Design of an enhanced 8051 microcontroller core architecture in IDaSS, synthesised with ASA

P.W.H. Heuts

Coach : Prof.ir. M.P.J. Stevens
Supervisor : Prof.ir. M.P.J. Stevens
Period : January 1994 - August 1994
Abstract

This master's thesis describes the upgrading of the frequently used 8051 microcontroller. The address range of the 8051 core has been extended from 64 Kbytes to 16 Mbytes. The stack depth has been increased from 256 bytes to 64 Kbytes. To handle these extensions, the instruction set of enhanced 8051 core has been adapted, but is still compatible to the standard 8051 instruction set. Furthermore, the number of clock cycles needed to execute an instruction is reduced considerably.

The Section of Digital Information Systems of the Eindhoven University of Technology developed an interactive design and simulation tool, called IDaSS. IDaSS is a very suitable design environment for complex hardware. The IDaSS library contains re-usable components, like processor cores and input/output interfaces. Interconnection of the library elements imposes strict rules for the interfaces. The IDaSS design of the enhanced 8051 core architecture satisfies these specifications. Since the enhanced 8051 core ought to be suitable for synthesising with the ASA Silicon compiler, restrictions of the 'idasstosid' converter and limits of ASA have been taken into account while designing the architecture.

The CMOS 1.0 micron process technology is used in ASA. The synthesised enhanced 8051 core consumes 28 mm² of die area. The maximum clock frequency is 7.5 MHz. Although the clock frequency is relatively low, the enhanced 8051 core executes instructions four times faster than the standard 8051. Further improvement of the enhanced 8051 core must be done to increase the processor speed and to decrease the amount of die area. Redesigning the ALU, stretching out several instruction executions and manually making a data pad layout of the register RAM are recommended actions.
Contents

1 Introduction .................................................. 1

2 Overview of the 8051 microcontroller ............................................. 2
  2.1 Memory Organization ........................................... 3
  2.2 Special Function Registers in the Core ....................................... 3
  2.3 Addressing Modes .................................................. 5
  2.4 Boolean Processor .................................................. 5
  2.5 Instruction Set .................................................... 5
  2.6 CPU Timing ....................................................... 6

3 Upgrading the Standard 8051 Core .................................................. 7

4 Core Interfaces ...................................................... 8
  4.1 Memory Interface .................................................. 8
  4.2 Input/Output Interface ............................................... 9
  4.3 Interrupt Interface .................................................. 9

5 Enhanced 8051 Core Design for ASA .............................................. 11
  5.1 State Controller .................................................... 12
  5.2 External Memory Interface ............................................. 13
  5.3 Internal RAM ........................................................ 14
  5.4 Direct Addressing Controller .......................................... 16
  5.5 Arithmetic and Logical Unit .......................................... 17
  5.6 Program Counter .................................................... 19
  5.7 ALU Related Registers ............................................... 21
  5.8 Stack Pointer ...................................................... 22
  5.9 Data Pointer ...................................................... 23

6 ASA synthesis ........................................................ 24

7 Conclusions ........................................................... 26

References .............................................................. 28

Appendix A: MCS-51 Instruction Set ............................................. 29

Appendix B: Enhanced 8051 Instruction Set .................................... 30
<table>
<thead>
<tr>
<th>Appendix C: Instruction Execution Time Specification</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>Appendix D: First IDaSS Design</td>
<td>36</td>
</tr>
<tr>
<td>Appendix E: IDaSS Document of the Enhanced 8051 Core</td>
<td>39</td>
</tr>
</tbody>
</table>
1 Introduction

This report presents the results of my master's thesis. This final project is executed for the Section of Digital Information Systems, Department of Electrical Engineering of the Eindhoven University of Technology. The Section of Digital Information Systems has developed an Interactive Design and Simulation System, abbreviated as IDaSS [2]. IDaSS is a design environment for complex synchronous (and in near future asynchronous) hardware. The design is described as a tree-like hierarchy of schematics. The schematics are entered graphically and represent elements like registers, ALUs, memories, and state controllers. The IDaSS designs can be converted into an operational ASIC, using the 'idasstosid' and ASA tools provided by Sagantec. Complete automatic synthesis is not yet possible, because the conversion program 'idasstosid' cannot convert all the IDaSS functions and is not free of bugs. Furthermore, the ASA silicon compiler has specific limitations.

The target of the master's thesis was: design an upgrade of the 8051 microcontroller core architecture in IDaSS that can be synthesised with ASA. The upgrades consist of a linear address range extension to 16 Mbyte and a stack extension to 64 Kbyte. The instruction set is adapted to handle the extended address range, but the assembly source code is compatible with the standard 8051 instruction set.

