AL8051S 8-BIT MICROCONTROLLER

Application Notes

Rev. [6/2012]

6-14-2012

Table of Contents

GENERAL INFORMATION 3
FEATURES 3
  Key features 3
  Design features 3
INTERFACE 4
  Symbol 4
  Signal description 5
BLOCK DIAGRAM 7
IMPLEMENTATION DATA 9
  Performance 9
  Memory Access Timing 10
  Instruction Cycles 12

GENERAL INFORMATION

The AL8051S Application Notes contains description of the AL8051S core architecture to explain its proper use.

AL8051S soft core is instruction set compatible with the 8051/52 8-bit microcontroller architecture and can achieve average performance of up to 20 million instructions per second in today's FPGA circuit technologies.

FEATURES

Key features

Software compatible with Intel 8051/51,

Up to six times faster than the original implementation by the equal clock frequency,

Up to 16M bytes of external Data Memory,

256 bytes of internal Data Memory,

- Up to 64K bytes of on chip Program Memory[1] ,
- 2-cycle multiplication,
- 4-cycle division,
- Up to three 16-bit timer/counters ,
- Full-duplex serial port,
- Support for External SFRs,
- Structure optimized for Xilinx Virtex™, SpartanII™ FPGA devices.

Design features

- External Special Function Register:

  Up to 84 External Special Function Registers (ESFRs) may be added to the AL8051/52S design. ESFRs are memory mapped into Data Memory space between addresses 80 hex and FF hex in the same manner as core SFRs and may occupy any address that is not occupied by a core SFR, i.e. except the following address spaces 80h..90h,98h,99h,a0h, a8h,b0h, b8h, c0h..d0h,e0h,f0h . ESFR should be connected by signals: SFRDATAI, SFRDATAO, SFRWE, SFROE, SFRADDRS, SFRADDRD (Table 1).
- External RAM

  The AL8051S can address up to 16M bytes of external Data RAM via the interconnect signals MEMADDR ,MEMDATAI, MEMDATAO, MEMRD, MEMWR (Table 1). Data Page Pointer is located at address 84h, and select 1 of 256 pages (64kB each). After reset first memory page is selected (DPP=00h)

[1] Program memory must be located inside FPGA or ASIC device because of timing requirements.

- Stretch memory cycle register

Stretch memory cycle register is located at address 85h. The writing 3-bit code in this register adjusts external RAM access time (MEMRD, MEMWR (Table 1) pulse width is between 1 – 8 clock cycles).

INTERFACE

Symbol

Fig.1 shows AL8051S core symbol.

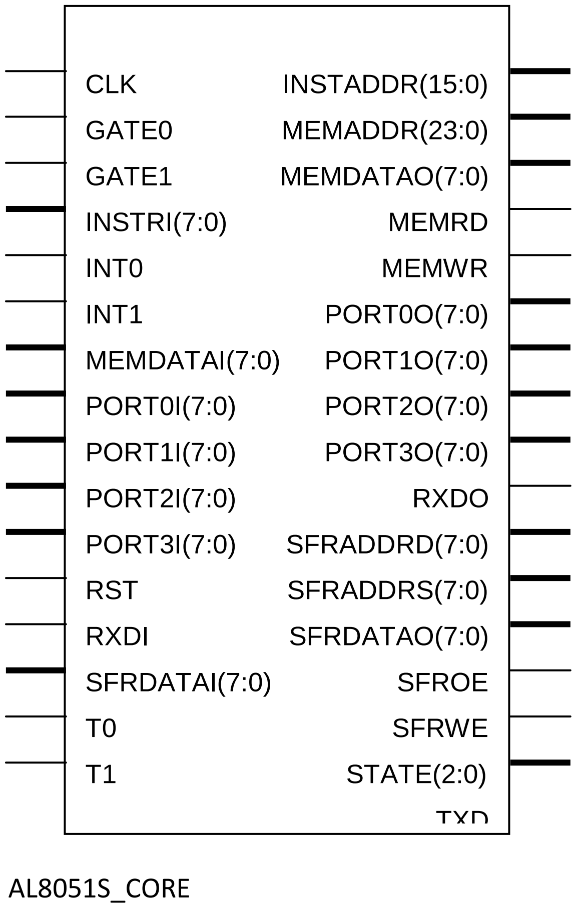

Figure 1. AL8051S symbol.

