 31 Port D and SCI features simulation setup.

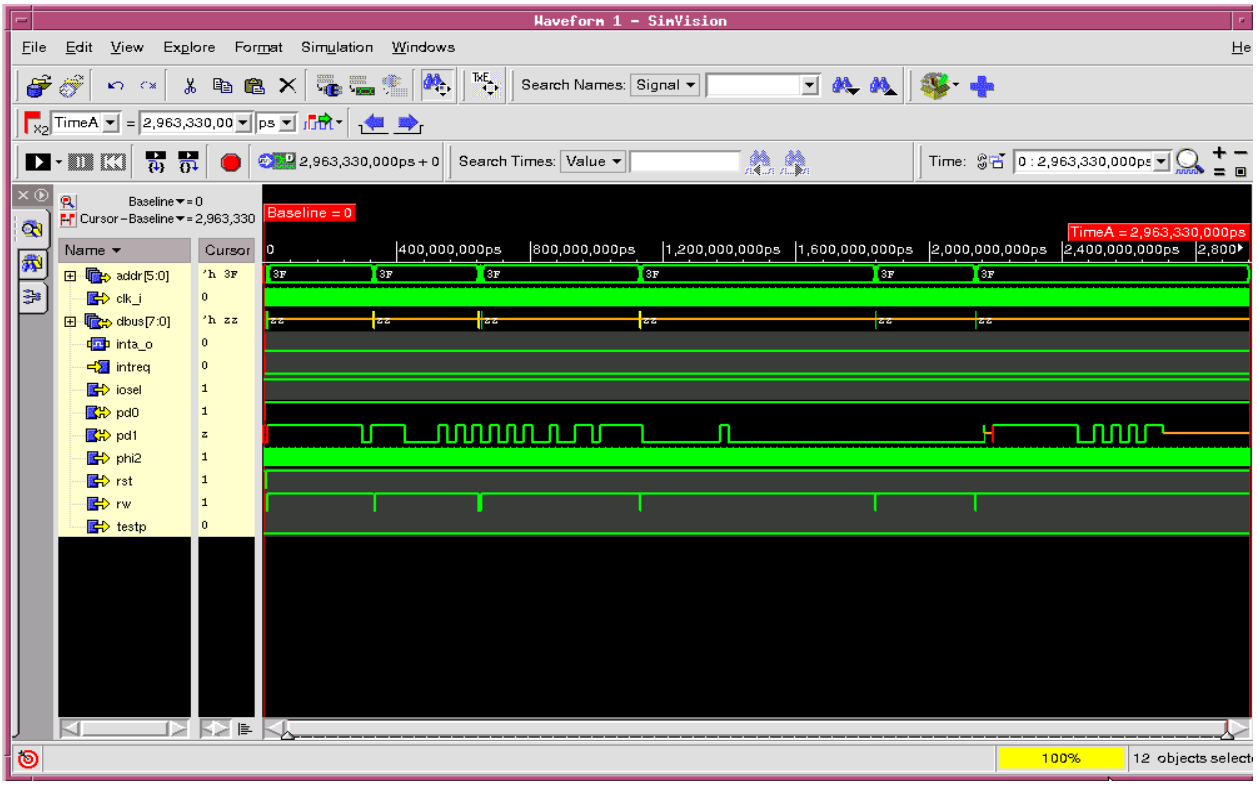


Figure 32 SCI Ultrasim simulation.

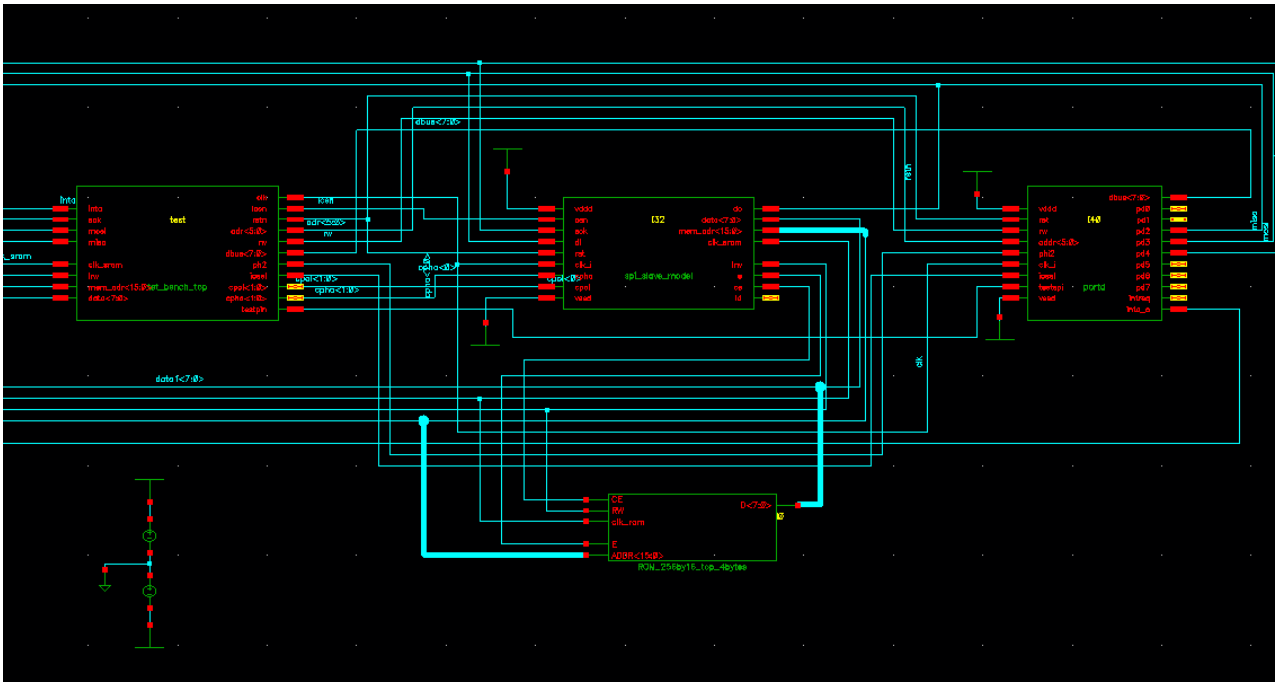


Figure 33 SPI features simulation setup. The port D SPI is configured as a SPI master, and communicates (write to/read read) with SPI-SRAM (SPI slave memory device). The test bench (test_bench_top) acts as CPU to supplies the test vectors to control Port D.

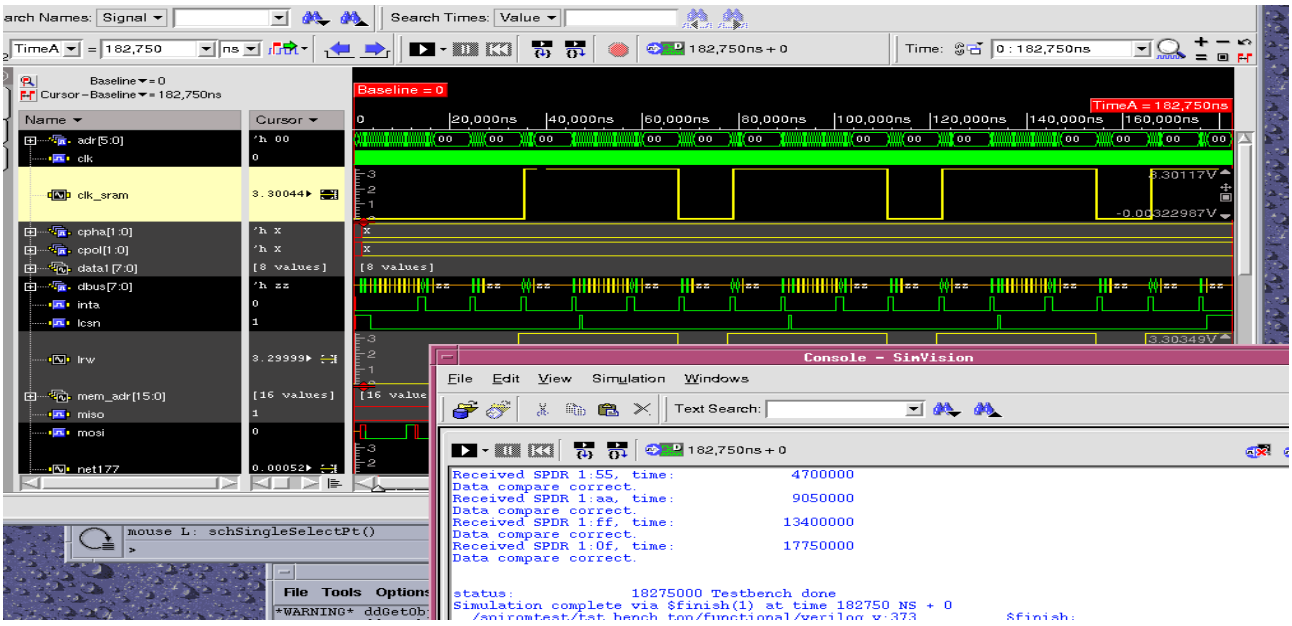


Figure 34 SPI Ultrasim simulation output window and waveform. It showed that the Port D's SPI port able to communicate with SPI slave via the SPI interface.

4 Source Files

4. Module verilog: portd101407pnr.v, spi101407.v, fifo1.v, sci101407.v

APPENDIX 7

HC11 On-CHIP ROM DOCUMENTS

TABLE OF CONTENTS

1 HC11 On-Chip ROM

HC11 on-chip ROM has the size of 512 bytes. HC11 microcontroller loads the self test and bootstrap from the on-chip ROM. The ROM cell consists of one PMOS transistor. A Divided word Line Architecture has been implemented in this ROM. It consists of ROM cells, an 8 to 256 Row Decoder, row logic, write circuitry and tristate buffers .The connection to vdd or vss of the ROM cell is programmed using SKILL language.

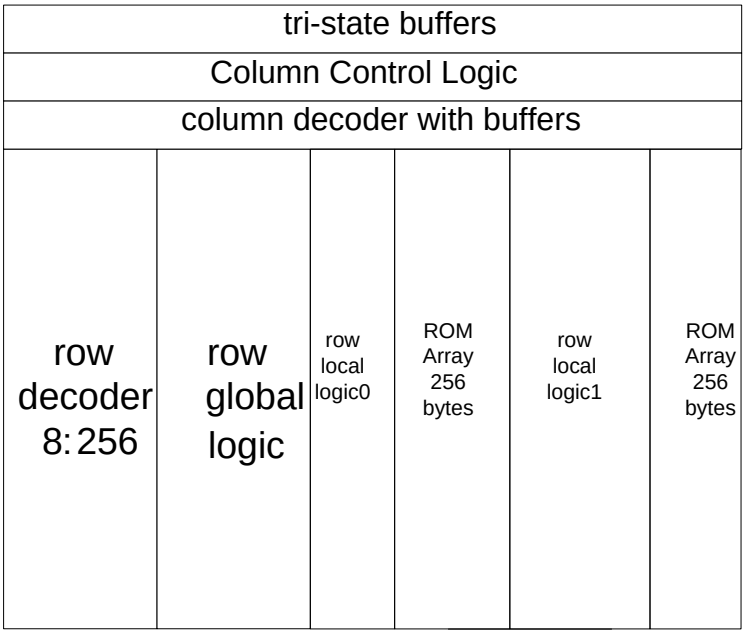


Figure 1. HC11 chip 512 bytes ROM block diagram

1.1 Critical timing and power consumption

	size	Decoder delay	Read access time	Bit line delay	Power consumption	area
HC11 on-chip ROM	512 size	20 ns	280 ns	20 ns	0.002W	0.97 mm^2

1.2 Architeture

ROM has the similar structure as the 4k SRAM (Appendix 8) except the ROM cell and sense amp (Fig. 3). Sense amp is not used in ROM because of the strong drive ability of ROM cell and much less column line capacitance.