The next section of this report gives a short overview of the standard 8051 microcontroller features. The enhancements of the standard 8051 are described in section three. The fourth section specifies the standard memory, input/output, and interrupt interfaces. The interfaces provide an easy connection of the enhanced 8051 microcontroller core with components present in the IDaSS library. In section five the architecture of the enhanced 8051 microcontroller core is clarified. The synthesis of the design in ASA is described in the sixth section. Finally, section seven concludes this work.
2 Overview of the 8051 microcontroller

A short overview of the 8051 microcontroller is given in this section. The 8051 is a member of INTEL's MCS-51 8-bit microcontroller family [1]. The MCS-51 architectural block diagram is displayed in figure 1. The major microcontroller features are:

- an 8-bit central processing unit;
- a Boolean processor;
- 64 Kbyte address space for the external program memory;
- 64 Kbyte address space for the external data memory;
- 32 I/O lines;
- multiplexed address and data lines;
- a full duplex serial port;
- a six-source interrupt structure with two priority levels;
- three 16-bit timer/counters.

![MCS-51 architectural block diagram](image-url)

*Figure 1: MCS-51 architectural block diagram.*
2.1 Memory Organization

The 8051 has separate address spaces for the program memory and the data memory. The program memory can be up to 64 Kbyte long and may partially reside on-chip. The 16-bit program counter is the addressing mechanism. The EA input determines the instruction fetch from the internal ROM or the external program memory. The interrupt service routines occupy the ROM locations 03H through 32H. The start address after a reset is 00H.

The data memory can consist of up to 64 Kbytes of off-chip RAM and is only accessed when an external move instruction is executed. The internal data memory is divided into three physically separate and distinct blocks: the lower 128 bytes of the RAM; the upper 128 bytes of the RAM; and the 128-byte special function register area. While the upper RAM area and the special function register area share the same address locations, they are accessed through different addressing modes.

In the lower RAM area four 8-register banks occupy locations 00H through 1FH. Only one of these banks is enabled at a time through a two-bit field in the program status word. The next sixteen bytes, locations 20H through 2FH, contain 128 bit addressable locations. The special function register area also has bit addressable locations.

An illustration of the memory organization is given in figure 2.

2.2 Special Function Registers in the Core

The 8051 microcontroller core can address 128 special function registers. Five of them are related to the core. A short description of their functionality is given below. The special function address is mentioned between brackets.

- The accumulator register (EOH) is the most important source and/or destination register provided that arithmetic and logical operations are performed. The accumulator is also used during special move instructions;
- The B register (FOH) is used during multiply and divide operations. The B register contains the multiplier or divider and receives the second result byte;
- The program status word register (DOH) reflects the program status information. The status bits specify:
  - Bit 0 is the parity flag. The parity flag shows whether the accumulator contains an even number of ones or not;
  - Bit 2 is the overflow flag. The overflow is set to indicate that the operation generated a result outside the data range;
  - Bit 3 and bit 4 are the register bank select control bits. The working register bank in the internal RAM corresponds with the binary contents of bit 4 and bit 3;
• Bit 6 is the auxiliary carry flag and is used for packed decimal operations;
• Bit 7 is the carry flag. The ALU uses the carry flag during arithmetic operations.

The Boolean processor applies the carry flag as accumulator.
• The stack pointer register (81H) is 8 bits wide. In case of a push action, the stack pointer
  is incremented before the data is stored. The stack may reside anywhere in the internal
  RAM, although it is initialized to start at location 08H after a reset;
• The data pointer consists of a high byte (82H) and a low byte (83H). Its intended function
  is to hold a 16-bit address. The data pointer may be manipulated as a 16-bit register or
  as two independent 8-bit registers.

The accumulator, the B register, and the program status word are also bit addressable.
2.3 Addressing Modes

The 8051 incorporates several addressing mechanisms which can each access a part of the memory spaces. The five addressing modes and the associated memory spaces are the following:

- Register addressing has access to the eight working registers of the selected register bank. The least three bits of the instruction op code indicate which register has to be used. The accumulator, the B register, the data pointer, and the carry flag can also be addressed as registers;
- Direct addressing is the only method of accessing the special function registers. The lower 128 bytes of the internal RAM are also directly addressable;
- Register-indirect addressing uses the contents of the first or the second register in the selected register bank as a pointer to a location in the internal RAM or the lower 256 bytes of the external data memory. Access to the full 64Kbyte external data memory address space is accomplished by using the 16-bit data pointer. The stack operations also use register-indirect addressing;
- Immediate addressing allows constants to be a part of the op code instruction in the program memory;
- Base-register plus index-register indirect addressing allows a byte to be accessed from the program memory via an indirect move from the location whose address is the sum of a base register (the data pointer or the program counter) and the index register (the accumulator). This mode facilitates look-up-table accesses.

2.4 Boolean Processor

The Boolean processor is an integrated bit processor within the 8051. It has its own instruction set, accumulator (the carry flag), and bit addressable RAM and I/O. The bit-manipulation instructions allow a bit to be set, cleared, complimented, jump-if-set, jump-if-not-set, jump-if-set-then-cleared, and moved to/from carry. Addressable bits, or their compliments, may logically ANDed or ORed with the contents of the carry flag. The result is returned to the carry register.