Signal description

The descriptions of the core signals are represented in the table 1.

SIGNAL	TYPE	DESCRIPTION
CLK	input	Global clock
GATE0	input	Timer 0 gate input
GATE1	input	Timer 1 gate input
INT0	input	External interrupt 0
INT1	input	External interrupt 1
PORT0I[7:0]	input	Port 0 input
PORT1I[7:0]	input	Port 1 input
PORT2I[7:0]	input	Port 2 input
PORT3I[7:0]	input	Port 3 input
INSTRI[7:0]	input	Data bus from program memory
RST	input	Global reset
RXDI	input	Serial receiver input
SFRDATAI[7:0]	input	Data bus from user SFR's
T0	input	Timer 0 input
T1	input	Timer 1 input
MEMDATAI[7:0]	input	Data bus from external data memory
INSTADDR[15:0]	output	Instruction address bus
PORT0O[7:0]	output	Port 0 output
PORT1O[7:0]	output	Port 1 output
PORT2O[7:0]	output	Port 2 output
PORT3O[7:0]	output	Port 3 output
TXD	output	Serial transmitter output
RXDO	output	Serial receiver output
SFRADDRS[7:0]	output	RAM and SFR's source address bus
SFRADDRD[7:0]	output	RAM and SFR's destination address bus
SFRDATAO[7:0]	output	Data bus to user SFR's
SFROE	output	User SFR's read
SFRWE	output	User SFR's write enable
MEMADDR[23:0]	output	External data memory address bus
MEMDATAO[7:0]	output	Data bus to external data memory
MEMWR	output	External data memory write
MEMRD	output	External data memory read
STATE[2]	output	When high, indicates an IRQ accepted.
STATE[1]	output	When high, indicates a data is fetching into the instruction queue and the Program Counter is going to change.
STATE[0]	output	When high, indicates the start of a new instruction execution in the next clock cycle.

Table 1. AL8051S core signal description.