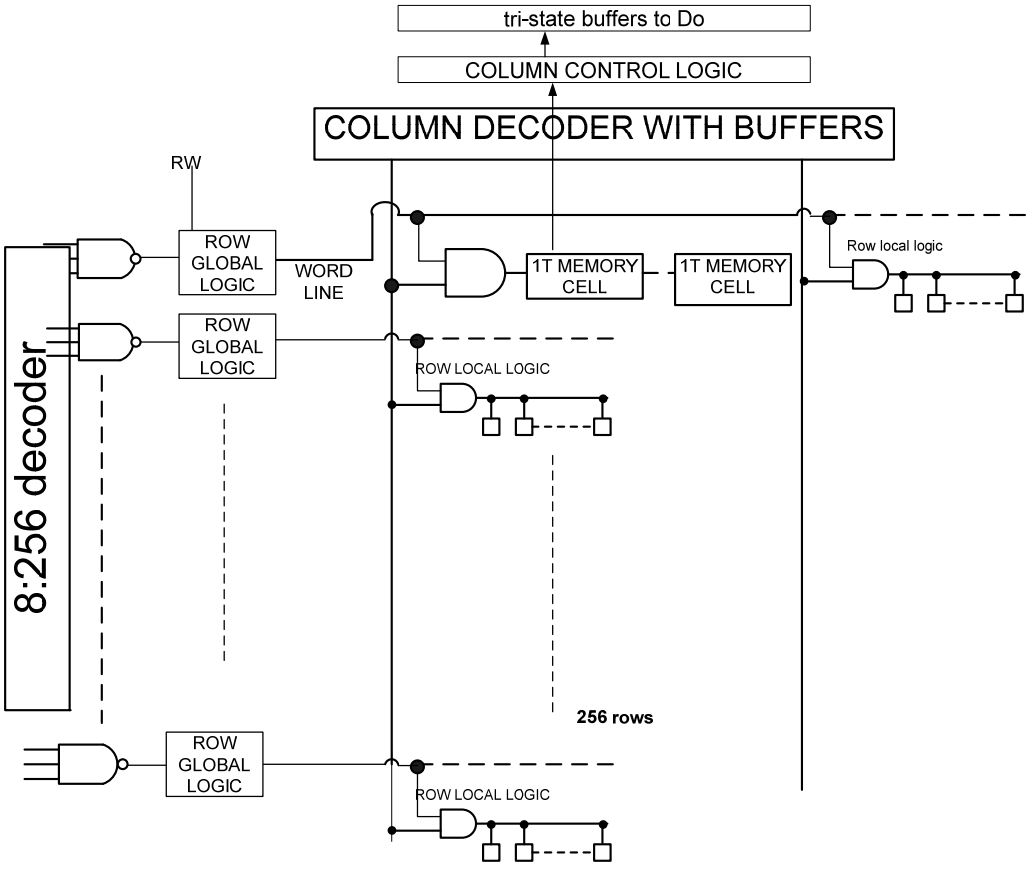


Figure 2 The architecture of the on-chip ROM

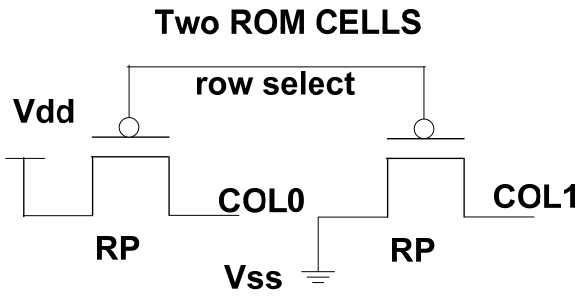


Figure 3 ROM cell schematic

1.3 ROM mask generation with the help of SKILL code

As shown in Fig. 3, the ROM cells may connect with either Vdd or Vss and are read from column lines. After the ROM layout without connection to Vdd/Vss layout is finished, with metal lines placed on the original layout to form the desired logic by using SKILL code written to instantiate the desired logic.

2. HC11 ROM Pin Descriptions

CPU and Module Interface Connections

Port	Width	Direction	Description
E	1	input	E-clock input
Phi1	1	input	Ph1-clock input
RW	1	input	Read/write control signal
Addr<15:0>	16	input	Address
D<7:0>	8	output	Data output bus

3. Layout for HC11 On-chip ROM

Area: 0.97 mm^2
Power: 0.002W

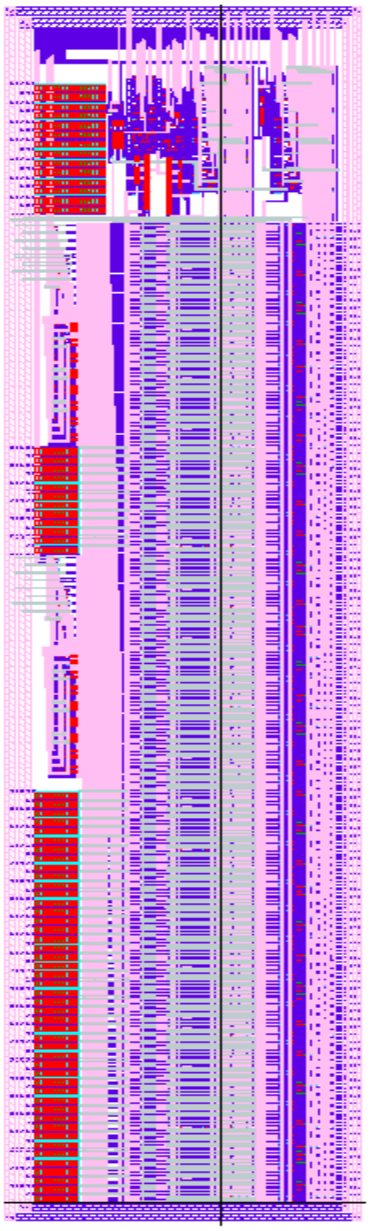


Figure 4. 512 bytes ROM layout

4 Simulation Results

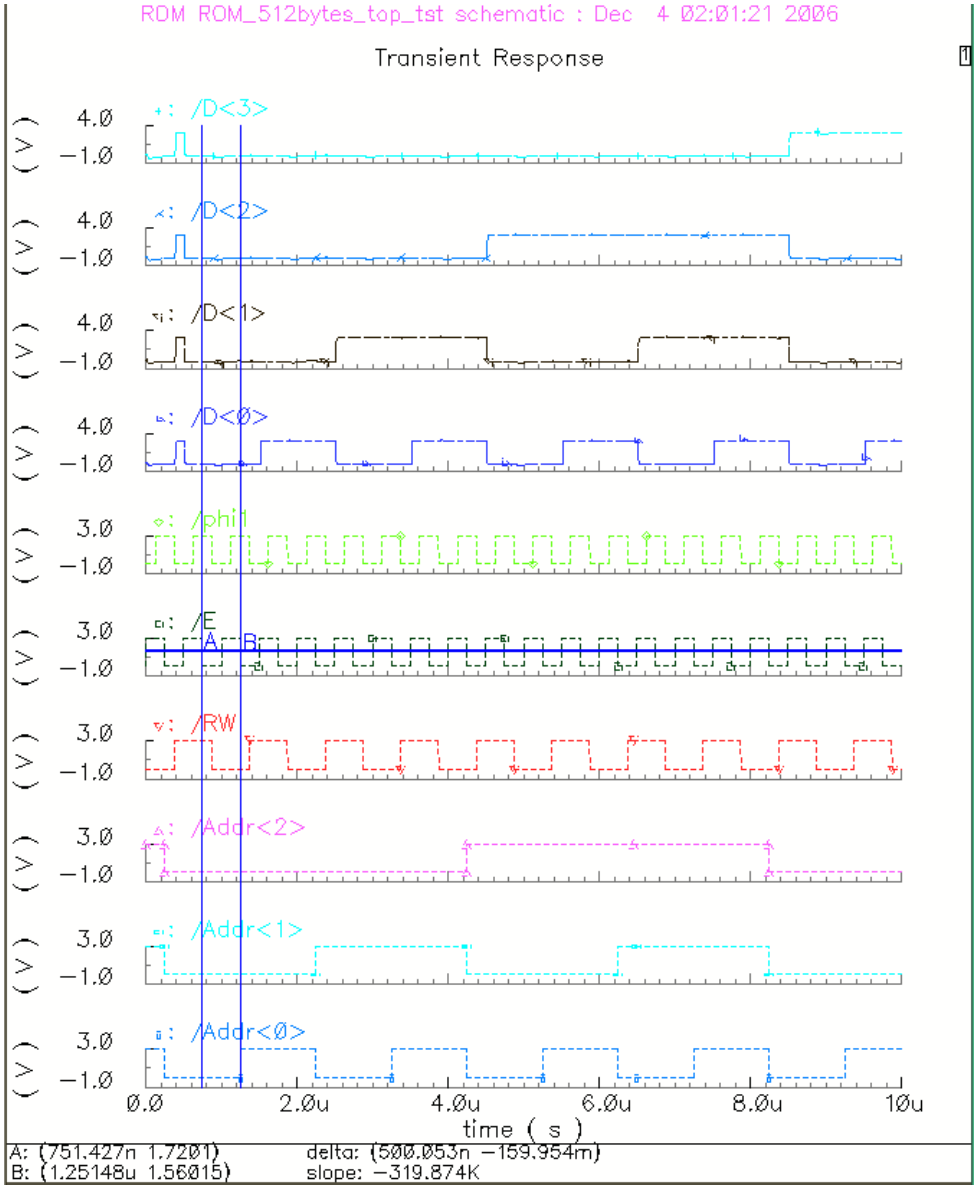


Figure 5. HC11 chip 512 bytes ROM simulation

5 Source File

Location—msvlsi:/export/home/HC11/ ROM_CORE_907/ROM_core_top_sep

APPENDIX 8

HC11 On-CHIP SRAM DOCUMENTS

TABLE OF CONTENTS

DOCUMENT **PAGE**

1 HC11 On-Chip SRAM

OSU HC11 has an on-chip SRAM of size 4k bytes, which is constructed from 16 SRAM banks logic decoders, row and column, and control logic. An SRAM bank includes: RAM cells, sense amps, write circuitry and buffers, and which are arrayed in a 256x8 arrangement. The SPI-SRAM is designed for low power applications up to 275 °C operating at 8 MHz.

- Total Memory Size : 4k bytes
- Number of Banks : 16
- One Bank's size : 256 rows by 8 Cols (0.235 mm^2)
- Module Area(4K) : 8.25mm^2 (3.75mm by 2.2mm)
- Switch and standby Power: 9.2mW (at 275 C and 8MHz)
- Standby Power :2.1mW (at 275 C)
- Read Access Time : 280ns
- Write Access time : 385ns
- Decoder Delay : 20ns
- Bit Line Delay : 30ns
- Useful read time : 50ns
- Useful Write time : 20ns

2 HC11 SRAM Block Diagram

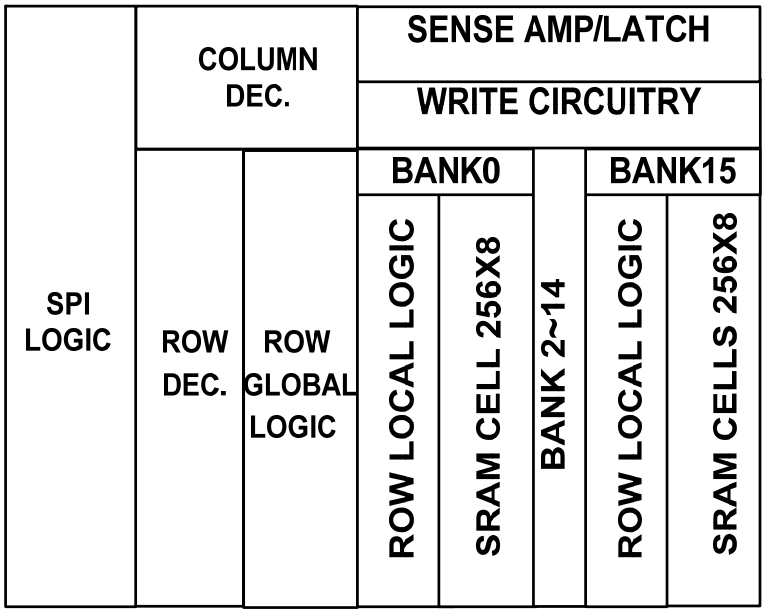


Figure. 1 The structure of 4k SPI SRAM

3 Design considerations

3.1 low power considerations

At elevated temperatures, Peregrine PMOS devices leak less than NMOS and affecting the Ion/ Ioff ratio [2]. To overcome the Ion/Ioff ratio and leakage problems and reduce power consumption, a larger than minimum channel length is used for both PMOS and NMOS devices in the SRAM design. Additional architectural features are included to enhance performance and reduce power consumption. These include [15]: Divided word line, predecoding technique and a PMOS voltage divider for the pre-charge voltage reference, VB equals VDD/2. As shown in Figure. 4, an enable signal, EN allows VB to be switched off to save power when not in use.

The RC delay associated with word lines and bit lines grows proportionately with the greater number of cells along the columns and rows respectively. Word line loading by the SRAM cell's access/pass transistors along the row and is proportional to the number of columns or bit lines. Additionally the power dissipation on the word lines increases linearly with capacitance. The use of divided word line techniques reduces the associated power consumption. Power consumption is directly depending on the required settling of the bit lines.

$$V_{final} = V_{settle} \bullet (1 - e^{-t \bullet gm / C_{COL}}) \qquad (1)$$

As indicated in equation (1), the relationship of delay and transconductance (gm) of the PMOS1/PMOS2, where V_{final} equal 90% VDD/2. , V_{settle} equal VDD/2.

