Sonne-8 Microcontroller Reference Schematics Rev. Myth

Clock Phases

PHA0: Put page index (C) and byte offset (PC) of instruction on address bus
PHA1: Latch opcode or IRQ Trap opcode into IBD-7, pre-decode instruction
PHA2: Decode instruction, set up source byte on DB or other action
PHA3: Latch results into target registers or other action
CLOSE: Finish latching etc.

Chip "Footprint" External Connections

D(OWN COUNTER) Inner-loops

Instruction Decoder

Predecode signals for early triggers

Pre-Decoders PAIR Target

Instruction-Word Latch

Start, latch instruction or interrupt vector

Interrupts disabled when "BUSY" and when running code in page 0 (See code page index reg schematics bottom right)

Pre-Decoder PAIR Source

Instruction byte:

all0: OPC_SYS	00000	xxx	See table @ SYS decoder
else: OPC_STASH	00001	xxx	See table @ STASH decoder
OPC_ALU	0001	xxxx	See table @ ALU
OPC_TRAP	001	xxxxx	B0-4: DEST PAGE
OPC_GETPUT	01 xx x xxx		B0-2 OFFS, b3:GET/PUT, B4-5:REG(GETPUT)
OPC_PAIR	1 xxx xxxx		B0-3 DST, b4-6 SRC

Hardwire IRQ TRAP instruction: 001.00000 to page 0

On instruction fetch: If IRQ and not BUSY, inject "TRAP 0" opcode, else fetch opcode from memory at C:PC

TRAP opcode triggers call to interrupt service routine in page 0, disabling further interrupts, and toggles ("BUSY")

BUSY state removed by RTI instruction

Scrounger

Remap ("scrounge") inherent NOPs such as AA, EE etc, and impracticable opcodes such as NM to other instructions

Prevent same-cycle memory-to-memory operations: Mem Read and Fwr will not execute

Remap ("scrounge") mem to accumulator and replace it

If Fx requested and NOT scrounged

Literal fetch, skip current instruction word, increment PC

Branching

Jump while loop counter register I not zero (decrements I)

Jump always

Jump if B not zero

Jump if B zero

Pair DST Decoder

Pair SRC Decoder

SYS Decoder

TRAP decoder

Immediate calls

BOPs Decoder

B0 Pointers

Opcode Type Decoder

B:3 Priority Encoder CD4532

GETPUT Decoder

(PC Program Counter)

On RET/xG/xC: Load instruction offset from O into PC

On RET: Restore PC value from O register (Center right)

On JUMP: Load target offset on DB into PC

Inc PC after each instruction cycle or literal fetch

Delay SUBR until CLOSE

CLR: Set PC to 0 during TRAP. Execution starts at offset zero, "head" of target page

Control Tree

A (Accumulator)

Zero detection

Writing to A saves old value to 2nd reg

ALU

Alternatively use two SST39SF040 NOR-Flash

BLUR3 LUT ROM layout:

16 maps @256x256

See list of instructions

ALU Instructions

0 IDA ("Identity A")	A := A
1 IDT ("Identity T")	A := T
2 OCA ("One's Complement T")	A := ~A
3 OCT ("One's Complement T")	A := ~T
4 SLA ("A shifted left")	A := A << 1
5 SLT ("T shifted left")	A := T << 1
6 SRA ("A shifted right")	A := A >> 1
7 SRT ("T shifted right")	A := T >> 1
8 AND ("A boolean-and T")	A := A & T
9 IOR ("A inclusive-or T")	A := A | T
10 IOR ("A exclusive-or T")	A := A ^ T
11 ADD ("Add T to A")	A := A + T, bits 0-7
12 CAR ("Carry bit of: A plus T")	A := 9th bit of: A + T, 0 or 1
13 AL. ("Is A less than T")	A := (A<T)? 0xFF:0
14 AE. ("Is A equal to T")	A := (A==T)? 0xFF:0
15 AG. ("Is A greater than T")	A := (A>T)? 0xFF:0