Typical Core Interconnection

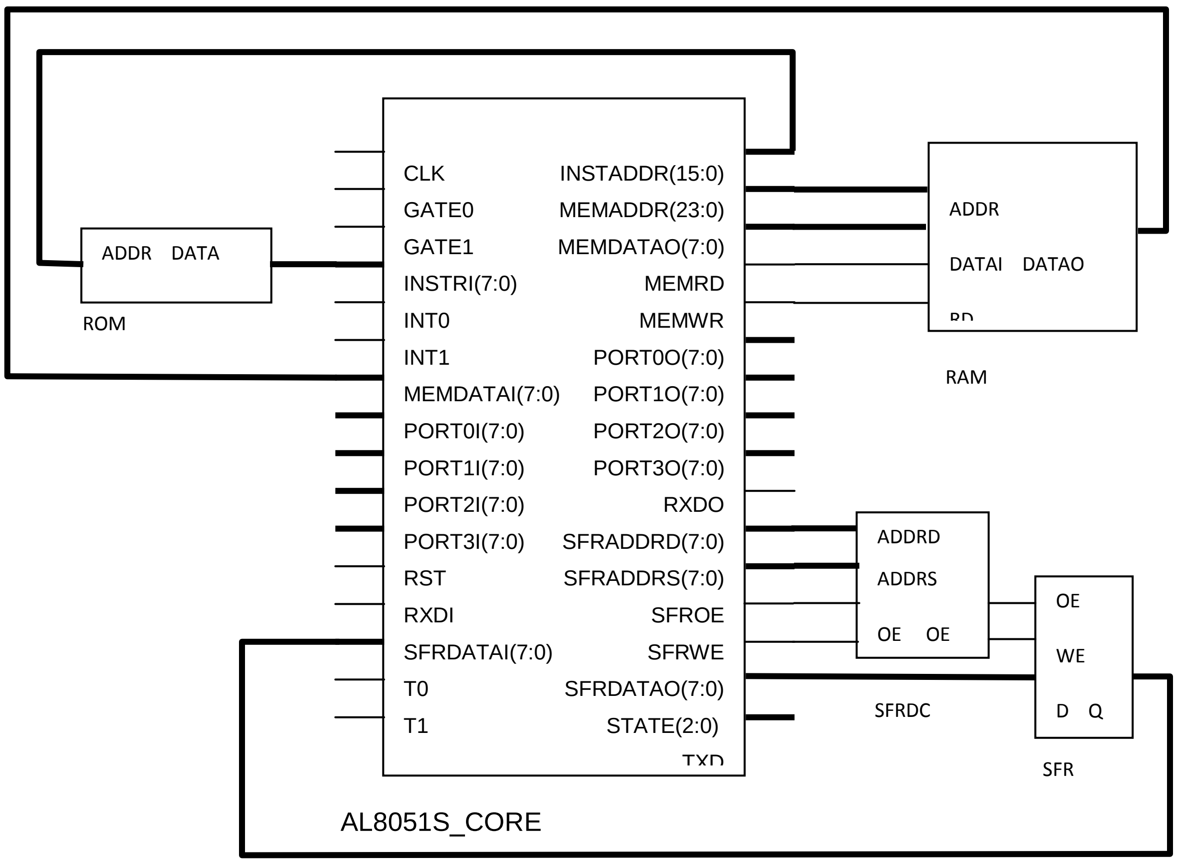

Figure 2 Core interconnection

Typical core interconnection is shown on figure 2.

Components:

- AL8051S_CORE - microcontroller core
- RAM - external data memory, asynchronous RAM up to 16 MB
- ROM - program memory up to 64kB
- SFRDC – decoder of the user SFR address
- SFR – user special function register

BLOCK DIAGRAM

The basic block diagram of the microcontroller core is shown in the fig.4.

The AL8051S core has five main blocks: Arithmetic-Logic Unit (ALU) U_A, Block of Data RAM (BLOCK_DRAM) U_DM, Block of Interrupt control (BLOCK_INTERRUPT) U_I, Program Counter block (PC) U_PC, and instruction and data buffer (BUF) U_BUF.

All the blocks are connected with:

1. common instruction bus INSTR,
2. two bidirectional data busses: DATA_BUS0 and DATA_BUS1,
3. other respective busses and lines.

ALU implements all the arithmetic and logic functions, bit functions and testing jump conditions. The operands and results are moved between ALU and BLOCK_DRAM through DATA_BUS0 and DATA_BUS1. The data in Accumulator register are moved to PC and to the external data memory through ACC_BUS and MEMDATA busses. The calculated jump condition is sent to PC by BLOCK_JMP line.

BLOCK_DRAM contains two port synchronous RAM implemented as BlockSelect RAM with the volume of 256 bytes. Due to the two port RAM all the instructions are implemented no more than 2, 3, or 4 clock cycles, including such time consumable instructions like LCALL, RET. BLOCK_DRAM has its own control unit, which controls most of MOV – type instruction calculation, and fetching-storing the data on both DATA_BUS0 and DATA_BUS1. The data from selected index register is put in the RI_OUT bus and the two bytes of the stack top are put in the STACK_OUT bus. BLOCK_DRAM generates output signals MEMWR, and MEMRD.

The generated signal in the line N_INSTR is active high when the byte on the INSTR bus is the first byte of the instruction, another words, it signs a new instruction. Therefore, it comes in all the blocks and enables the instruction code registering.

When signal on the line S_INTERRUPT is high then BLOCK_DRAM recognizes the interrupt vectoring, and the 00h code is set to the DATA_BUS busses, and the PC data is stored in the stack space.

PC block controls the program flow. It calculates the next instruction address using the data from program counter, instruction bytes, DPTR register on the DPTR_BUS and accumulator on the ACC_BUS. This address is outputted to the external program memory through the INSTADDR bus. When CALL-type instruction is implemented, the return address is sent to BLOCK_DRAM trough ADDRINSTR bus. However when RET or RETI instructions are implemented, this address is returned into PC through the STACK_OUT bus.