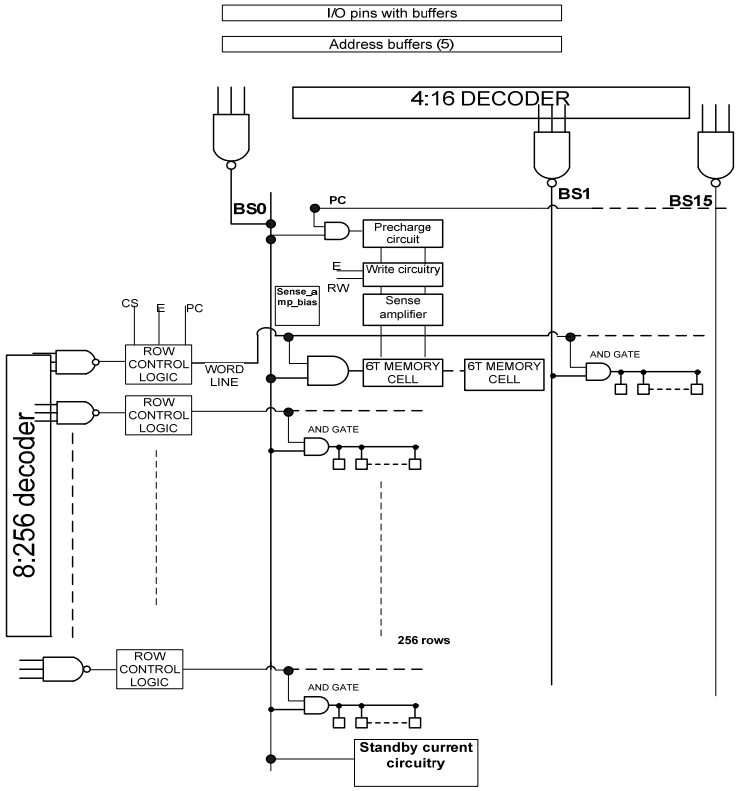


Figure. 2 The architecture of 4k SRAM

3.2 8 to 256 Decoder

Predecoding is used in decoder design to reduce the power consumption, increase the circuit speed and reduce the burden for hand-layout. With the simulation, carried out across at worst case corner(275 °C slow model,3V),the decoder delay is 15 ns. The layout area is 0.72 mm^2.

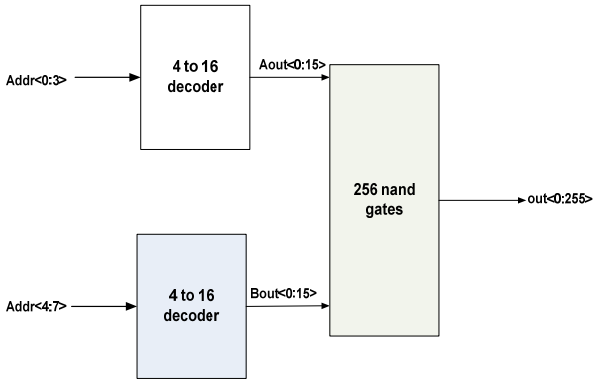


Figure. 3 8 to 256 decoder

3.3 PMOS diode Bias circuit

PMOS3 is a switch to save the power, which is on only during SRAM precharge timen(Figure. 4). PMOS1 and PMOS2 are properly sized to assure fast charging or discharging the bit line capacitance while at the same time not to wasting power. PMOS3 is a much wider transistor than PMOS2 and PMOS1 to lower the on resistance and assure VB is maintained at Vdd/2.

$$V_{final} = V_{settle} \bullet (1 - e^{-t \bullet gm / C_{COL}}) \quad (2)$$

Equation (1), establishes the relationship of delay to the gm of PMOS1/PMOS2 and the Column capacitance where V_{final} =Vdd/2 = V_{settle} .

Compared to the voltage regulator design in the previous submission (Jan. 2007), the PMOS diode bias circuit saves more than 80% on bias or stand by power consumption. In the first submission, a voltage regulator was used to bias the circuit to Vdd/2. By comparison the PMOS diode bias circuit, voltage regulator a bias generator which uses extra power and the opamp voltage regulator. In addition the voltage regulator was not switched in and out on a per column basis.

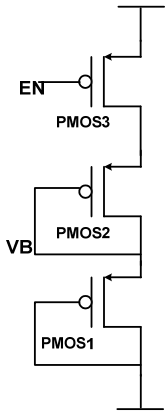


Figure.4 PMOS diode bias circuit

3.4 Sense amp design

The SRAM read circuitry is show in Figure. 5. In the read circuitry, the sense amp sense signal (SE) is delayed to assure that the SRAM cell drives the sensed signal amplitude at the sense amp inputs to a value larger than the anticipated input referred offset of the sense amp. This ensures a reliable read b the sense amp. CLOCK 2, figure 5, controls both SRAM row select and sense amp sense enable signal. The DELAY is designed to ensure that SRAM conversion does not start prematurely across the temperature corners. [6]. The DELAY circuit is designed based on the following analysis.

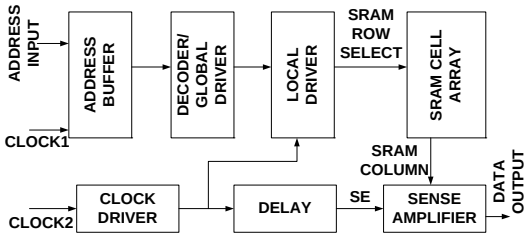


Figure.5 SRAM read circuitry

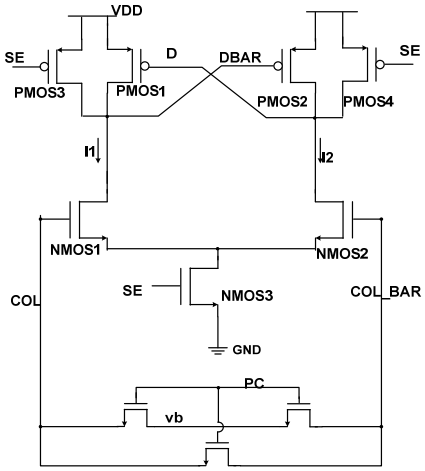


Figure. 6 current latch sense amp with precharge

The sense amp is shown in Figure. 6. The read cycle starts with the precharging. COL and COL_BAR to VDD/2, while D and DBAR are precharged to VDD. Precharging the columns to VDD/2 decreases the sense amp delay. With COL and COL_BAR precharged to VDD/2, the sense amplifier is enabled by SE, the SRAM cell select line. After the column sense signal, ΔV, is ensured of exceeding sense amp the offset voltage, $V_{os,}$ the sense amp is allowed to regenerate. Sizing up of P1, P2, N1, and N2 minimizes sense amp V_{os} reducing settling time at the expense of power and with little or no decrease in delay. The column sense or cell delay is represented by the following equation,

$$t_{cell} = C_{COL} \bullet \Delta V / I_{cell}\,, \qquad (3)$$

where C_{COL} is the SRAM COL line capacitance, I_{cell} is SRAM cell on current. The total read timing is approximated by equation (4):

$$t_{read} = C_{COL} \bullet \Delta V / I_{cell} + 2 \bullet C_{gdop} \bullet V_{thp} / I_{SA} + \frac{Cgsp}{gm_p + gm_n - go} \ln \frac{V_{final}}{\Delta V} \qquad (4)$$

where C_{gsp} and C_{gdop} are the gate to source capacitance and over overlap capacitance of P1 and P2. $gm_{n,}$ and $gm_{p,}$ are the transconductance, of the NMOS

differential pair and PMOS cross coupled pair respectively. I_{SA} is the tail current of sense amp, and go is the output conductance at node D or DBAR, V_{final}=VDD-V_{thp}.

3.5 SRAM cell

The 6T SRAM cell, Figure. 7, uses PMOS devices to reduce both area and leakage current. Cell and pull-up ratios are calculated to assure the read and write stability. Cell ratio is defined as the size ratio between pull down transistor (N1,N2) and pass transistor (P3, P4) and pull-up ratio of the cell is defined as the size ratio between pull up transistor (P1, P2) and pass transistor (P3, P4) [14].

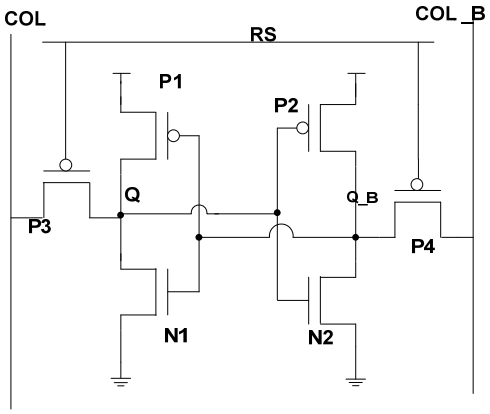


Fig 7. 6T SRAM cell schematic.

3.6 Standby current circuitry

As shown in Figure.8, when BS0 is 1, Bank 0 is selected, $\overline{BS0}$= 0 and turns on pdrive transistor and turns off ndrive transistor and VVDD drives to 3.3V. Bank1 ~ Bank15 are not selected and ndrive transistors are turned on, pdrive transistors are turned off. In this case, the SRAM cells power VVDD is 2.2V and the SRAM banks work in the standby mode.

The large power switch with transistors need to be large enough for 256*8 SRAM cells to draw the current (pdrive need 256*8*1.3uA = 2.1 mA, ndrive need 256*8* 0.7uA =1.5mA) without an IR drop in the power supply. Based on simulation, the large pdrive and ndrive transistors must be 20@ 1.2 um / 0.8 um, and 15@ 1.4um/1.4um respectively.

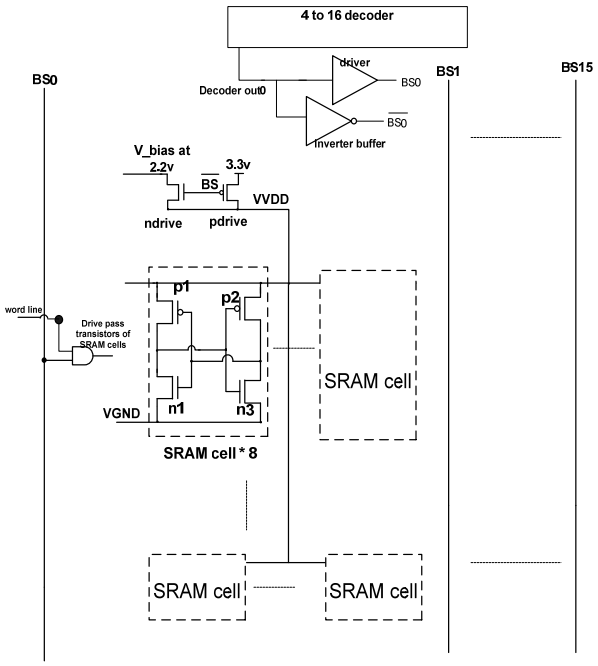


Figure. 8 standby current circuitry

4 Layout for HC11 On-chip SRAM

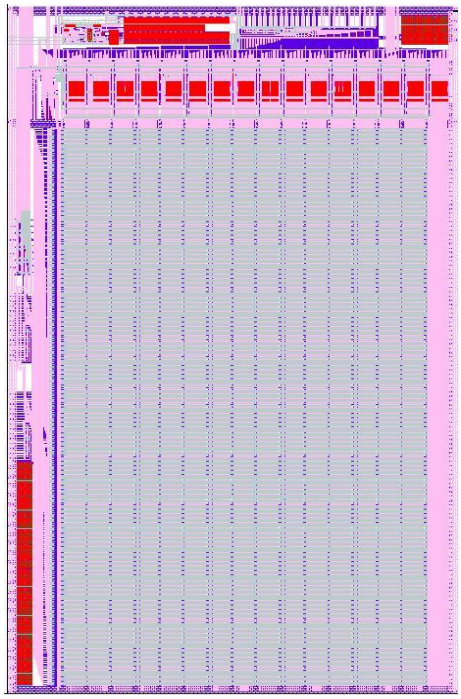


Figure. 9 4k SRAM layout

5 HC11 SRAM Pin Descriptions

Pin Name.	Pin Function	Pin Direction	
E	E clock	Input	1 pin
RW	Read/Write	Input	1 pin
STNBY	Memory Standby signal from CPU	Input	1 pin
VBIAS	Turn-off Control for Internal Bias-Generator	Input (external)	1 pin
address<15:0>	Address signals	Input	16 pins
phi1	phi1 Clock	Input	1 pin
vb	Pre-charge Voltage for Cell Read	In/Out	1 pin
VRAM	Internal Cell-array Power Supply monitoring pin	In/Out (external)	1 pin
STNBY_EN	Standby mode turn-on control	Input (external)	1 pin
VSTNBY	Standby power supply	In/Out (external)	1pin
Din<7:0>	Data input	In	8 pins
Dout<7:0>	Data output	Out	8 pins

6 Simulation Results

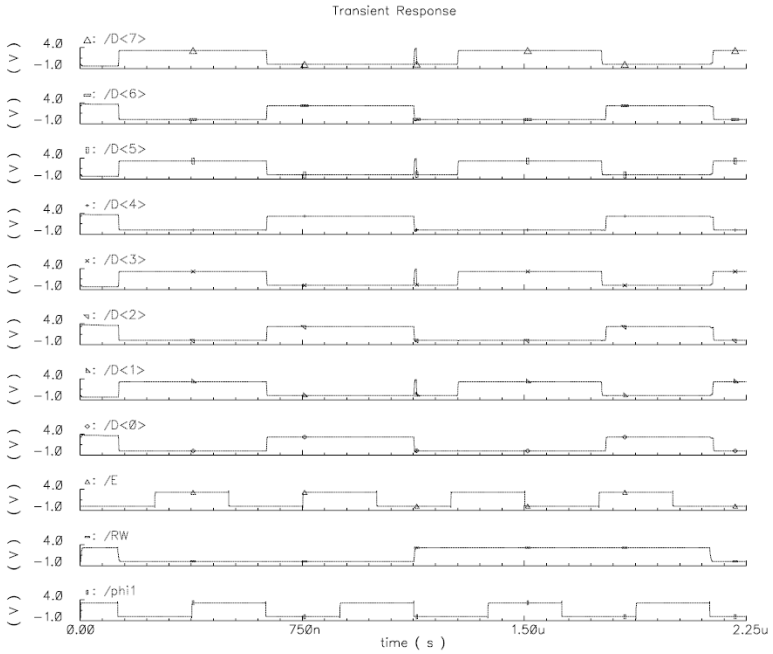


Figure. 10 4k SRAM simulation

7 Source File

Location (2008)--msvlsi: /export/home/zyuan/HC11/SRAM_CORE1216/SRAM_4k
Location (2007)-- msvlsi:/export/home/zyuan/HC11/ROM_4K_LAYOUT/ SRAM_4k

References

[1] V. Jeyaraman, "Design, characterization, and automation of a high temperature (200 °C) standard cell library," in *Electrical and Computer Engineering*. Stillwater, Oklahoma: Oklahoma State University, 2004.

