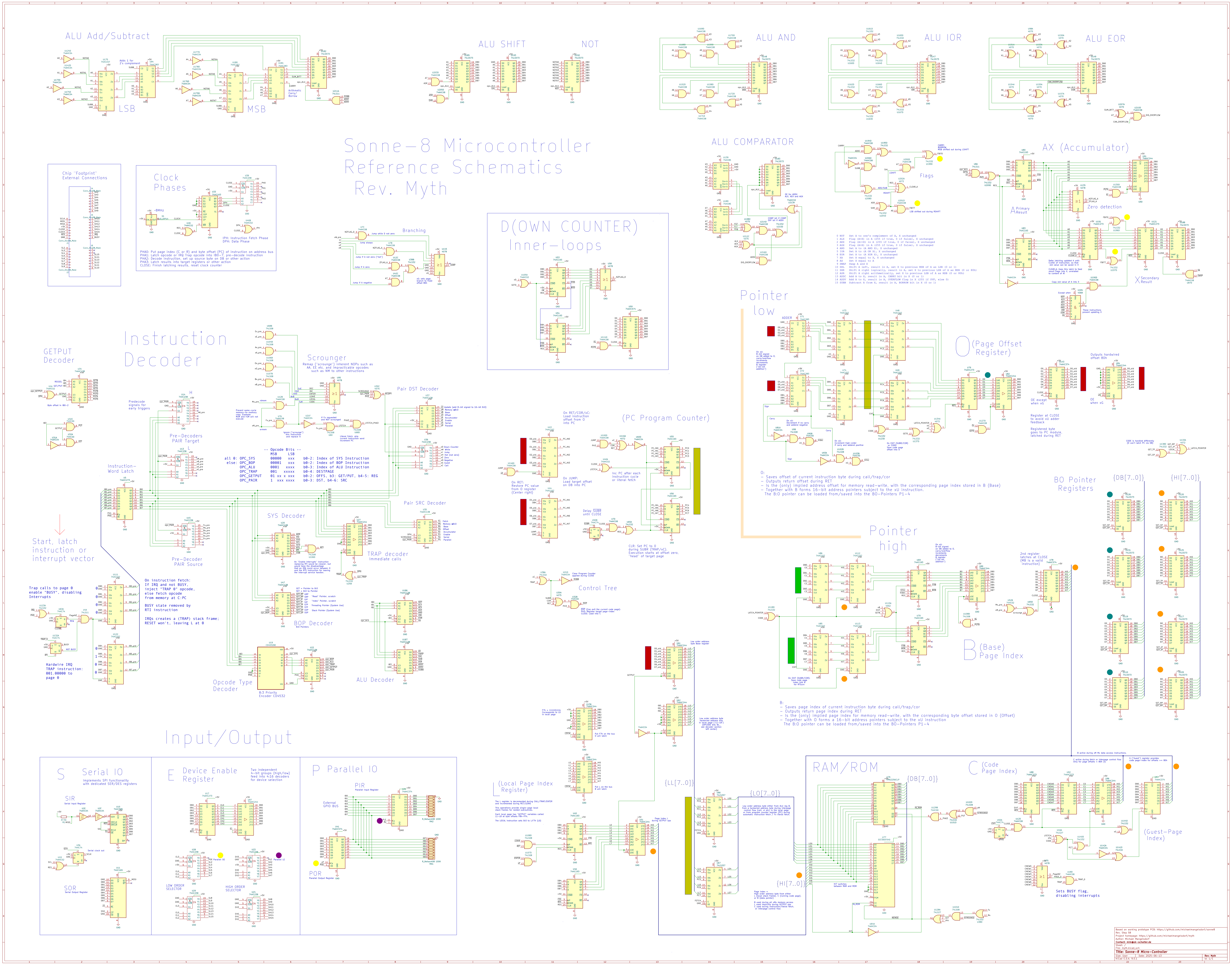
Sonne-8 Microcontroller
Reference Schematics
Rev. Myth
ALU Add/Subtract
Adds 1 for 2's complement
LSB
MSB
Arithmetic Carry/Borrow
ALU SHIFT
NOT
ALU AND
ALU IOR
ALU EOR
ALU COMPARATOR
OE for ADDV, ALX, AEX and AGX
COMP bit if COMP OVF bit if ADDV
CARRY, BORROW, MSB shifted out during LSHIFT
LSB shifted out during RSHIFT
Flags
AX (Accumulator)
Primary Result
Zero detection
Secondary Result
Defer latching updated A until CLOSE of instruction, so that old value can be saved in X
CLEAR_A: Uses this latch to feed result flags into X, unrelated to content of A!