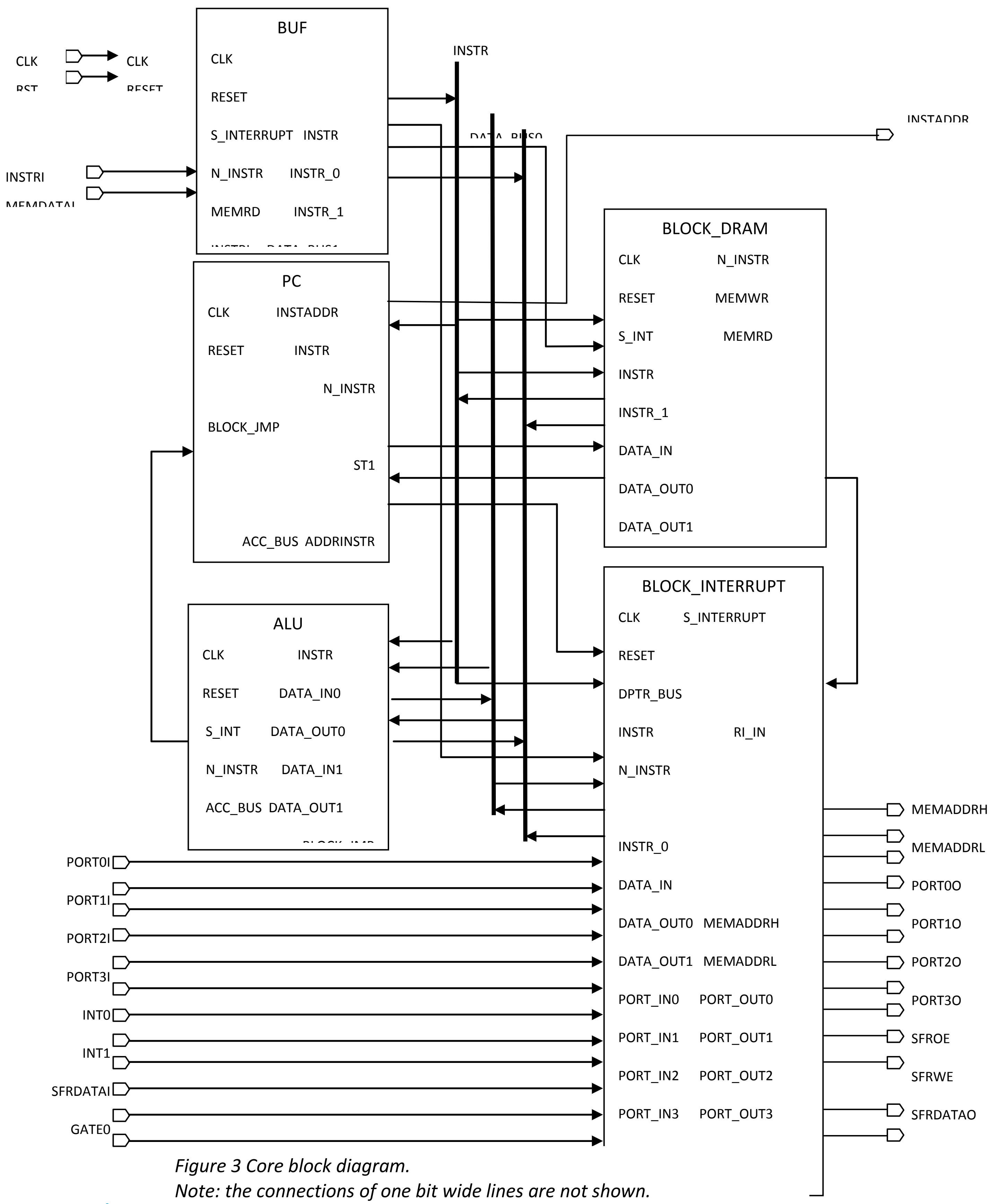

Figure 3 Core block diagram.
Note: the connections of one bit wide lines are not shown.

BLOCK_INTERRUPT makes interrupt signal latching, decoding and polling. It contains most of special function registers: DPH, DPL, PORT0, PORT1, PORT2, PORT3, IP, IE, TCON, TMOD, TH0, TL0, TH1, TL1, SCON, SBUF, PCON, as well as Timer0, Timer1, and UART. The signal of interrupt on the line S_INTERRUPT is sent to BLOCK_DRAM, ALU, and BUF. The data in the DPTR is usually set in the DPTR_BUS, but when interrupt occurs, the interrupt vector is set on this bus.