[2] U. Badam, S. Viswantathan, V. Jeyaraman, C. Hutchens, C. Liu, and R. Schultz, "High temperature SOS cell library,"presented at International Conference on High Temperature Electronics (HITEC), Santa Fe, New Mexico, 2006.

[3] W.Agaststein, K. McFaul, P.Themins, "Validating an ASIC Standard Cell Library", Intel Corporation, 1990.

[4] Chris Hutchens, Steven Moris, and Chia-min Liu, "A proposed 68HC11 chip set for 275 degrees C," IMAPS International Conference on High Temperature Electronics (HiTEC 2006), Santa Fe, NM, May 15 - 18, 2006.

[5] Chris Hutchens, Chia-Ming Liu and Hooi Miin Soo, "High temperature Down-hole Microcomputer System, Switched-Mode Power supply Component Development," *GasTIPS,* vol. 13, no. 1, 2007.

[6] J. Tao, N. Cheung, and C. Ho, "An Electromigration Failure Model for Interconnects Under Pulsed and Bidirectional Current Stressing," *IEEE Transactions on Electron Devices*, vol. 41, pp. 539, 1994.

[7] J. Tao, N. Cheung, and C. Ho, "Metal Electromigration Damage Healing Under Bidirectional Current Stress," *IEEE Electron Device Letters*, vol. 14, pp. 554, 1993.

[8] *Peregrine Semiconductor, Foundry Services. [cited Mar. 8, 2006]; Available from: http://www.peregrine-semi.com/content/foundry/foundry.html.*

[9] *68HC11 Reference Manual, Document 68HC11RM/D, Rev 6 4/200,. Freescale Semiconductor Inc, 6501 William Cannon Drive West, Austin, Texas, U.S.A., 2002.*

[10] Nambu, H.; Kanetani, K.; Yamasaki, K.; Higeta, K.; Usami, M.; Fujimura, Y.; Ando, K.; Kusunoki, T.; Yamaguchi, K.; Homma, N., "A 1.8-ns access, 550-MHz, 4.5-Mb CMOS SRAM," *IEEE Journal of Solid-State Circuits*, vol.33, no.11, pp.1650-1658, Nov 1998.

[11] Kobayashi, T.; Nogami, K.; Shirotori, T.; Fujimoto, Y., "A current-controlled latch sense amplifier and a static power-saving input buffer for low-power architecture," *IEEE Journal of Solid-State Circuits*, vol.28, no.4, pp.523-527, Apr 1993.

[12] Seki, T.; Itoh, E.; Furukawa, C.; Maeno, I.; Ozawa, T.; Sano, H.; Suzuki, N., "A 6-ns 1-Mb CMOS SRAM with latched sense amplifier," *IEEE Journal of Solid-State Circuits*, vol.28, no.4, pp.478-483, Apr 1993.

[13] Takeda, K.; Hagihara, Y.; Aimoto, Y.; Nomura, M.; Nakazawa, Y.; Ishii, T.; Kobatake, H., "A read-static-noise-margin-free SRAM cell for low-VDD and high-speed applications," *IEEE Journal of Solid-State Circuits*, vol.41, no.1, pp. 113-121, Jan. 2006.

[14] Jan M.Rabaey, ”Digital Integrated circuit- A design perspective”,p.658-p.661.

[15] Amrutur, B.S.; Horowitz, M.A., "Fast low-power decoders for RAMs," *IEEE Journal of Solid-State Circuits*, vol.36, no.10, pp.1506-1515, Oct 2001.

[16] Lovett, S.J.; Gibbs, G.A.; Pancholy, A., "Yield and matching implications for static RAM memory array sense-amplifier design," *IEEE Journal of Solid-State Circuits*, vol.35, no.8, pp.1200-1204, Aug 2000.

APPENDIX 9

4K SPI BUS SERIAL ROM DOCUMENTS

TABLE OF CONTENTS

DOCUMENT **PAGE**

1 Description

The SPI serial bus ROM is 2Kbyte memory device. The memory is accessed through a serial peripheral interface (SPI) bus. The SPI is a serial synchronous communication protocol that requires minimum of 3 wires, clock input (sck), data in (di) and data out (do) bus lines. The device is enabled through the chip select enable pin (csn).

The device has to be reset via reset (rst) pin, a toggle of low to high transition to complete the reset cycle. Since the device does not have an on-chip crystal/ clock oscillator, the external clock source is required to supply through clock input line (clk_i). The device supports two operating modes with cpol = 0, cpha = 0 and cpol = 1 and cpha = 0.

To help in debug, pins ld, ce and clk_rom are used for debugging purpose. For receiving every 8-bit, ld will be asserted high and returned to the low state. Pin clk_rom is the clock supplied to the ROM to shift out the data from ROM to SPI data buffer.

1.1 Features

Max clock 8MHz

3.3V low-power CMOS technology

2K x 8bit organization

Sequential read (Page Read/Burst mode) not supported

Read cycle time: 280ns max.

Temperature range supported: -125 ^{o}C to +275 ^{o}C

Table 1. Instruction Set

Instruction Name	Instruction format	Description
READ	0000 0011	Read data from memory array at the selected address

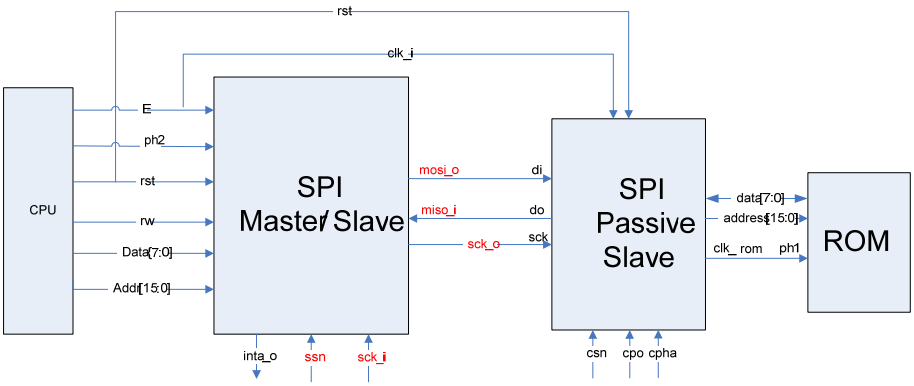


Figure 1 Master and Slave SPI devices communication diagram.

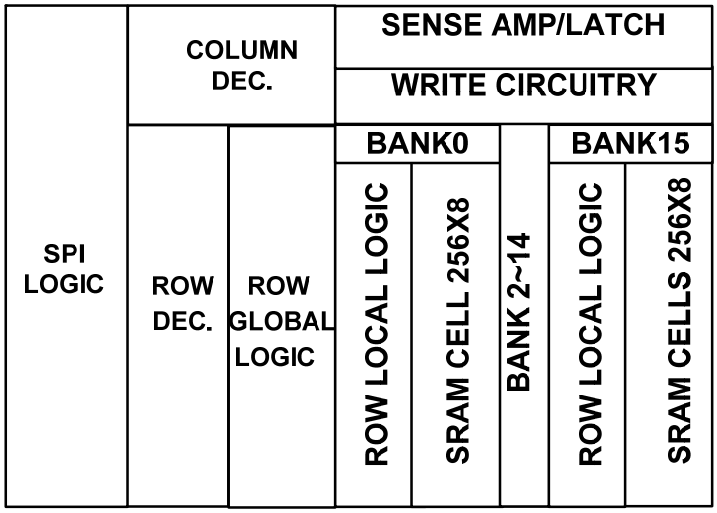


Figure 2 ROM architecture diagram.

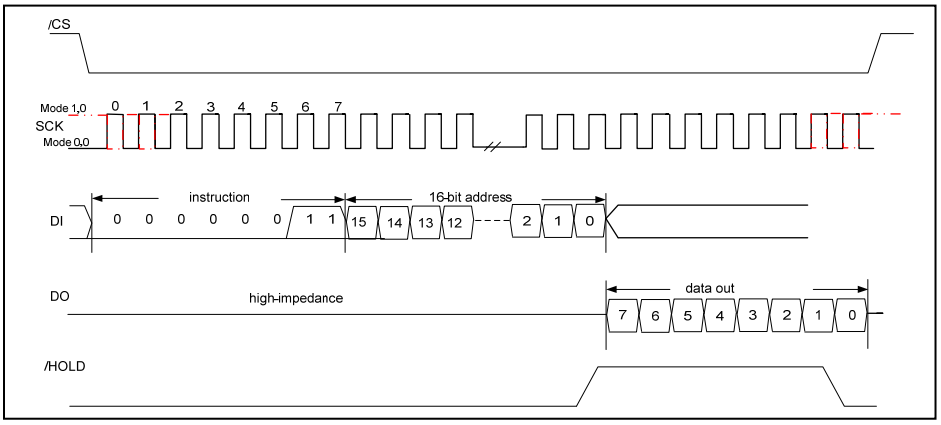


Figure 3 Serial read sequence timing.

2 Pins

Pads Out(External Connections)

Port	Width	Direction	Description
ctrl_en	1	input	Control enable
lrw	1	output	Scan chain enable
clk_rom	1	output	Debug pin. Clock pulse supplys by SPI slave to the ROM memory circuitry.
vss	1	input/output	Power pin
ce	1	output	Debug pin. Toggle bit to start the transferring of data from ROM to SPI data buffer
sck	1	output	SPI input clock
di	1	input	Slave data in
do	1	input/output	Slave data out
csn	1	input	chip select/enable active low
e	1	input	E-clock
rst	1	input	System rest
cpol	1	input	Mode select: CPHA,CPOL: 00 or 11
cpha	1	input	Mode select: CPHA,CPOL: 00 or 11
ld	1	output	Debug pin. ld will be asserted after receiving every 8-bit.
scanin	1	input	Scan input
scanout	1	output	Scan output
scanclk	1	input	Scan clock
Eclki	1	input	E clock
sck	1	output	SPI input clock
csn	1	input	Chip select/enable active low

SPI Slave and ROM Circutry Interface Connections

Port	Width	Direction	Description
data	8	input/output	Data input
address	16	input	The memory location the CPU wants to write or read
clk_rom	1	input	Memory internal clock
rw	1	input	rw given by SPI
e	1	input	E-clock input
CE	1	input	Read enable signal, active high.