2.5 Instruction Set

The MCS-51 instruction set includes 111 instructions, 49 of which are single-byte, 45 two-byte, and 17 three-byte. The instruction op code format consists of a function mnemonic followed by a "destination, source" operand field. This field specifies the data type and
addressing method(s) to be used. The complete instruction set of the MCS-51 family can be seen in appendix A. The instruction set is divided into four functional groups:

- **Data Transfer**: These operations perform the internal and external data byte movements;
- **Arithmetic**: The 8051 has four basic mathematical operations. Only 8-bit operations using unsigned arithmetic are supported directly. The overflow flag, however, permits the addition and subtraction operation to serve for both unsigned and signed binary integers. Arithmetic can also be performed on packed decimal representations.
- **Logic**: The 8051 performs basic logic operations on both bit and byte operands;
- **Control Transfer**: All control transfer operations cause, some upon a specific condition, the program execution to continue at a non-sequential location in the program memory. There are three classes of control transfer operations:
  - Unconditional calls, returns and jumps;
  - Conditional jumps;
  - Interrupts.

### 2.6 CPU Timing

The 8051 machine cycle consists of 12 oscillator periods. Most one-byte and two-byte instructions execute in one machine cycle. Multiply and divide are the only instructions that take more than two cycles to complete. They take four cycles. Responding to an interrupt needs 38 to 81 oscillator periods. Regarding the oscillator frequency of 12 MHz, the mean number of instruction executions per second is about 0.7 MIPS.
3 Upgrading the Standard 8051 Core

The standard 8051 core, described in the previous chapter, has been upgraded. The enhancements consist of enlarged address spaces. 16 Mbyte address space for both the program memory and the data memory is accessible now. The stack address space is extended to 64 Kbytes. Space for eight additional instructions is created in the existing instruction map. The assembly source code is kept compatible with the standard 8051 instruction set. Still six instruction codes are free to implement new instructions (e.g. for a frame pointer).

The upgraded 8051 contains the following additions and adaptations:

- 24 not multiplexed address lines.
- 8 not multiplexed data lines.
- A 16-bit stack pointer for the 64 Kbyte stack addressing. The stack continues to the external data memory if the stack exceeds address FFH.
- A 24-bit program counter for 16 Mbyte program memory addressing.
- A 24-bit data pointer gives 16 Mbyte address space for specific external move operations.
- The data pointer load instruction, op code 90H, has become a 4-byte instruction to move the 24-bit immediate data.
- The absolute jump and subroutine call address range is reduced to a 1K page. This frees eight op codes at the addresses 81H, 91H, A1H, B1H, C1H, D1H, E1H, and F1H in the instruction set for new instructions.
- A global jump instruction, GJMP addr24, has been added with op code 81H. This is a 4-byte instruction that jumps to an address location in the 16 Mbyte program memory.
- A global subroutine call instruction, GCALL addr24, has been added with op code 91H. This is a 4-byte instruction that calls a subroutine at a location in the 16 Mbyte program memory.
- The subroutine call instructions and the interrupt routine call push a 3-byte return address onto the stack.
- The return from subroutine call and interrupt instructions pop a 24-bit address from the stack.

The enhanced 8051 instruction set is to be seen in appendix B. The new and altered instructions are shaded.
4 Core Interfaces

The IDaSS design ought to satisfy the specifications of the Object-Oriented (Hardware) System Design project library [3]. This library contains re-usable processor cores and input/output controllers. With this library, building a custom processor to perform specific tasks is eased to the point that only a few library components must be picked from the library, interconnected in IDaSS and converted to silicon with ASA. To be able to interconnect the elements of this library, strict rules for the interfaces (hardware and timing) must be followed. The boundary of the enhanced 8051 core provides the connection to the program and data memory, the input/output devices, and the interrupt controller. The configuration of the enhanced 8051 core is shown in figure 3. The I/O devices symbolise the timer/counters, serial port and so forth.

![Diagram of the enhanced 8051 core configuration.](image)

Figure 3: The enhanced 8051 core configuration.

4.1 Memory Interface

The interface to the program memory and the data memory consists out of four busses:
- An 8-bit bidirectional data bus DATAbus;
- The 25-bit address bus ADDRbus indicates which data word in the memory must be read or written. The most significant bit (bit 24) determines the selection between the program memory (%0) and the data memory (%1);
- The 2-bit control bus CTRLbus This output bus controls the read and write action in memory. The least significant bit (bit 0) indicates a read action. The most significant bit (bit 1) indicates a write action. The read and write strobes on the control bus are all active
Section of Digital Information Systems

Enhanced 8051 Core

high, a one indicates an action is to be performed. Simultaneously reading and writing is not allowed;

- The 1-bit ready bus Ready. This input bus is used for a simple handshake protocol that allows synchronisation of slow memories. The ready condition signals the last clock cycle of a memory access cycle with a non-zero value.

Whenever a read or write action is started the output must be kept stable until the ready signal is detected. A zero value puts the processor on hold.

4.2 Input/Output Interface

The input/output interface performs the access to the special function registers outside the processor core. The input/output interface is fitted with a separate input and output data bus to be able to update a special function register within a single clock cycle. The combined control and address bus is used to indicate the operation to be performed. No handshake is done in hardware. The input/output interface uses three buses:

- The 8-bit data output bus SFRo;
- The 8-bit data input bus SFRi;
- The 10-bit control/address bus SFRaddr. This output bus is build up with the special function register address (bit 0 through bit 7), the write bit (bit 8), and the read bit (bit 9). The read and write bits are both active high. A read-modify-write action is performed, when bit 8 and bit 9 are ones at the same time. If no input or output is performed, bit eight and bit nine should be zero and the address bits may take any value.