This block is connected with the inputs and outputs of the core by the busses and lines: INT0, INT1, GATE0, GATE1, T0, T1, TXD, RXDI, RXDO, SFRWE, SFROE, SFRADDRS, SFRADDRD, SFRDATAI, SFRDATAO, PORT0I, PORT0O, PORT1I, PORT1O, PORT2I, PORT2O, PORT3I, PORT3O. Outputs MEMADDRH and MEMADDRL form 24-bit wide bus MEMADDR.

BUF transfers the data from INSTRI, and MEMDATAI core inputs to INSTR, INSTR_0, INSTR_1, and DATA_BUS1 busses respectively. When interrupt occurs, the code of the instruction JMP @A+DPTR is set on the INSTR bus, and the call of the interrupt routine is implemented.

Lines S_INTERRUPT, ST1, and NINSTR form the STATE output, which helps to monitor the states of the microcontroller when running the program.

IMPLEMENTATION DATA

Performance

The following table illustrates the AL8051S core performance in Xilinx VIRTEX™ device.

Core configuration	Core without timers and UART	Core with 3 timers and UART
Target device	XCV300BG352-6	XCV300BG352-6
Select Memory	1 BlockSelect RAM	1 BlockSelect RAM
Area	933 Slices (30%)	1261 Slices (41%)
System clock fmax	59MHz	56MHz

Table 2. Implementation Data – Xilinx VIRTEX

Memory Access Timing

External Memory Access Timing when read and write operations is illustrated on Fig.4, and Fig.5, Table 3, and Table 4.

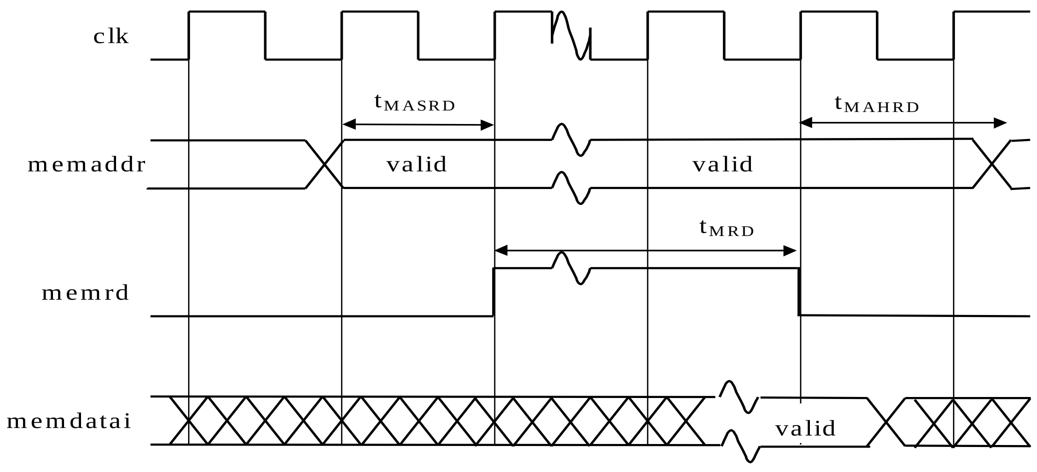

Figure 4.. External RAM read cycle.

	min	Max	Comments
t_{MRD}	1 clk period	8 clk period	**Lenght** of read pulse.
t_{MASRD}	1 clk period	-	RAM address **setup** time before read pulse
t_{MAHRD}	1 clk period	-	RAM address **hold** time after read pulse.

Table 3. Memory access timing when read operation.