Pointer low

ADDER

On xU: B-bit signed or DB added to O. carry/overflow decrements B register ("16-bit addition")

(Page Offset Register)

Register at CLOSE xU instruction adder would feedback

B0-7 goes to PC module latched during RET

On EXIT (SUBR/xG): Save code page index in B register for return

On xU: Add signed or DB added to O. carry/overflow increments B register ("16-bit addition")

O:
- Saves offset of current instruction byte during call/trap/go
- Outputs return offset during RET
- Is the (only!) implied address offset for memory read-write, with the corresponding page index stored in B (Base)
- Together with B forms 16-bit address pointers subject to the xU instruction. The B:O pointer can be loaded from/saved into the B0-Pointers P1-4

Pointer high

On xU: B-bit signed or DB added to O. carry/overflow decrements O register ("16-bit addition")

2nd register latches at CLOSE (MB is a valid instruction)

B (Base) Page Index

B:
- Saves page index of current instruction byte during call/trap/go
- Outputs return page index during RET
- Is the implied page index for memory read-write, with the corresponding byte offset stored in O (Offset)
- Together with O forms a 16-bit address pointers subject to the xU instruction. The B:O pointer can be loaded from/saved into the B0-Pointers P1-4

{PTRB[7..0]}

{DB[7..0]}

BO Pointer Registers

Input/Output

S (SIR / SOR) Serial IO

Implements SPI functionality with dedicated SER/DES registers

E Device Enable

Two independent 4-bit groups (high/low) feed into 4x16 decoders for device selection

LOW ORDER SELECTOR

HIGH ORDER SELECTOR

P

PIR

POR

5731 Transparent Latch follows input

Parallel IO

8-bit GPIO

L (Local Page Index reg)

The L register is decremented during CALL/TRAP/ENTER and incremented during RET/LEAVE

This operations provide single-page local stack frames for nested subroutines.

Each local page has "GETPUT"-variables called L0-L7 at byte offsets 0x80-0xff.

Low order address byte Base register

Low order address byte of current instruction in local page (L0-L7, ZOPCODE bits 0b..., or GETPUT operation offset center)

{LL[7..0]}

{LO[7..0]}

Low order address byte either from ALU reg B, L register (memory read-write, call/trap control flow (call, xC etc) in the low 128 page, or from program counter register (PC) during automatic instruction fetch / Fx literal fetch.

Page index L during GETPUT ops

{HI[7..0]}

Page index = High order address byte from either B register (memory R-W, Calling code page), or 0 (data pointer).

B used during all xMx memory access. L used implicitly during GETPUT ops. C used during instruction/literal fetch, or interpage control flow

H7 switches between RAM and ROM

RAM/ROM

Enable memory writes:
- During IPH4 (instruction fetch phase)
- When writing a memory value (xM)
- If this instruction is not conflicting (scrounge)
- During PUT type GETPUT instructions (writing to local memory)

Enable memory output:
- During IPH4 (instruction fetch phases)
- When fetching a literal (Fx) or memory value (Mx).
- If these are not conflicting (scrounge)
- During GET type GETPUT instructions (reading from local memory)

B active during xM Mx data access instructions.

C active during fetch or interpage control flow

C (ODE) Page index

Code page index active when fetching instructions (IPH) or data literals (LFETCH/Fx)

Put C on the bus B will latch

Based on working prototype PCB: https://github.com/michaelmangelsdorf/sonne8
Rev. Step "Myth"
Code and schematic repo: https://github.com/michaelmangelsdorf/Myth
Author: Michael Mangelsdorf
Contact: mimgsk-schalter.de
Sheet: /
File: myth.kicad_sch

Title: Sonne-8 Micro-Controller

Size: A3	Date: 2025-06-04	Rev: Myth
KiCad 8.0.4, 9.0.1		Id: 1/1