4.3 Interrupt Interface

I/O devices must be capable of interrupting the processor core with which they are communicating. An external interrupt controller (present in the library) detects interrupts taken priorities into account, and outputs an interrupt vector. In this case extra interrupt masking flags within the processor core are not necessary.

The interrupt interface consists out of two buses:

- The 3-bit interrupt vector bus Ireq. This input bus contains the interrupt number generated by the interrupt controller. All bits high (value 7) indicates no interrupt. The other seven values specify the interrupt routine to be performed. If more interrupt vectors are required in the future, the interrupt vector bus width can be increased. In that case all bits high still indicates no interrupt request;
- The 2-bit handshake bus Iack. This output bus signals to the interrupt controller whether the processor core started or finished interrupt handling. When bit 0 is one clock period pulse high (%1), the core has started handling the interrupt routine indicated by the vector
on the interrupt vector bus during the previous clock cycle. A one clock period pulse high (\%1) of bit 1 indicates that the processor core has finished an interrupt routine. This pulse is used within the interrupt controller to select the next interrupt for handling and reset mask bits automatically. The interrupt controller does not generate a new interrupt vector in the clock period which follows the interrupt routine end pulse.
Chapter 5 Enhanced 8051 Core Design for ASA

The purposes mentioned below have been taken into account as the architecture of the enhanced 8051 core was designed.

- The core interfaces have to satisfy the specifications for IDaSS library components;
- The number of clock cycles necessary to execute an instruction ought to be restricted to a minimum;
- The design must be suitable for synthesising with the ASA silicon compiler;
- The design may not explode in complexity and size.

In essence, the requirements mentioned are contradictory. The processor performance is in proportion to the die area consumption, wherein an optimum has to be found. The enhanced 8051 core reads sequentially the instruction code bytes from the program memory. In the last cycle of the instruction execution the next op code is fetched. This method is called prefetching. In contrast to pipelining, prefetching does not request a lot of additional control logic. By means of prefetching, one clock cycle is saved with loading the instruction bytes. Further on, the architecture is designed to perform the instruction execution as quickly as possible. The modification of data bytes occurs within one clock cycle independent of the addressing mode, although only the input/output interface dictated the one clock cycle data modification. The consequence is that the internal data bus structure consists of unidirectional data buses. In combination with the several addressing modes, the data buses will consume a considerable amount of die area.

The architecture of the enhanced 8051 core has been designed in IDaSS. At first, a design without any restrictions has been made. This ideal architecture, briefly described in appendix D, cannot be synthesised with the ASA silicon compiler. Therefore, a new architecture has been designed specifically for synthesis with the ASA silicon compiler. Designing for ASA means many restrictions have to be taken into account. On the one side, some useful IDaSS functions are simply not supported, on the other side the hardware consequences have to be examined. For example, the functionality of an IDaSS register is very extensive. In ASA, the IDaSS register functionality is implemented via counters and adders. To avoid redundancy, it is recommended to use only the basic register functionality and put the extra functionality in an additional operator.

The converter 'idasstosid' and the ASA silicon compiler each have their limitations [4]. Many limitations can simply be get around, but two problems have influenced the architecture design. At first, an asynchronous signal, like the handshake signal of the external memories, has to be detected without losing a clock cycle. The solution resulted in an implementation of an internal bus interface which is described in subsection 5.2. The second restriction comes from ASA that only has synchronous read RAM. This means that every read or modify action
with a RAM data byte needs two clock cycles. Therefore, the register banks in the internal RAM are implemented as registers to increase the execution time. Details are given in subsection 5.3.

Figure 4 displays the entire enhanced 8051 core architecture in IDaSS. The architecture is composed of subschematics. The functionality of the subschematics is explained in the following subsections. The design is based on small autonomous operators and a simple state machine instead of putting the whole intelligence in a complex, ergo large and slow, state machine. Appendix E contains the entire IDaSS description.

![Diagram of the enhanced 8051 core architecture in IDaSS](image)

**Figure 4: The enhanced 8051 core architecture in IDaSS.**

### 5.1 State Controller

The state machine ARBITER has eight states and co-ordinates the core activities. The first state is the wait state during the multiply or divide instruction execution, but serves also as the start up state after a reset. The next four states fetch the instruction bytes from the external program memory. After all the instruction bytes are fetched, a 10-bit instruction code word is generated in the IR_Sel subschematic. The state machine selects a 2-bit code to be concatenated to the instruction register IR. The resulting 10-bit instruction code word indicates the instruction execution status and runs through the whole design to drive the operators. Some comprehensive instructions, like the move code byte and the move external, are partially handled by the state machine. In that case, relatively simple state descriptions fulfil
the remaining complex operations. The last three states provide the program counter push onto and pop from the stack during calls and returns.