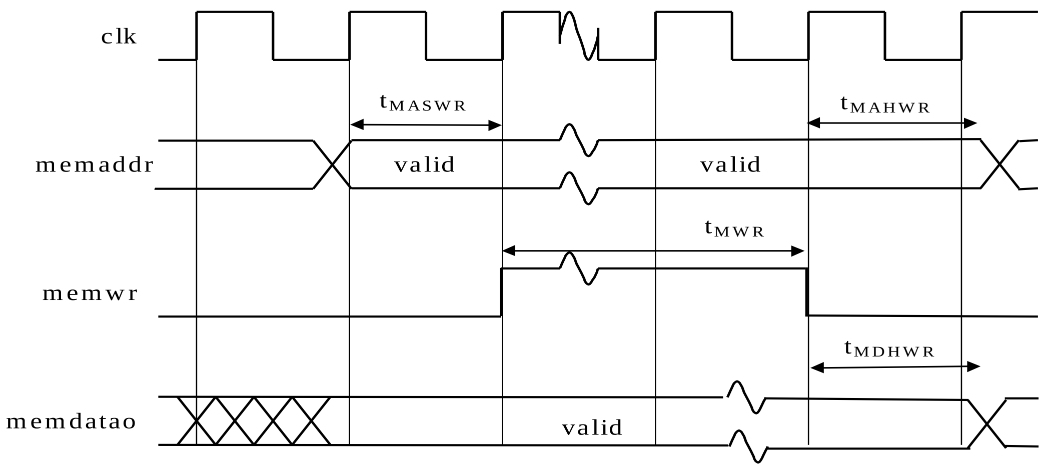

Figure 5. External RAM write cycle.

	min	Max	Comments
t_{MWR}	1clk period	8 clk period	**Length** of RAM write pulse.
t_{MASWR}	1 clk period	-	RAM address **setup** time before write pulse.
t_{MAHWR}	1 clk period	-	RAM address **hold** time after write pulse.
t_{MDHWR}	1 clk period	-	RAM data **hold** time after write pulse.

Table 4. Memory access timing when read operation.

Program Memory Access Timing is shown on Fig.6 and Table 5.

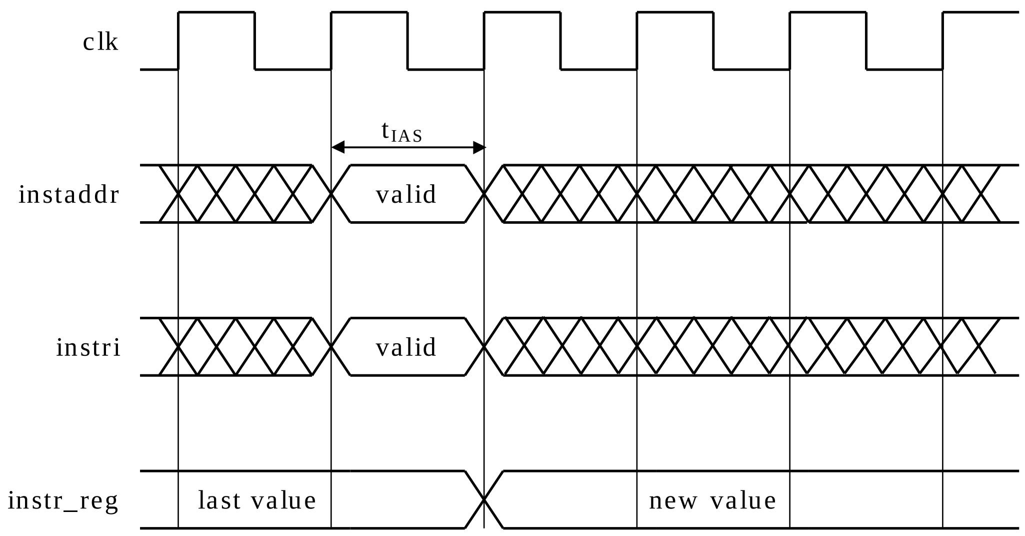

Figure 6. Program memory read cycle.

	min	max	Comments
t_{IAS}	1 clk period	-	Instruction address **setup** time before FETCH or MOVC.

Table 5. Memory access timing when read the instruction.

Instruction Cycles

The following tables give information about the instruction cycles of the AL8051S core. Table 7 and Table 8 contain notes for mnemonics used in Instruction set tables. Tables 9-13 show instruction hexadecimal codes, number of bytes and clock periods for each instruction.

Rn	Working register R0-R7
Direct	128 internal RAM locations, Special Function Registers
@Ri	Indirect internal or external RAM location addressed by index register R0 or R1
#data	8-bit constant included in instruction
#data 16	16-bit constant included as bytes 2 and 3 of instruction
Bit	256 software flags, any bit-addressable I/O pin, control or status bit
A	Accumulator

Table 7. Mnemonics on data addressing modes

Addr16	Destination address for LCALL and LJMP may be anywhere within the 64-Kbyte of program memory address space.
Addr11	Destination address for ACALL and AJMP will be within the same 2-Kbyte page of program memory as the first byte of the following instruction.
Rel	SJMP and all conditional jumps include an 8-bit offset byte. Range is +127/-128 bytes relative to the first byte of the following instruction