3 Layout

SPI ROM Description:
Size: 2K bytes
Number of banks: 8
One bank size: 256 rows by 8 columns
Module Area: 2.4 mm x 2.13 mm
Standby Leakage power: 0.66mW (at 275 C)
Switch and standby Power: 1.3mW (at 275 C and 8MHz)
Row decoder delay: 20ns
Read access time: 280ns
Bit line delay: 20ns

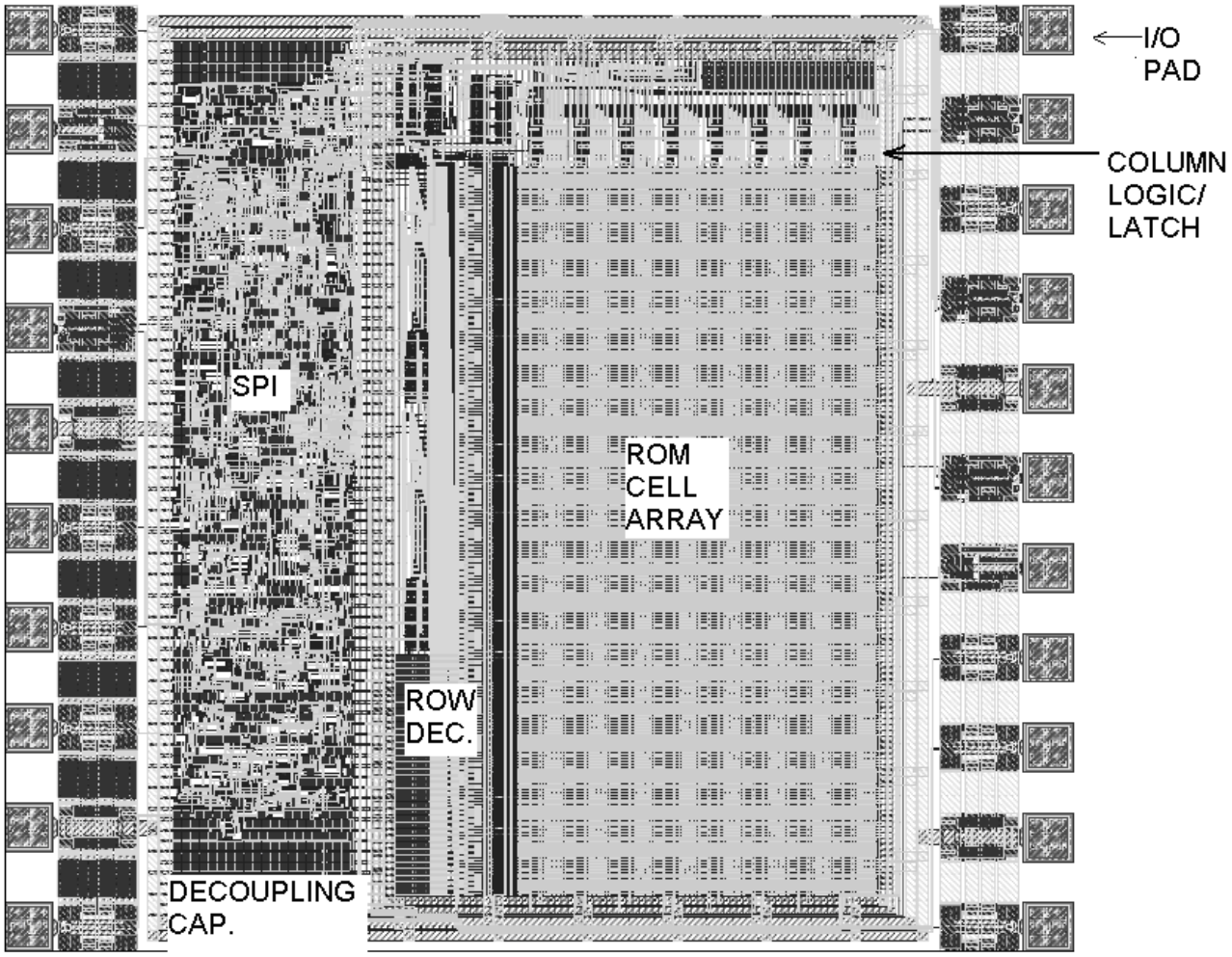
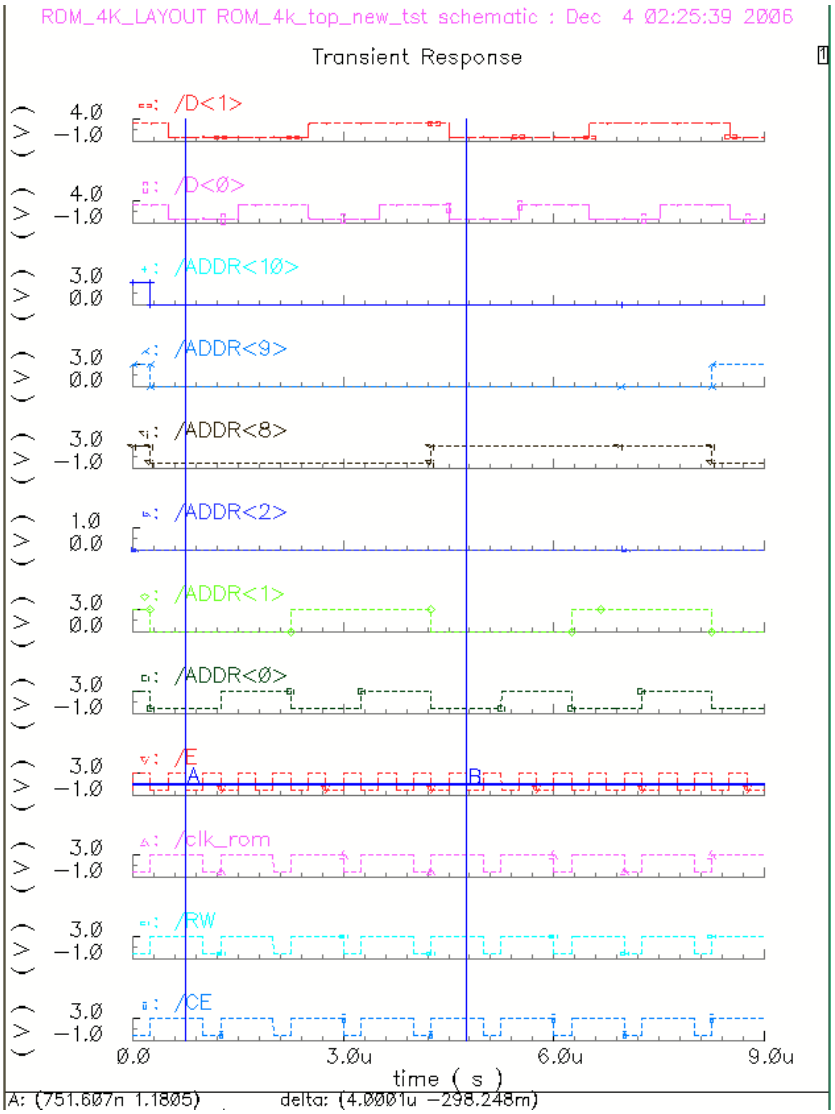


Figure 4 The layout of 2k SPI ROM

4 Testing and Simulation

Simulator: Cadence Simulation tool

The timing and functional test with parasitic capacitance is tested on the cadence simulation tool.



HC11 off chip 2K ROM simulation

5 Electrical Characteristics

Maximum Ratings

Vcc……..….3.6V
All input and outputs w.r.t. Vss……..3.6V
Storage temperature…….....-65°C to 275°C
Ambient temperature under bias....-65°C to 275°C

Table 1: DC Characteristics

T_A = -65°C to 275°C Vcc = 2.0V to 3.3V

Parameter	Symbol	Min	Max	Units	Test conditions
High level input voltage	V_{IH}	2.0	Vcc+0.7	V	
Low level input voltage	V_{IL}	-0.5	0.8	V	
Low level output voltage	V_{OL}	-	0.4	V	I_{OL} =
High level output voltage	V_{OH}	V_{CC}-0.6	-	V	I_{OH} =
Input leakage current	I_{LI}	-1.9	1.9	µA	$\overline{CS}$ = V_{CC}, V_{IN}=GND to V_{CC}
Output leakage current	I_{LO}	-5.4	5.4	µA	$\overline{CS}$ = V_{CC}, V_{OUT}=GND to V_{CC}
Internal capacitance (all inputs and outputs)	C_{INT}	-	0.250	pF	
Operating Current	$I_{CC\ read}$	-	0.39	mA	V_{CC}=3.3V;SO=Open, F_e= 8MHz (Note)
Standby Current	I_{CCS}	-	0.20	mA	$\overline{CS}$ = V_{CC}

Note: This parameter is periodically sampled and not 100% tested.

Table 2: AC Characteristics

T_A = -65°C to 275°C Vcc = 2.0V to 3.3V						
Param. No.	Symbol	Parameter	Min	Max	Units	Test conditions
1	F_e	E Clock Frequency	-	8	MHz	$2.0V \le V_{CC} \le 3.3V$
2	F_{SCK}	SPI Clock Frequency	-	4	MHz	$2.0V \le V_{CC} \le 3.3V$
3	T_{CSS}	$\overline{CS}$ Setup Time	125	-	ns	$2.0V \le V_{CC} \le 3.3V$
4	T_{CSH}	$\overline{CS}$ Hold Time	250	-	ns	$2.0V \le V_{CC} \le 3.3V$
5	T_{CSD}	$\overline{CS}$ Disable Time	125	-	ns	$2.0V \le V_{CC} \le 3.3V$
6	T_{SU}	Data Setup Time	4	-	ns	$2.0V \le V_{CC} \le 3.3V$
7	T_{HD}	Data Hold Time	16	-	ns	$2.0V \le V_{CC} \le 3.3V$
8	T_R	SCK Clock Rise Time	-	2	µs	(Note)
9	T_F	SCK Clock Fall Time	-	2	µs	(Note)
10	T_{HI}	SCK Clock High Time	0.125	-	µs	$2.0V \le V_{CC} \le 3.3V$
11	T_{LO}	SCK Clock Low Time	0.125	-	µs	$2.0V \le V_{CC} \le 3.3V$
12	T_{CLD}	SCK Clock Delay Time	62.5	-	ns	$2.0V \le V_{CC} \le 3.3V$
13	T_V	Output Valid from SCK Clock Low (mode cpol = 0, cpha = 0) Output Valid from SCK Clock High (mode cpol = 1, cpha = 0)	- -	125 125	ns ns	$2.0V \le V_{CC} \le 3.3V$
14	T_{HO}	Output Hold Time	16	-	ns	
15	T_{DIS}	Output Disable Time		80	ns	(Note)
16	T_{HS}	$\overline{HOLD}$ Setup Time	125	-	ns	$2.0V \le V_{CC} \le 3.3V$
17	T_{HH}	$\overline{HOLD}$ Hold Time	250	-	ns	$2.0V \le V_{CC} \le 3.3V$
18	T_{RC}	Internal Read Cycle Time (byte)	-	½ Te +30	ns	$2.0V \le V_{CC} \le 3.3V$

Note:
1. This parameter is periodically sampled and not 100% tested.
2. Refer to Figure 5 and Figure 6.
3. $T_e = 1/F_e$.

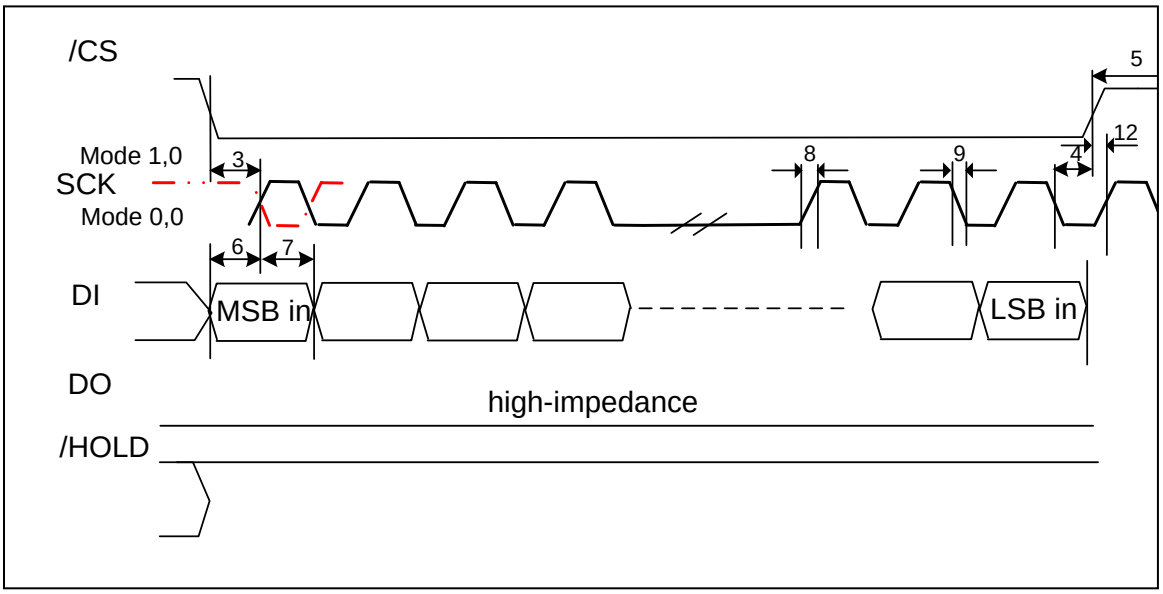


Figure 5 Serial input timing.