5.2 External Memory Interface

The BUSIFACE subschematic, to be seen in figure 5, provides the external memory access. According to the memory interface as described in subsection 4.1, a ready signal indicates the last cycle of a memory access. Unfortunately, IDaSS signals are not supported by the converter 'idasstosid'. So, the Ready signal had to be clocked into a register which can be checked in the state machine. Around the ready register RDY, an external memory interface is constructed to control the external memory access. If the continuous loading RDY register contains the value one, an external memory access has completed and the next processor

---

Figure 5: The external memory interface.
action can occur. Normally, the program counter unit drives the external memory interface, but the SPwrop operator can switch the ADDR_op and WR_op multiplexers over to an external stack access.

An external memory access starts when the WR_CTRL subschematic receives a read or write signal. At the same time, the Addr_out subschematic contains the external memory address. During a write cycle the Data_out subschematic outputs the data. The external memory interface continues the memory access until the ready signal is detected to support slow responding external memories. Therefore, the output status is retained in the Addr_out, Data_out, and WR_Ctrl subschematics. In case of a read cycle the Data_in subschematic is active instead of the Data_out subschematic and one of the four data input registers will be loaded. The data input registers are the instruction register IR, the immediate data register IM, the direct address register AR, and the relative jump offset register REL. Placing these registers in the external memory interface avoids a delaying data movement from a data input register to one of the four registers elsewhere in the core. The register selection operator RSop determines the data input register to be loaded. The RSbuf register buffers the register selection code word during a slow read cycle.

Because the converter 'idasstosid' is incapable of performing bidirectional buses automatically, the continuous data output bus DATobus, the data input bus DATibus and the 1-bit three-state driver bus enable are connected manually to the 8-bit bidirectional data bus by editing the SID-file.

5.3 Internal RAM

The RAMunit subschematic, shown in figure 6, comprehends the 256 byte on-chip RAM. Unfortunately, the ASA RAM is functionally different from IDaSS RAM. The ASA RAM read behaviour is synchronous, though in IDaSS it is asynchronous. Writing is asynchronous as in IDaSS. In essence, only a single valid IDaSS RAM model can be converted into working silicon. The IDaSS library contains RAM models for simulation and synthesis. The simulation model imitates the ASA RAM behaviour in IDaSS and is replaced with an equivalent convertible model before synthesis. The design contains the control bus driven model BUSIMPL.

The synchronous read behaviour means that the data is delayed one clock cycle before it is available for further operations. This results in a two clock cycle register addressing execution and even four clock cycles for the execution of indirect register addressed instructions. To obtain faster (indirect) register addressing, the four RAM register banks (the lowest 32 bytes) are implemented as registers. These 32 registers, situated in the ASALRAM subschematic, consume nevertheless much more die area than ordinary ASA RAM. Besides that, the partial
Figure 6: The internal RAM.

register RAM implementation requires an extra address decoder and a RAM selector. The RAMSelect and ADDRgen operators represent the RAM selector. The ADDRgen operator subtracts 20H from the ASA RAM address, because the ASA RAM starts at address 00H. As a consequence of the synchronous read behaviour, the modification of an ASA RAM data byte within one clock cycle demands buffers and some extra logic. The HWmod register stores the one clock cycle delayed modify (i.e. write) signal which activates the HRAM operator the next clock cycle. The RAM address is stored in the HADDRbuf register.

The DVop operator builds the 2-bit RAM data valid signal from the RAM control signals. The RAM data valid signal constitutes the automatic instruction execution handling.

The most significant bit (bit 2) of the RAM control bus RAMIWR indicates the indirect register addressing mode if set. The indirect address is read in the selected register bank and stored in the IADDR register. The next clock cycle the indirect addressing is performed with the control signal buffered in the IWR register.

The Busyop operator and accompanying Busy register avoid a new RAM access until the RAM data valid signal of an earlier access is generated. The state machine overrules the busy protection during the execution of instructions which may access the RAM in successive clock cycles. The RAMDATobuf subschematic stores the latter RAM data output to rectify the difference in RAM access time.
The RAM_SEL subschematic, shown in figure 7, provides the internal RAM control signals and addresses. The REG_SEL operator generates the (indirect) register addresses and control signals. The IWR_Sel operator multiplexes the RAM addresses and control signals.

Figure 7: The internal RAM address and control signal generator.

5.4 Direct Addressing Controller

Figure 8 shows the DIRECT subschematic where the direct addressing is arranged. The RWop operator generates the 2-bit read, write, or modify code word for the direct addressed data byte. In the AR_SEL operator the RW code word is concatenated to the direct address byte. In case of bit addressing the address byte containing the bit address is selected.

Figure 8: The direct addressing mode control unit.
The Dir_Sel operator, controlled by the 10-bit SFR output of the AR_SEL operator, multiplexes the direct data input, drives the external special function register control/address bus SFRaddr and generates the 1-bit direct data valid signal DIRDV. The DIRDV signal indicates the direct access status, since the ASA RAM is also partially directly addressable. In spite of the different data access times, all the direct data are modified within one clock cycle.

The DIRbuffer subschematic serves as buffer to eliminate the timing problem with the move direct byte to direct and (indirect) register instructions. The direct data byte is only stored if it is not obtained from the ASA RAM. The following clock cycle, the buffer outputs the data byte or transits the ASA RAM data byte.