Table 8. Mnemonics on program addressing modes

Mnemonic	*Description*	**Code**	**Bytes**	*Clk periods*
ADD A,Rn	Add register to accumulator	28-2F	1	3
ADD A,direct	Add direct byte to accumulator	25	2	3
ADD A,@Ri	Add indirect RAM to accumulator	26-27	1	3
ADD A,#data	Add immediate data to accumulator	24	2	3
ADDC A,Rn	Add register to accumulator with carry flag	38-3F	1	3
ADDC A,direct	Add direct byte to A with carry flag	35	2	3
ADDC A,@Ri	Add indirect RAM to A with carry flag	36-37	1	3
ADDC A,#data	Add immediate data to A with carry flag	34	2	3
SUBB A,Rn	Subtract register from A with borrow	98-9F	1	3
SUBB A,direct	Subtract direct byte from A with borrow	95	2	3
SUBB A,@Ri	Subtract indirect RAM from A with borrow	96-97	1	3
SUBB A,#data	Subtract immediate data from A with borrow	94	2	3
INC A	Increment accumulator	04	1	3
INC Rn	Increment register	08-0F	1	3
INC direct	Increment direct byte	05	2	3
INC @Ri	Increment indirect RAM	06-07	1	3
DEC A	Decrement accumulator	14	1	3
DEC Rn	Decrement register	18-1F	1	3
DEC direct	Decrement direct byte	15	1	3

DEC @Ri	Decrement indirect RAM	16-17	2	3
INC DPTR	Increment data pointer	A3	1	2
MUL A,B	**Multiply A and B**	**A4**	**1**	**2**
DIV A,B	**Divide A by B**	**84**	**1**	**4**
DA A	Decimal adjust accumulator	D4	1	2

Table 9. Arithmetic operation instructions

Mnemonic	**Description**	**Code**	**Bytes**	*Clk periods*
ANL A,Rn	AND register to accumulator	58-5F	1	3
ANL A,direct	AND direct byte to accumulator	55	2	3
ANL A,@Ri	AND indirect RAM to accumulator	56-57	1	3
ANL A,#data	AND immediate data to accumulator	54	2	3
ANL direct,A	AND accumulator to direct byte	52	2	3
ANL direct,#data	AND immediate data to direct byte	53	3	3
ORL A,Rn	OR register to accumulator	48-4F	1	3
ORL A,direct	OR direct byte to accumulator	45	2	3
ORL A,@Ri	OR indirect RAM to accumulator	46-47	1	3
ORL A,#data	OR immediate data to accumulator	44	2	3
ORL direct,A	OR accumulator to direct byte	42	2	3
ORL direct,#data	OR immediate data to direct byte	43	3	3
XRL A,Rn	Exclusive OR register to accumulator	68-6F	1	3
XRL A,direct	Exclusive OR direct byte to accumulator	65	2	3
XRL A,@Ri	Exclusive OR indirect RAM to accumulator	66-67	1	3
XRL A,#data	Exclusive OR immediate data to accumulator	64	2	3
XRL direct,A	Exclusive OR accumulator to direct byte	62	2	3
XRL direct,#data	Exclusive OR immediate data to direct byte	63	3	3
CLR A	Clear accumulator	E4	1	2
CPL A	Complement accumulator	F4	1	2
RL A	Rotate accumulator left	23	1	2
RLC A	Rotate accumulator left through carry	33	1	2
RR A	Rotate accumulator right	03	1	2
RRC A	Rotate accumulator right through carry	13	1	2
SWAP A	Swap nibbles within the accumulator	C4	1	2