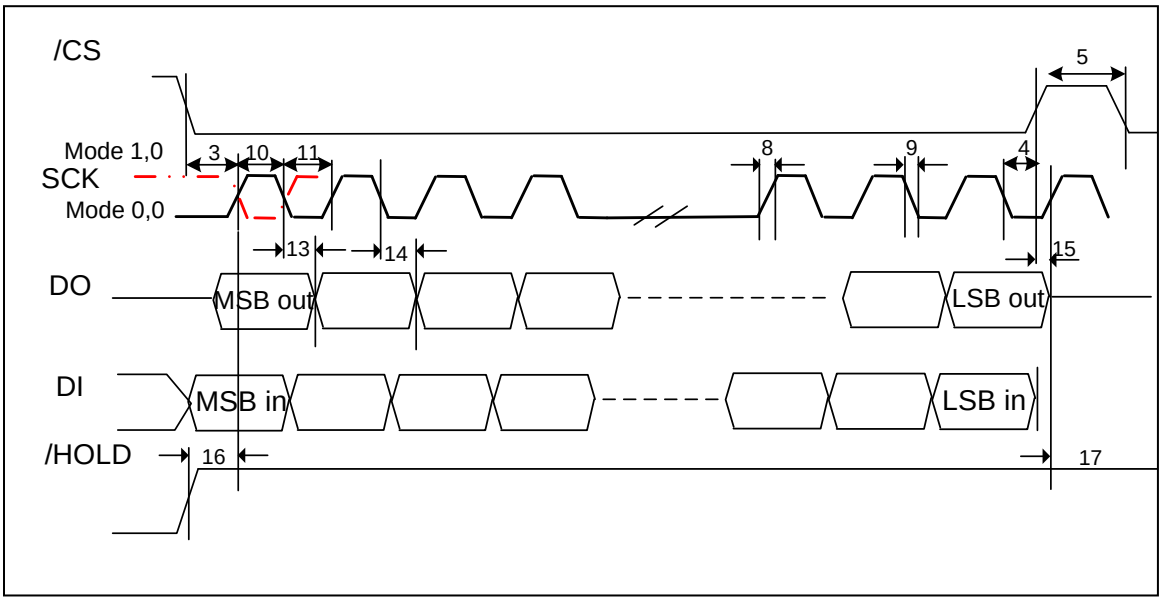


Figure 6 Serial output timing.

APPENDIX 10

4K SPI BUS SERIAL SRAM DOCUMENTS

TABLE OF CONTENTS

DOCUMENT **PAGE**

1 Description

The SPI serial bus SRAM is 4Kbyte memory device (Figure 2). The memory is accessed through a simple serial peripheral interface (SPI) compatible serial bus. The SPI is a serial synchronous communication protocol that requires minimum of three wires, the SPI clock input (**sck**), data in (**di**) and data out (**do**) bus lines. The device is enabled through the chip select enable pin ($\overline{csn}$)by setting the $\overline{csn}$ low (Figure 1).

The device has to be reset via the ($\overline{rst}$) pin, a toggle of low to high transition completes the reset cycle. Since the device does not have an on-chip crystal/clock oscillator, the external clock source is required to be supplied through clock input line (**mclk**). The device supports two operating modes with cpol = 0, cpha = 0 and cpol = 1 and cpha = 0. The read operation is shown in Figure 3.The byte write sequence is shown in Figure 4. The burst write mode is not implemented.

To help in debug, pins **ld**, **ce** and **clk_sram** are used for debugging purpose via an internal scan chain. For receiving, every 8-bit, **ld** will be asserted high and back to a low state. A pulse can be observed at pin **ce** as a toggle of transferring data from SRAM to SPI data buffer. Pin **E_ram** is the clock pulse applied to the SRAM to shift out the data from SRAM to SPI data buffer.

1.1 Features

Max clock 8MHz

3.3V low-power CMOS technology

4K x 8bit organization

Sequential read (Page Read/Burst mode) not supported

Read cycle time: 425 ns max.

Write cycle time: 375 ns max.

Temperature range supported: -25 ^{o}C to +275 ^{o}C

Table 1. Instruction Set

Instruction Name	Instruction format	Description
READ	0000 0011	Read data from memory array at the selected address
WRITE	0000 0010	Write data to memory array at the selected address

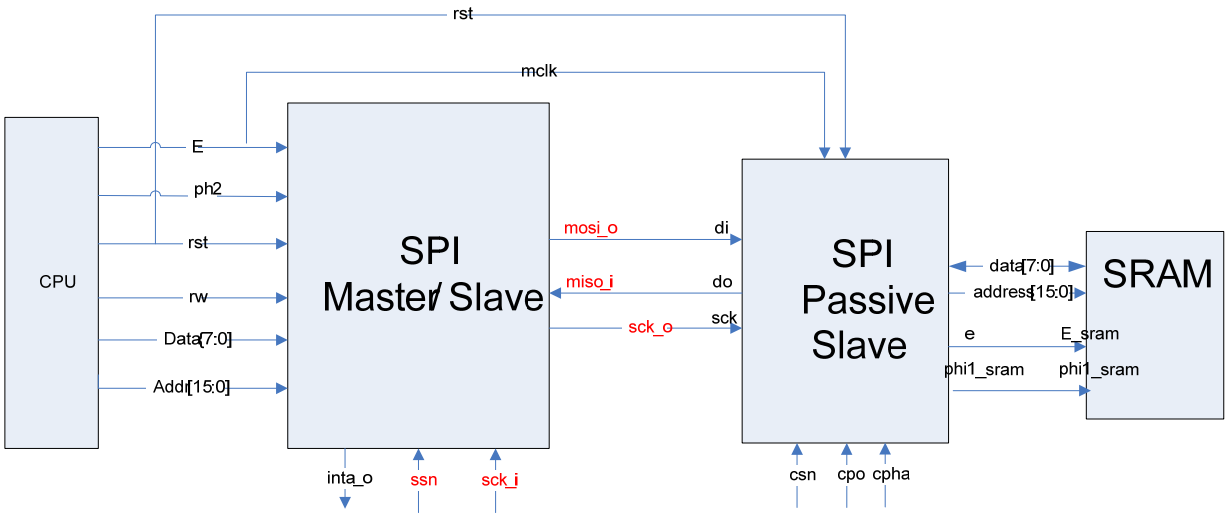


Figure 1 Master and Slave SPI devices communication diagram.

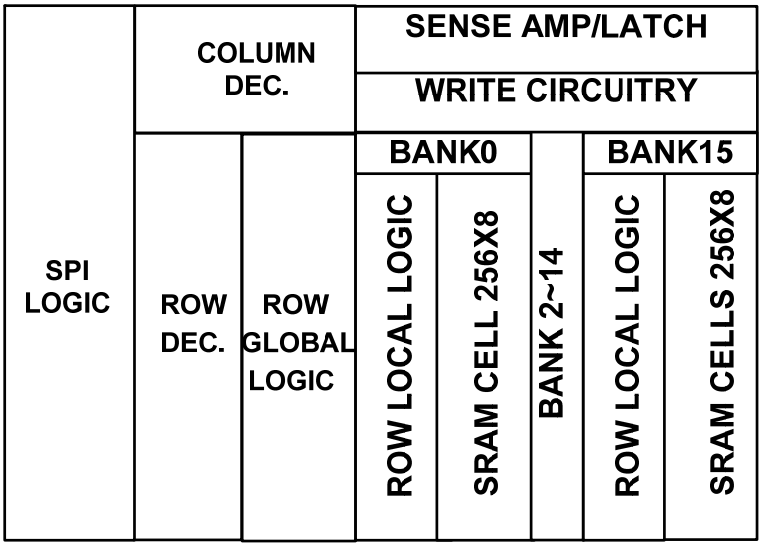


Figure 2 SRAM architecture diagram.

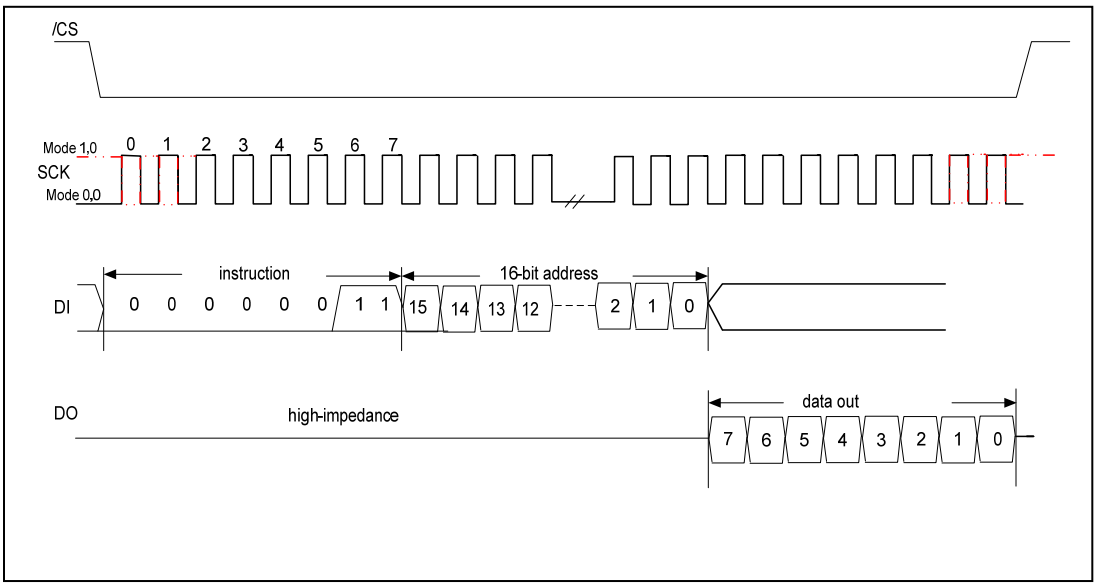


Figure 3 Serial read sequence timing.

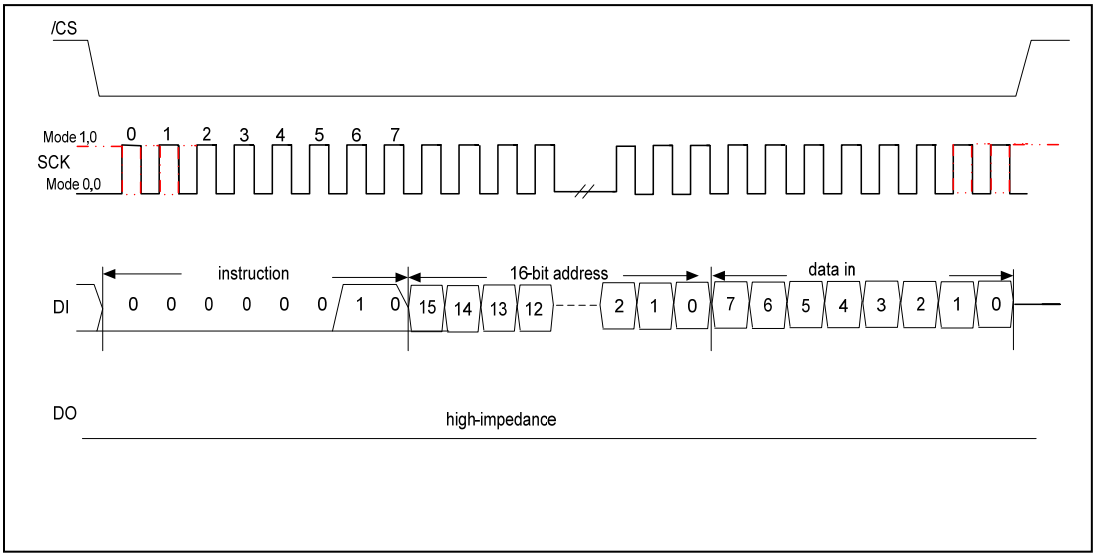


Figure 4 Serial write sequence timing.

2 Pins

Package pins

Port	Package pin number	Width	Direction	Description
di	1	1	input	Slave data in
sck	2	1	output	SPI input clock (1/8 frequency of mclk)
$\overline{csn}$	3	1	input	Chip select/enable active low
vdd	4	1	input/output	Power digital
vss	5	1	input/output	Power digital
$\overline{rst}$	6	1	input	System reset ,active low reset
cpol	7	1	input	Mode 0 or 1
cpha	8	1	input	Mode 0
vdd	9	1	input /output	Power digital
ctrl_en	10	1	input	Scan chain control enable, must tie to ground to disable scan chain (debugging purpose).
NC	11	1		
NC	12	1		
NC	13	1		
vss	14	1	input/output	Power digital
do	15	1	output	Slave data out
mclk	16	1	input	Main clock

Note: NC= Not connection needed.