5.5 Arithmetic and Logical Unit

The arithmetic and logical operations take place in the ALUunit subschematic, to be seen in figure 9. Furthermore, the ALUunit determines the 1-bit conditional jump vector Jstt.

Figure 9: The arithmetic and logical unit.

The ALUunit consists of a byte and a bit processor. Figure 10 shows the byte processing subschematic ALUs. The byte operations have been distributed among four ALUs, because
ASA is not capable of synthesising a complex block like one huge ALU. The 10-bit adders in ALU1 and ALU2s are re-used by means of performing similar operations with the same ASA library element. This implies some adaption to the function descriptions, but the amount of generated hardware is reduced to a minimum. The bit processing occurs in the BOOL operator. The bit is extracted from the direct data byte in the BYTEop operator. The three least significant bits of the AR register indicate the bit number.

The multiplexers ALUin1MUX and ALUin2MUX pick out the data bytes in such a way that plural function implementation could be prevented. The ALUunit also transits data bytes during move instructions which results in less routing and smaller multiplexers in the core. Since the ALU is always present in the critical path and three-stated output ports are slow, the ALU output ports are multiplexed.

Figure 10: The byte processors.

In ALU1 one 10-bit adder performs the execution of the increment, decrement, add, add with carry, and subtract with borrow instruction. Unfortunately, the re-use of the 10-bit adder could
not proceed with the multiply and divide instruction, because the operator became too complex for logic optimisation in ASA. Therefore, a second 10-bit adder is present in ALU2. In ALU3 two 4-bit adders were needed to execute the decimal adjust instruction. Joining this instruction in ALU1 or ALU2 resulted again in too complex operators. At last, ALU4 contains an 8-bit unsigned compare less than operator to set or reset the carry flag during the compare and jump if not equal instruction execution. All in all, a redesign of the four ALUs should be done to improve the re-use of the ASA library elements which reduces the total die area consumption.

The multiply and divide instruction executions take eight clock cycles. In this way, the hardware requirements have been restricted to a minimum. In the future, the multiplication and division can be speeded up, if desirable. The CNTR register is used as counter to indicate the number of passed multiply or divide cycles. The TMP register, attached to ALU2, contains the temporary results during a multiplication or division.

The DV_CTRL operator generates the 1-bit control signal CTRL which indicates that the data on the buses is valid and can be stored. Depending on the instruction code, the CTRL signal is the read or write bit of the RAM data valid signal or the direct data valid signal DIRDV.

5.6 Program Counter

The PCunit subschematic, shown in figure 11, provides the interrupt handling, new program counter contents and memory addressing. Because of the variable instruction execution length, in particular as a consequence of synchronous read RAM, the CTRL signal arisen from the DV_CTRL operator in the ALUunit was needed to indicate the end of an execution cycle.

If the interrupt number register Ino contains a value between zero and six during the last instruction execution cycle, an interrupt is requested. In case of an interrupt request the INTR operator generates the 2-bit interrupt acknowledge signal and stores the program counter. The interrupt acknowledge signal activates the Iop operator that sets the 1-bit register I to indicate the interrupt handling. The state machine decodes the I register in state STAGE1 and pushes the first byte of the program counter onto the stack. The states STACK1 and STACK2 push the remaining two program counter bytes onto the stack. The last state STACK3 resets the interrupt handling register I and loads the program counter with the interrupt routine address.

When no interrupt is requested, the CTRL signal is used for prefetching. The CTRL signal activates the read signal in the WR operator and the PCinc operator via the PCop operator. The PCinc operator increments the new program counter value before loading it into the program counter register PC. In this way, the program counter register always represents the next instruction fetch address. The program counter register is continuously loading, except for the move code byte instruction executions.
The RELop operator adds the signed 8-bit offset byte REL to the program counter. If the Jtst signal appoints a short relative jump, the JUMP operator transits the branch address instead of the program counter contents to the PCgen operator. The jump, call and return addresses are generated with the PCbuf register contents in the PCgen operator. The move code byte address is also calculated by the PCgen operator. Then the PCadgen receives the new program counter contents. The PCadgen operator provides the addresses for the external program memory and the external data memory. If an address is allocated to the external program memory, a zero bit is concatenated in front of it. During the move external instructions the most significant address bit is a one.

The PCbufop operator arranges the storage of the new program counter bytes in the PCbuf register during jumps, calls, interrupt handling and returns (from interrupt). The input comes from the internal memory, the external memory, or the interrupt number register. During calls and interrupt handling the PCdatop operator extracts the program counter bytes to be pushed onto the stack.

The program counter increment following upon a relative jump calculation means two 24-bit add operations in succession. These adders return in the critical path and slow down the processor operation frequency dramatically. Therefore, it is recommended to re-design the
PCunit in the near future and omit the prefetching after a relative jump to increase the processor speed.

5.7 ALU Related Registers

The SFR_core subschematic, shown in figure 12, contains the accumulator register A, the program status word register PSW, and the B register which are primary related to the ALU, but also directly addressable. Since each register has the same modification logic, the 'X' in the operator names below replaces the registers names.