Table 10. Logic operation instructions

Mnemonic	Description	*Code*	Bytes	Clk periods
MOV A,Rn	Move register to accumulator	E8-EF	1	2
MOV A,direct	Move direct byte to accumulator	E5	2	2
MOV A,@Ri	Move indirect RAM to accumulator	E6-E7	1	2
MOV A,#data	Move immediate data to accumulator	74	2	2
MOV Rn,A	Move accumulator to register	F8-FF	1	2
MOV Rn,direct	Move direct byte to register	A8-AF	2	2
MOV Rn,#data	Move immediate data to register	78-7F	2	2
MOV direct,A	Move accumulator to direct byte	F5	2	2
MOV direct,Rn	Move register to direct byte	88-8F	2	2
MOV direct1,direct2	Move direct byte to direct byte	85	3	3
MOV direct,@Ri	Move indirect RAM to direct byte	86-87	2	3
MOV direct,#data	Move immediate data to direct byte	75	3	3
MOV @Ri,A	Move accumulator to indirect RAM	F6-F7	1	2
MOV @Ri,direct	Move direct byte to indirect RAM	A6-A7	2	2
MOV @Ri,#data	Move immediate data to indirect RAM	76-77	2	2
MOV DPTR,#data16	Load data pointer with a 16-bit constant	90	3	3
MOVC A,@A+DPTR	Move code byte relative to DPTR to accumulator	93	1	4
MOVC A,@A+PC	Move code byte relative to PC to accumulator	83	1	4
MOVX A,@Ri	Move external RAM (8-bit address) to A	E2-E3	1	3*
MOVX A,@DPTR	Move external RAM (16-bit address) to A	E0	1	3*
MOVX @Ri,A	Move A to external RAM (8-bit address)	F2-F3	1	3*
MOVX @DPTR,A	Move A to external RAM (16-bit address)	F0	1	3*
PUSH direct	Push direct byte onto stack	C0	2	2
POP direct	Pop direct byte from stack	D0	2	2
XCH A,Rn	Exchange register with accumulator	C8-CF	1	2
XCH A,direct	Exchange direct byte with accumulator	C5	2	2
XCH A,@Ri	Exchange indirect RAM with accumulator	C6-C7	1	2
XCHD A,@Ri	Exchange low-order nibble indirect RAM with A	D6-D7	1	2

Table 11. Data transfer instructions

* MOVX cycle count depends on STRETCH register content.

Mnemonic	**Description**	*Code*	**Bytes**	**Clk periods**
ACALL addr11	Absolute subroutine call	11-F1	2	3
LCALL addr16	Long subroutine call	03	3	4
RET	Return from subroutine	22	1	3
RETI	Return from interrupt	32	1	3
AJMP addr11	Absolute jump	01-E1	2	3
LJMP addr16	Long jump	02	3	4
SJMP rel	Short jump (relative address)	80	2	3
JMP @A+DPTR	Jump indirect relative to the DPTR	73	1	3
JZ rel	Jump if accumulator is zero	60	2	4
JNZ rel	Jump if accumulator is not zero	70	2	4
JC rel	Jump if carry flag is set	40	2	4
JNC	Jump if carry flag is not set	50	2	4
JB bit,rel	Jump if direct bit is set	20	3	4
JNB bit,rel	Jump if direct bit is not set	30	3	4
JBC bit,direct rel	Jump if direct bit is set and clear bit	10	3	4
CJNE A,direct rel	Compare direct byte to A and jump if not equal	B5	3	4
CJNE A,#data rel	Compare immediate to A and jump if not equal	B4	3	4
CJNE Rn,#data rel	Compare immediate to reg. and jump if not	B8-BF	3	4
CJNE @Ri,#data rel	Compare immediate to ind. and jump if not equal	B6-B7	3	4
DJNZ Rn,rel	Decrement register and jump if not zero	D8-DF	2	4
DJNZ direct,rel	Decrement direct byte and jump if not zero	D5	3	4
NOP	No operation	00	1	2

Table 12. Program jump instructions

Mnemonic	Description	Code	Bytes	Clk periods
CLR C	Clear carry flag	C3	1	2
CLR bit	Clear direct bit	C2	2	3
SETB C	Set carry flag	D3	1	2
SETB bit	Set direct bit	D2	2	3
CPL C	Complement carry flag	B3	1	2
CPL bit	Complement direct bit	B2	2	3
ANL C,bit	AND direct bit to carry flag	82	2	3
ANL C,/bit	AND complement of direct bit to carry	B0	2	3
ORL C,bit	OR direct bit to carry flag	72	2	3
ORL C,/bit	OR complement of direct bit to carry	A0	2	3
MOV C,bit	Move direct bit to carry flag	A2	2	3
MOV bit,C	Move carry flag to direct bit	92	2	3

Table 13. Boolean manipulation instructions