Copy old value of A into X
Except when
These instructions prevent updating X
0 NOT Set A to one's complement of A, X unchanged
1 ALX Flag (A<X) in A (255 if true, 0 if false), X unchanged
2 AEX Flag (A==X) in A (255 if true, 0 if false), X unchanged
3 AGX Flag (A>X) in A (255 if true, 0 if false), X unchanged
4 AND Set A to (A AND X), X unchanged
5 IOR Set A to (A OR X), X unchanged
6 EOR Set A to (A XOR X), X unchanged
7 XA Set A equal to X, X unchanged
8 AX Set X equal to A
9 SWAP Swap A and X
10 SHL Shift A left, result in A, set X to previous MSB of A as LSB (0 or 1)
11 SHR Shift A right logically, result in A, set X to previous LSB of A as MSB (0 or 80h)
12 ASR Shift A right arithmetically, result in A, set X to previous LSB of A as MSB (0 or 80h)
13 ADDC Add A to X, result in A, CARRY bit in X (0 or 1)
14 ADDV Add A to X, result in A, OVERFLOW flag in X (255 if OVF, else 0)
15 SUBB Subtract A from X, result in A, BORROW bit in X (0 or 1)
Chip "Footprint" External Connections
Clock Phases
~8MHz
IPH: Instruction Fetch Phase
DPH: Data Phase
PHA0: Put page index (C or R) and byte offset (PC) of instruction on address bus
PHA1: Latch opcode or IRQ Trap opcode into IB0-7, 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 results, reset clock counter
Branching
Jump while D not zero
Jump always
Jump if A not zero ("hot")
Jump if A zero
Jump if A negative
xG: sets page, branch to fixed offset 80h
D(OWN COUNTER)
Inner-loops
Instruction Decoder
GETPUT Decoder
Byte offset in IB0-2
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 copy. Example: MxM and FxM will not execute
Ignore ("scrounge") this instruction and replace it
If Fx requested and NOT scrounged
Literal Fetch, skip current instruction word Increment PC
Predecode signals for early triggers
Pre-Decoders PAIR Target
Pre-Decoder PAIR Source
Instruction-Word Latch
-- Opcode Bits --
MSB LSB
all 0: OPC_SYS 00000 xxx b0-2: Index of SYS Instruction
else: OPC_BOP 00001 xxx b0-2: Index of BOP Instruction
OPC_ALU 0001 xxxx b0-3: Index of ALU Instruction
OPC_TRAP 001 xxxxx b0-4: DESTPAGE
OPC_GETPUT 01 xx x xxx b0-2: OFFS, b3: GET/PUT, b4-5: REG
OPC_PAIR 1 xxx xxxx b0-3: DST, b4-6: SRC
Pair DST Decoder
Pair SRC Decoder
SYS Decoder
TRAP decoder
Immediate calls
An "Enable Interrupts" instruction replacing RTI would be cleaner, but would have the disadvantage that an IRQ could occur between it and the RTS instruction for leaving the interrupt service handler.
BOP Decoder
B:O Pointers
GET = Pointer to B:O
SET = B:O to Pointer
"Read" Pointer, scratch
"Index" Pointer, scratch
Threading Pointer (System Use)
Stack Pointer (System Use)
ALU Decoder
Opcode Type Decoder
8:3 Priority Encoder CD4532
Start, latch instruction or interrupt vector
Trap calls to page 0 enable "BUSY", disabling Interrupts
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
BUSY state removed by RTI instruction
IRQs creates a (TRAP) stack frame; RESET won't, leaving L at 0
(PC Program Counter)
On RET/COR/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 SUBR (TRAP/xC). Execution starts at offset zero, "head" of target page
Control Tree
Clear Program Counter applied during CLOSE
EXIT (Can exit the current code page): PH3: Register target page-index CLOSE: Load into C
Pointer low
ADDER
On xU: 8-bit signed on DB added to O, carry/overflow increments decrements B register ("16-bit addition")
On xU: Decrement if no carry and addend negative
On xU: Increment high-order if carry and addend positive
On EXIT (SUBR/COR) or CODE: Save code page offset into O
O (Page Offset Register)
Outputs hardwired offset 80h
OE except when xG
OE when xG
Register at CLOSE to avoid xU adder feedback
Registered byte goes to PC module, latched during RET
CODE is handled differently (O can't latch PC via DB)
O:
- Saves offset of current instruction byte during call/trap/cor
- 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 BO-Pointers P1-4
BO Pointer Registers
{DB[7..0]}
{HI[7..0]}
Pointer high
2nd register latches at CLOSE (MB is a valid instruction)
On EXIT (SUBR/COR): Save code page index into B for RTSurn
B (Base) Page Index
B:
- Saves page index of current instruction byte during call/trap/cor
- Outputs return page index during RET
- Is the (only) 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 BO-Pointers P1-4
Input/Output
S Serial IO
Implements SPI functionality with dedicated SER/DES registers
SIR
Serial Input Register
Serial clock out
SOR
Serial Output Register
E Device Enable Register
Two independent 4-bit groups (high/low) feed into 4:16 decoders for device selection
LOW ORDER SELECTOR
HIGH ORDER SELECTOR
P Parallel IO
PIR
Parallel Input Register
External GPIO BUS
POR
Parallel Output Register
L (Local Page Index Register)
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 L1-L8 at byte offsets F8h-FFh.
The LOCAL instruction sets B:O to L:F7h (L0)
F7h = 11110111b Corresponds to L0 in local page
Put F7h on the bus O will latch
Put L on the bus B will latch
Page index L during GETPUT ops
Low order address byte Base register
Low order address byte hardwired address bits in local page ("L1-L8") (OPCODE bits IB... see decoder section left center)
{LL[7..0]}
{LO[7..0]}
{HI[7..0]}
Low order address byte either from ALU reg B, from a hardwired address byte during interpage control flow (call, xl etc) in the local page, or from program counter register (PC) during automatic instruction fetch / Fx literal fetch.
Page index = High order address byte from either L (local stack frame), C (running code page), or B (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
RAM/ROM
{DB[7..0]}
HI7 switches between RAM and ROM
B active during xM Mx data access instructions.
C active during fetch or interpage control flow
Only for page offsets < 80h (!)
G ("Guest") register provides code page-index for offsets >= 80h
C (Code Page Index)
(Guest-Page Index)
Sets BUSY flag, disabling interrupts
Based on working prototype PCB: https://github.com/michaelmangelsdorf/sonne8
Rev. Step 58
Project homepage: https://github.com/michaelmangelsdorf/myth
Author: Michael Mangelsdorf
Contact: mim@ok-schalter.de
Sheet: /
File: myth.kicad_sch
Title: Sonne-8 Micro-Controller
Size: User Date: 2025-06-13 Rev: Myth
KiCad E.D.A. 9.0.1 Id: 1/1