![Figure 12: The ALU related registers.](image)

The direct addressed write signal is extracted from the SFR bus in the DirXop operator. The XWop operator generates a write signal depending on the instruction code word or transit the direct write signal. The accumulator and the program status word write signal is only valid if the CTRL signal is high (%1). The XiMUX operators multiplex the new register contents.

The program status word is split up into two registers; one for the parity bit, the P register, and the other one for the remaining seven status bits, the PSW register. The PSWop operator
provides the 8-bit program status word output and the new contents for the PSW register. The Parity register, bit 0 of the program status word, represents the even parity of the accumulator. The actual accumulator contents determine the parity via the Pop operator. So, the parity update is one clock cycle delayed and does not contribute to the critical path length anymore. In spite of this delay, the correct program status word is available when it is addressed directly. In that case, fetching of the instruction bytes overlaps the delay.

5.8 Stack Pointer

The stack pointer is placed in the SP_CTRL subschematic, shown in figure 13. The 16-bit stack pointer is divided into a low and high byte for the direct addressing mode. The special function registers SPL and SPH take possession of the addresses 81H respectively 85H. The SP_Sel operator extracts and SPL and SPH from the stack pointer. The SPop provides the modification of the continuous loading stack pointer register SP_reg.

Figure 13: The stack pointer logic.
The stack pointer addresses the 64 Kbyte stack. The stack may reside anywhere in the internal RAM and can be continued to the external data memory. The STACKop operator contains a 16-bit adder to perform the push and pop addresses and increments respectively decrements the stack pointer at the same time. Since the STACKop operator modifies the stack pointer during stack operations, the stack pointer register needs no extra functionality. The state machine controls the STACK operator which generates the stack write/read control signal. In case of popping SPL or SPH from the stack, the stack pointer may not be decremented. This exception is obtained with the NoDec operator. Although this exception is compatible with the standard instruction set, this special case may be removed for the extended stack pointer on account of diminished profit.

The decoding complexity in the state machine had to be reduced for the stack operations. This is achieved by decreasing the number of decoding bits. During a pop from the stack the POPop operator encodes the 16-bit POP address into a 3-bit code word and stores it in the POPreg register. Five code words distinguish the register RAM, ASA RAM, external RAM and the transitions between the different kinds of memories. The POPreg register contents determine the movement of the popped data and is controlled by the state machine. The stack valid register SV points out the end of a stack operation. In case of an external access the SV register is loaded with the ready signal, otherwise the SVop operator loads the SV register with a one. The 1-bit SV register replaces also the stack pointer decoding in the state machine during push operations.

5.9 Data Pointer

The data pointer is placed in the DP_reg subschematic, to be seen in figure 14. The 24-bit data pointer is divided into three bytes in the direct addressing mode. The three data pointer bytes DPL, DPH and DPE take possession of the special function register addresses 82H, 83H, and 84H, respectively. The DP_Sel operator extracts the addressed byte from the data pointer. The data pointer register DPreg is default loading. Therefore, the operator DPop gives on the old data pointer contents unless the data pointer has been modified. The data pointer can be loaded with a new constant. During this instruction execution the DP_CTRL operator generates the data pointer special function addresses to replace the data pointer register contents with immediate data. In this way, the internal structure is re-used.
Figure 14: The data pointer logic.

The data pointer increment makes use of the IDaSS register increment function, however the data pointer is implemented as a 24-bit counter in ASA. On the other hand, the instruction execution takes only one clock cycle.
6 ASA synthesis

Before the ASA silicon compiler can be used, the IDaSS design must be converted to the ASA description language SID by means of the conversion tool 'idasstosid'. The conversion succeeded after some changes in the IDaSS design, like renaming reserved words, adapting the control connector coding to the binary form and appending a parallel input connector to the control connector if an operator function uses the control value. In the SID-file the bidirectional data bus is connected. Also, the type parameter of the IDaSS registers and the adders have been set to maximum speed in the expense of die area consumption. Sometimes the mysterious error 'segmentation fault' appeared after the design had been changed. In that case, the only solution was performing the changes in the previous design in another way.

In first instance, the ASA compiler had a problem with deeply nested state machines. The converter had cut off some states, because 'idasstosid' restricts the line length to 80 characters. Since deeply nested state machines explode in complexity, the states have been rewritten with less transitions. Furthermore, the test conditions must be kept simple. For example, testing the 16-bit stack pointer register contents was too complex and was replaced by testing the 3-bit POPreg register. On the other hand, certain operators have been simplified or even split up, as the ALUs, to perform the logical optimalisation in ASA.

After the conversion to gates, a dummy inverter is added to the STACK operator in the SID-file, because the STACK operator contained no logic. The dummy inverter is removed automatically when the layout is made.

The synthesis time reduces considerably when a control connector calls a function only once. Now, the synthesis of the complete design to a layout takes only three hours. The total die area consumption is 28 mm². The maximum clock frequency is 7.5 MHz. In appendix C, the number of needed execution clock cycles is listed per instruction. Broadly, an instruction execution takes the same number of clock cycles as instruction bytes. An ASA RAM access takes one clock cycle more and stack operations take three clock cycles extra. Since about 40 percent of the instruction set is executed in one clock cycle and around 50 percent of the remaining instructions is executed in two or three clock cycles, the mean number of instruction executions per second is approximately 3.5 MIPS.