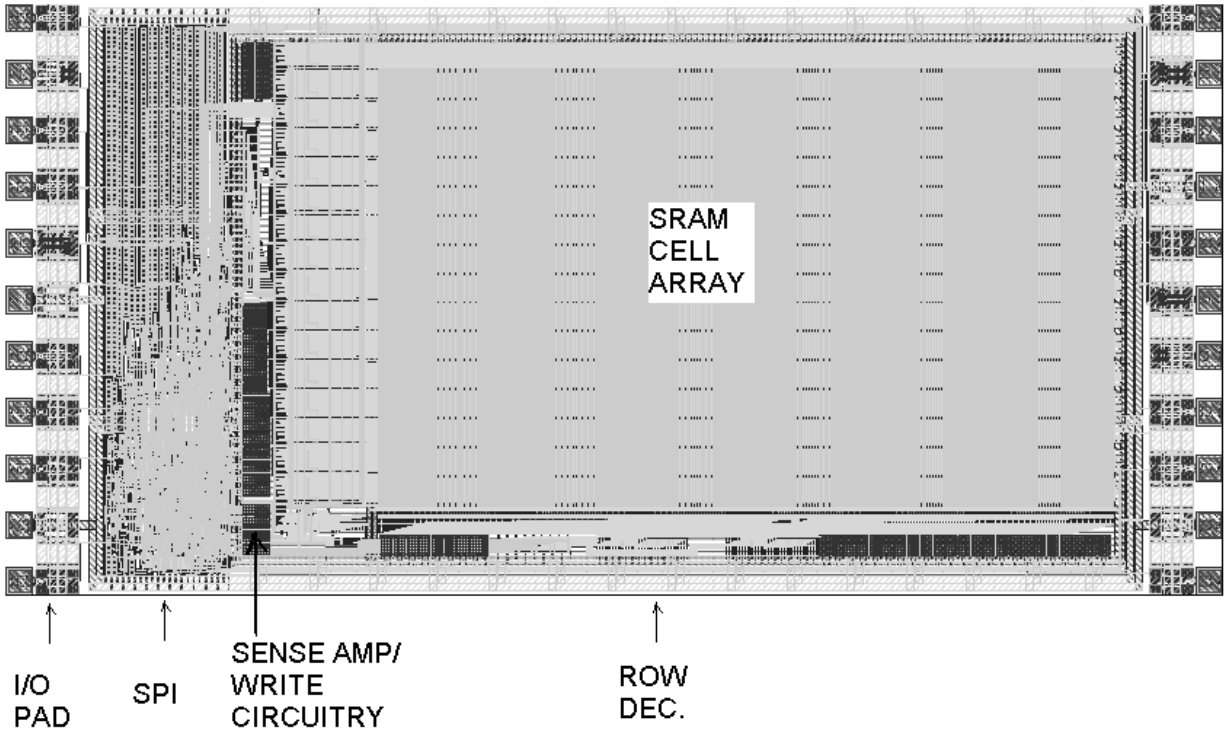
SPI Slave and SRAM Circutry Interface Connections

Port	Width	Direction	Description
D	8	input/output	Data input/output
address	16	input	The memory location the CPU wants to write or read
RW	1	input	RW given by SPI
E_sram	1	input	E-clock input
phi1_sram	1	Input	¼ cycle delayed from E
CE	1	input	Read enable signal, active high.
vdd	1	input/output	Power digital
vss	1	input/output	Power digital

3 Layout

SPI SRAM Description:
Total Memory Size: 4k bytes
Number of banks: 16
One bank size: 256 rows by 8 columns
Module area (4K): 8.5mm^2
Switch and standby Power: 9.2mW (at 275 C and 8MHz)
Standby Power: 2.1mW (at 275 C)

Read Access Time : 238ns
Write Access time : 190ns
Decoder Delay : 20ns
Bit Line Delay : 30ns
Useful read time : 50ns
Useful Write time : 20ns



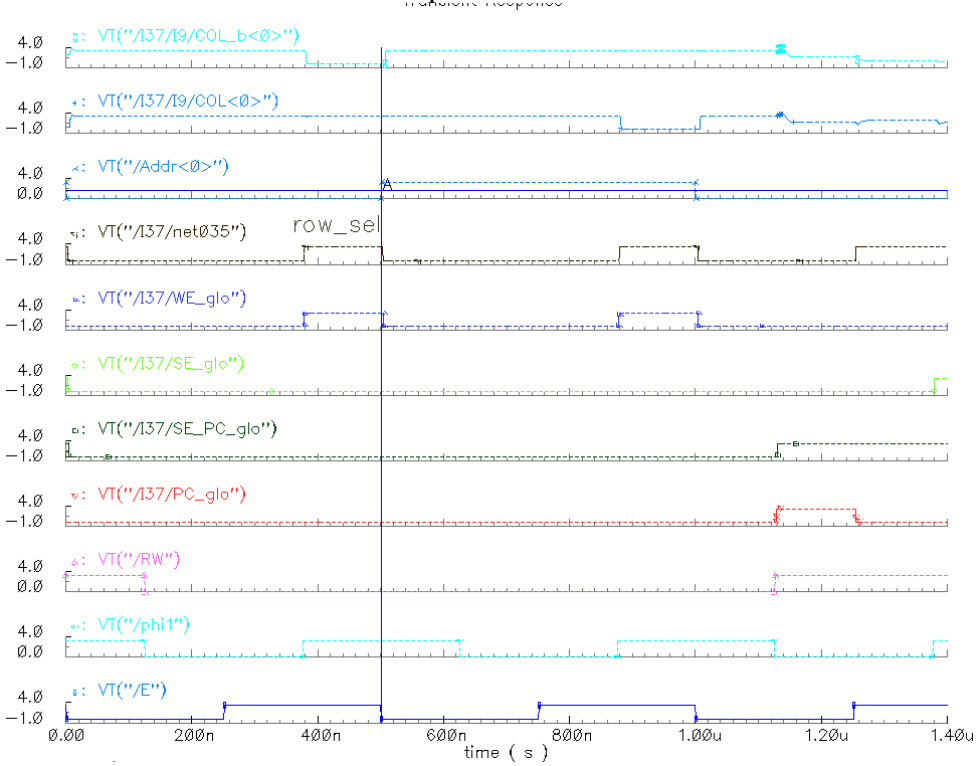
The layout of 4k SPI SRAM

4 Testing and Simulation

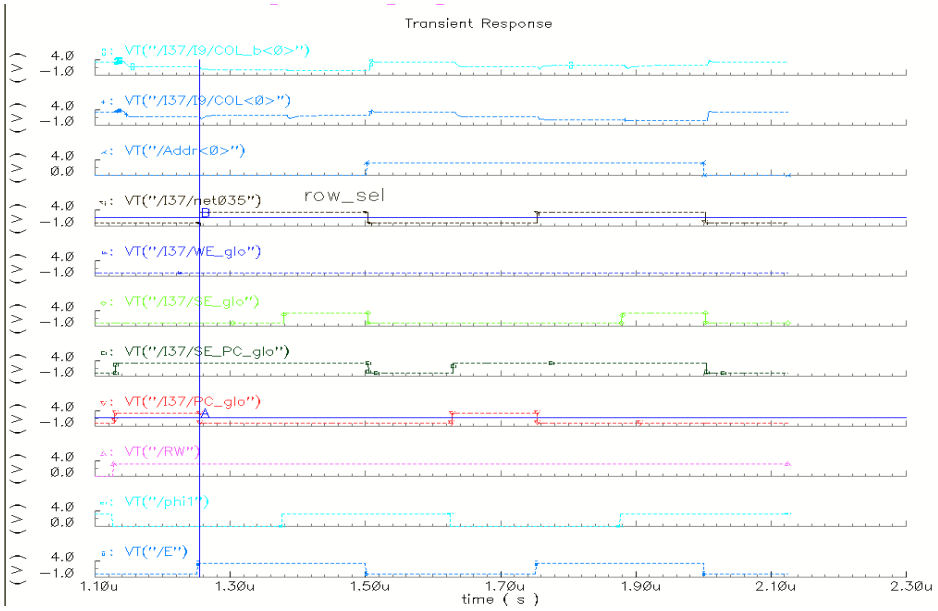
Simulator: Cadence Simulation tool

The timing and functional test with parasitic capacitance is tested on the cadence simulation tool.

HC11 off chip 4k SRAM simulation



Write cycle


Transient Response
VT("/I37/I9/COL_b<0>")
VT("/I37/I9/COL<0>")
VT("/Addr<0>")
VT("/I37/net035")
row_sel
VT("/I37/WE_glo")
VT("/I37/SE_glo")
VT("/I37/SE_PC_glo")
VT("/I37/PC_glo")
VT("/RW")
VT("/phi1")
VT("/E")
4.0
-1.0
0.0
1.10u
1.30u
1.50u
1.70u
1.90u
2.10u
2.30u
time (s)

Read cycle

5 Electrical Characteristics

Maximum Ratings

Vcc…….….3.3V
All input and outputs w.r.t. Vss……..3.3V
Storage temperature…….....-65°C to 275°C
Ambient temperature under bias....-65°C to 275°C

Table 3: DC Characteristics

T_A = -65°C to 275°C Vcc = 2.0V to 3.3V

Parameter	Symbol	Min	Max	Units	Test conditions
High level input voltage	V_{IH}	2.0	Vcc+0.7	V	
Low level input voltage	V_{IL}	-0.5	0.8	V	
Low level output voltage	V_{OL}	-	0.4	V	I_{OL} =
High level output voltage	V_{OH}	V_{CC}-0.6	-	V	I_{OH} =
Input leakage current	I_{LI}	-100	100	nA	$\overline{CS}$ = V_{CC}, V_{IN}=GND to V_{CC}
Output leakage current	I_{LO}	-5.4	5.4	uA	$\overline{CS}$ = V_{CC}, V_{OUT}=GND to V_{CC}
Internal capacitance (all inputs and outputs)	C_{INT}	-	0.2	pF	
Operating Current	$I_{CC\ write}$	-	1.58	mA	V_{CC}=3.3V;SO=Open, F_e= 8MHz (Note)
	$I_{CC\ read}$	-	2.00	mA	V_{CC}=3.3V;SO=Open, F_e= 8MHz (Note)
Standby Current	I_{CCS}	-	0.31	mA	$\overline{CS}$ = V_{CC}

Note: This parameter is periodically sampled and not 100% tested.

Table 4: AC Characteristics

T_A = -65°C to 275°C Vcc = 2.0V to 3.3V						
Param. No.	Symbol	Parameter	Min	Max	Units	Test conditions
1	F_e	E Clock Frequency	-	8	MHz	$2.0V \le V_{CC} \le 3.3V$
2	F_{SCK}	SPI Clock Frequency	-	4	MHz	$2.0V \le V_{CC} \le 3.3V$
3	T_{CSS}	$\overline{CS}$ Setup Time	125	-	ns	$2.0V \le V_{CC} \le 3.3V$
4	T_{CSH}	$\overline{CS}$ Hold Time	250	-	ns	$2.0V \le V_{CC} \le 3.3V$
5	T_{CSD}	$\overline{CS}$ Disable Time	125	-	ns	$2.0V \le V_{CC} \le 3.3V$
6	T_{SU}	Data Setup Time	4	-	ns	$2.0V \le V_{CC} \le 3.3V$
7	T_{HD}	Data Hold Time	16	-	ns	$2.0V \le V_{CC} \le 3.3V$
8	T_R	SCK Clock Rise Time	-	2	µs	(Note)
9	T_F	SCK Clock Fall Time	-	2	µs	(Note)
10	T_{HI}	SCK Clock High Time	0.125	-	µs	$2.0V \le V_{CC} \le 3.3V$
11	T_{LO}	SCK Clock Low Time	0.125	-	µs	$2.0V \le V_{CC} \le 3.3V$
12	T_{CLD}	SCK Clock Delay Time	62.5	-	ns	$2.0V \le V_{CC} \le 3.3V$
13	T_V	Output Valid from SCK Clock Low (mode cpol = 0, cpha = 0) Output Valid from SCK Clock High (mode cpol = 1, cpha = 0)	- -	125 125	ns ns	$2.0V \le V_{CC} \le 3.3V$
14	T_{HO}	Output Hold Time	16	-	ns	
15	T_{DIS}	Output Disable Time	-	80	ns	(Note)
16	T_{WC}	Internal Write Cycle Time (byte)	-	½ Te +30	ns	$2.0V \le V_{CC} \le 3.3V$

Note:
4. This parameter is periodically sampled and not 100% tested.
5. Refer to Figure 5 and Figure 6.
6. $T_e = 1/F_e$.

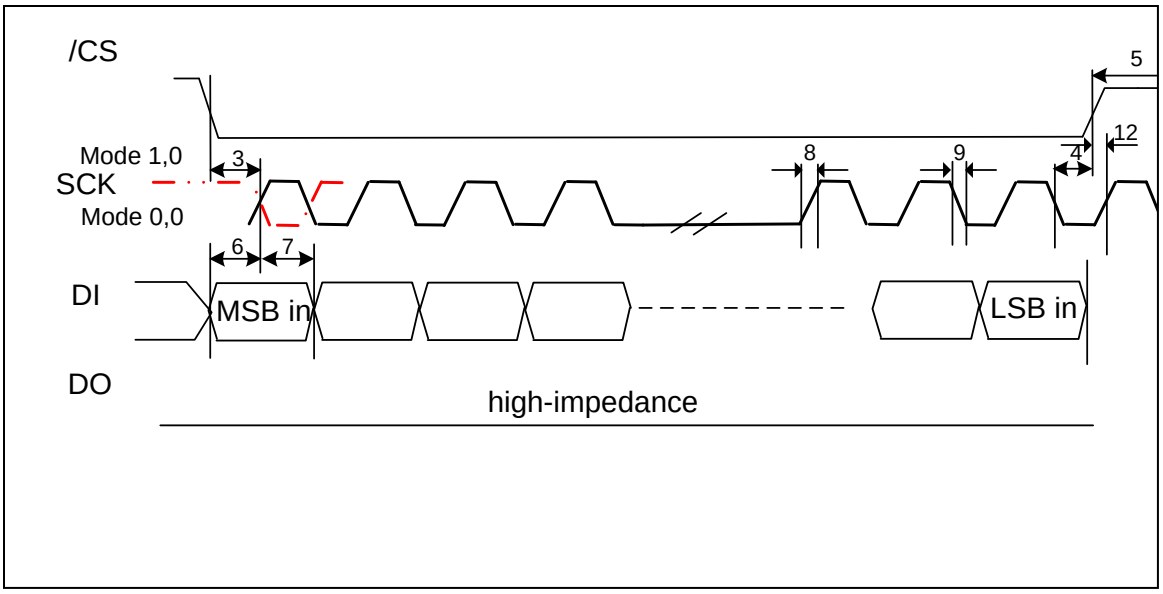


Figure 5 Serial input timing.

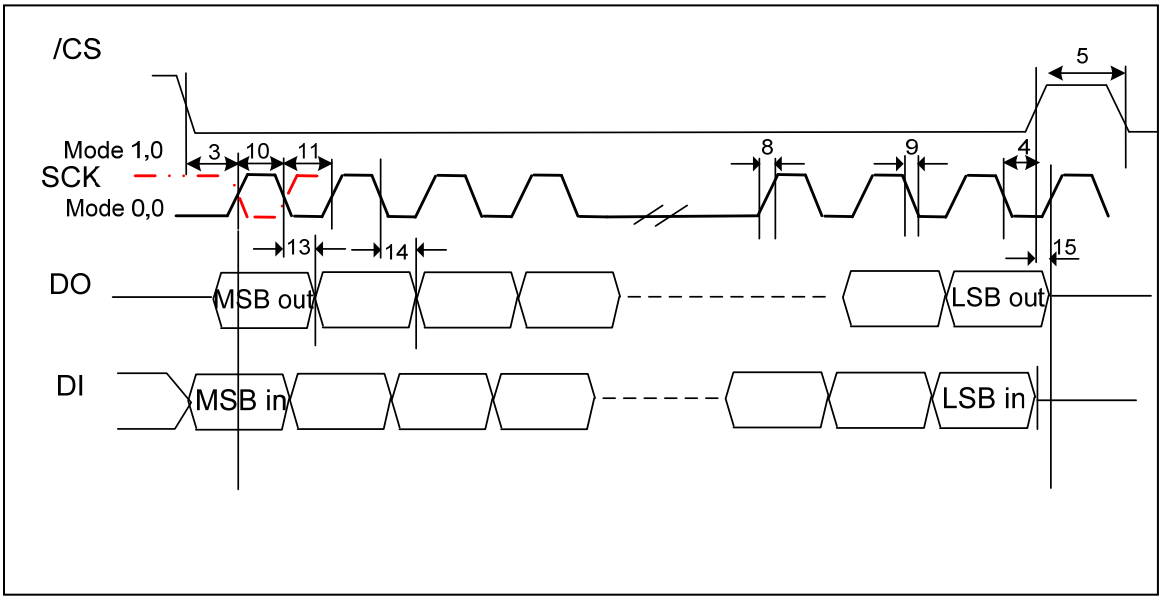


Figure 6 Serial output timing.

APPENDIX 11

PACKAGE PIN OUT DOCUMENTS

TABLE OF CONTENTS

DOCUMENT **PAGE**

1 DMS 68HC11 pads out

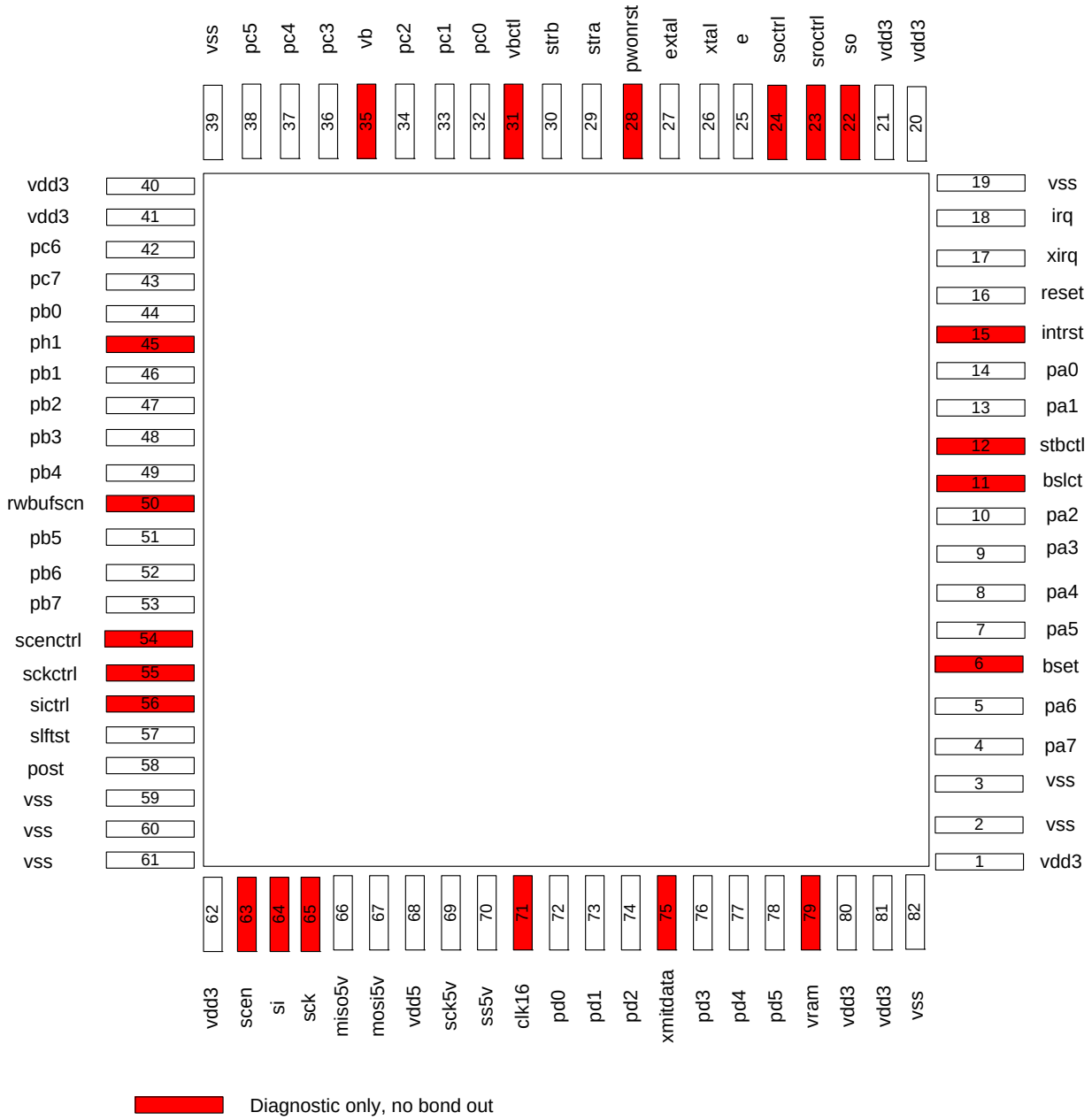


Figure 1. Pads out of the downhole microcomputer system (DMS) 68HC11 microcomputer chip.

2 4K SPI Data RAM pads out

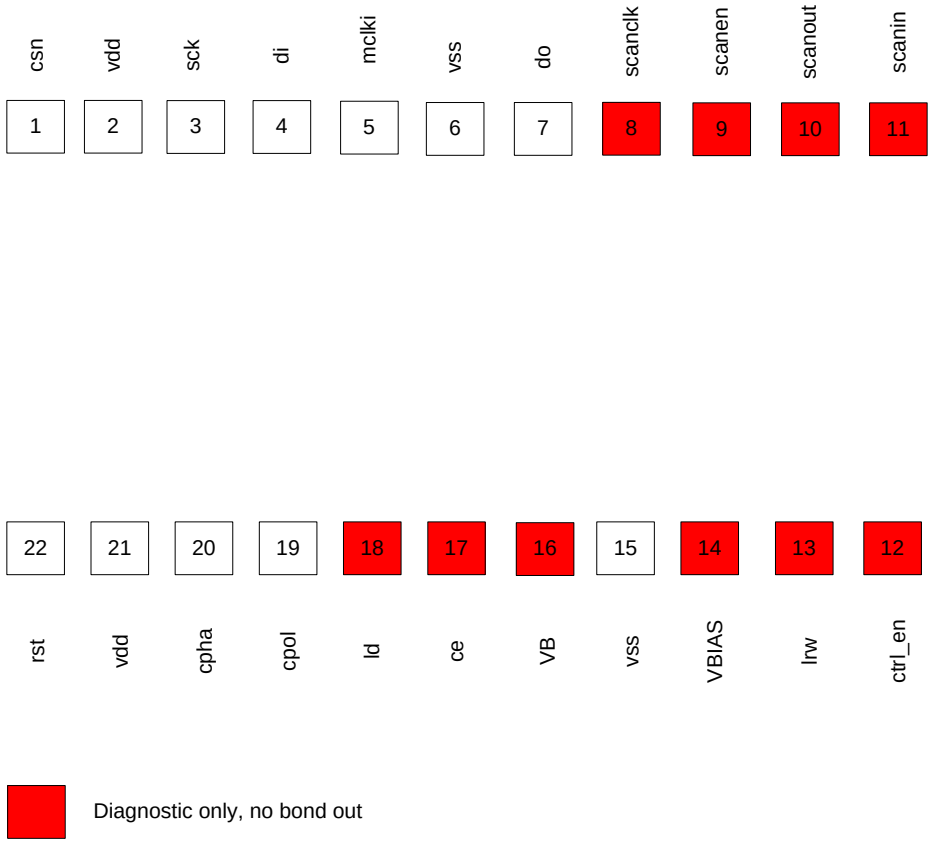


Figure 2. Pads out of the a 4K SPI Data RAM chip.

Note:
1. ctrl_en
= '0': lrw, and ce becomes output pin for observation only,
= '1': lrw, and ce becomes input pin to supply desired signal for external control.

2. scanen
= '0': Disable scan chain
= '1': Enable scan chain

3 4K SPI Mask ROM pads out

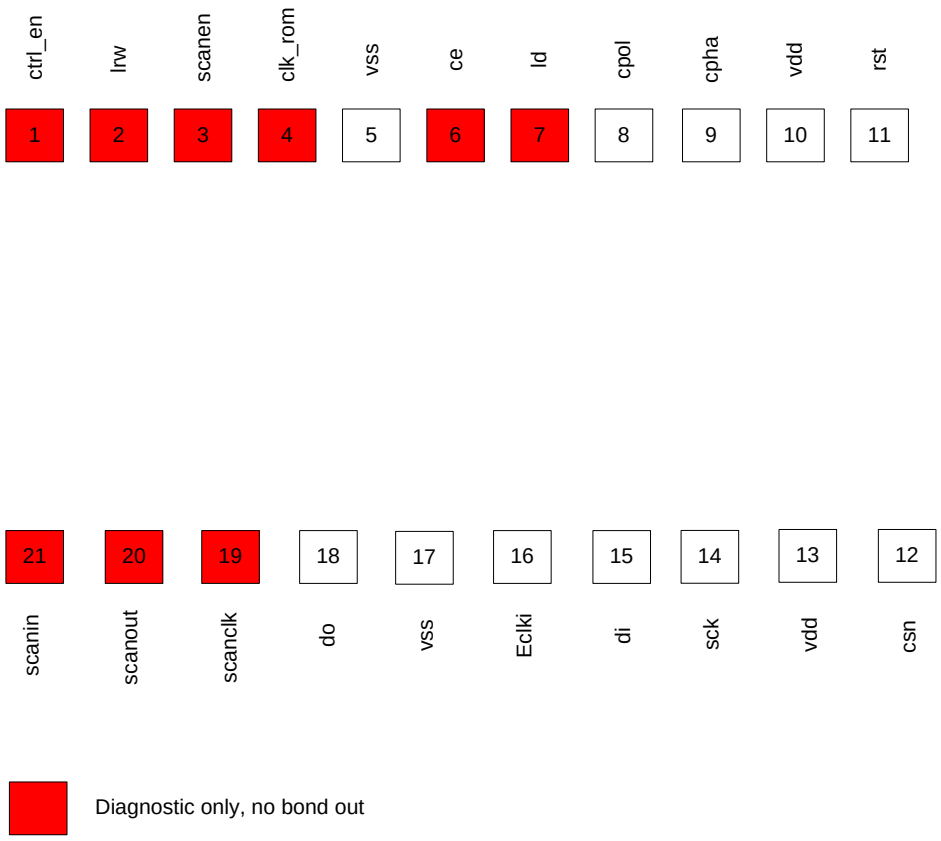


Figure 3. Pads out of the a 4K SPI MASK ROM chip.

Note:
1. ctrl_en
= '0': clk_rom, lrw, and ce becomes output pin for observation only,
= '1': clk_rom, lrw, and ce becomes input pin to supply desired signal for external control.

2. scanen
= '0': Disable scan chain
= '1': Enable scan chain

National Energy Technology Laboratory

626 Cochrans Mill Road
P.O. Box 10940
Pittsburgh, PA 15236-0940

3610 Collins Ferry Road
P.O. Box 880
Morgantown, WV 26507-0880

One West Third Street, Suite 1400
Tulsa, OK 74103-3519

1450 Queen Avenue SW
Albany, OR 97321-2198

2175 University Ave. South
Suite 201
Fairbanks, AK 99709

Visit the NETL website at:
www.netl.doe.gov

Customer Service:
1-800-553-7681