The response to an interrupt request takes between four and eleven clock cycles. The response time includes finishing of the instruction execution (one to seven clock cycles), pushing the program counter onto the stack (three clock cycles) and fetching the first byte of the interrupt routine (one clock cycle).
7 Conclusions

The designed enhanced 8051 core can address 16 Mbyte of external memory and has a 64 Kbyte stack depth. The instruction set is adapted, but is still compatible with the standard 8051 instruction set. The core satisfies the IDaSS library interfaces and is synthesised with the ASA silicon compiler.

The synthesis technology is the CMOS 1.0 micron process. The architecture die area consumption is 28 mm². The ASA timing analyzer calculates a maximum clock frequency of 7.5 MHz. This frequency is based on the longest path in the circuit. In practice, this path will not be the real critical path in the circuit, thus the enhanced 8051 core can probably operate on a higher clock frequency.

The Idass register implementation of the four register banks in the internal RAM demands an incredible amount of die area. Making a data pad layout of the RAM register part manually in ASA is compacter and saves a lot of die area. It is a pity that ASA only has synchronous read RAM. The presence of asynchronous read and write RAM would simplify the RAMunit considerably. The additional control logic that serves the ASA RAM byte modification and the selection between the ASA RAM and the RAM registers becomes unnecessary. Also the CTRL signal can be removed, because the execution time of every instruction is fixed in that case.

The 24-bit adders in the program counter unit cause a part of the long critical path length. Abandon the prefetching, e.g. during a relative jump execution, prevents two 24-bit add operations in succession. Performing the add operations by means of one 24-bit adder increases clock frequency and saves much die area too.

The four byte processors in the ALUunit do not optimally re-use the hardware library elements. Redesigning the byte processors in such a way that only one 10-bit adder is necessary to perform all the arithmetic operations reduces the hardware requirements and the critical path delay. The operator which contains the 10-bit adder receives the data input via some intelligent multiplexers in order to avoid a too complex operator for ASA optimization.

Furthermore, it is recommended to update the instruction set. Some exceptions, such as no decrement after popping stack pointer low or high byte, are not relevant anymore and can be removed. If the stack would only reside in the external data memory, the extra stack logic in the SP_CTRL subschematic becomes superfluous and the three STACK states in the state machines can be integrated in other states. This will result in a less complex state machine.
The instruction execution needs a minimum of clock cycles; 40 percent of the instructions is executed in one clock cycle and 50 percent of the instruction executions takes two or three clock cycles. Since the clock frequency is 7.5 Mhz, the mean number of instruction executions per second is estimated at 3.5 MIPS. The standard 8051 runs at 12 Mhz, but needs at least 12 clock cycles for an instruction execution. So, the enhanced 8051 core is four times faster than the standard 8051 core.

Finally, further conformations as described above will improve the core performance. A 10 MHz clock frequency and 30 percent die area save should be achievable.
References


[4] Verschueren, A.C., "IDaSS does and don'ts to enable synthesis of ASIC's with the ASA silicon compiler", Internal publication of the Eindhoven University of Technology, Department of Electrical Engineering, Section of Digital Information Systems, the Netherlands, September 17, 1993.
## Appendix A: MCS-51 Instruction Set

<table>
<thead>
<tr>
<th>L</th>
<th>H</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6 .. 7</th>
<th>8 .. F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>NOP</td>
<td>AJMP page 0</td>
<td>LJMP addr16</td>
<td>RR</td>
<td>INC</td>
<td>INC</td>
<td>INC</td>
<td>INC</td>
<td>INC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td></td>
<td>dir</td>
<td>@Ri</td>
<td></td>
<td>Rn</td>
</tr>
<tr>
<td>1</td>
<td>JBC bit, rel</td>
<td>ACALL page 0</td>
<td>LCALL addr16</td>
<td>RRC</td>
<td>DEC</td>
<td>DEC</td>
<td>DEC</td>
<td>DEC</td>
<td>DEC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td></td>
<td>dir</td>
<td>@Ri</td>
<td></td>
<td>Rn</td>
</tr>
<tr>
<td>2</td>
<td>JB bit, rel</td>
<td>AJMP page 1</td>
<td>RET</td>
<td>RL</td>
<td>ADD</td>
<td>ADD</td>
<td>ADD</td>
<td>ADD</td>
<td>ADD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>GO</td>
<td>A, #data</td>
<td>A, dir</td>
<td>A, @Ri</td>
<td>A, Rn</td>
</tr>
<tr>
<td>3</td>
<td>JNB bit, rel</td>
<td>ACALL page 1</td>
<td>RETI</td>
<td>RRC</td>
<td>ADDC</td>
<td>ADDC</td>
<td>ADDC</td>
<td>ADDC</td>
<td>ADDC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>GO</td>
<td>A, #data</td>
<td>A, dir</td>
<td>A, @Ri</td>
<td>A, Rn</td>
</tr>
<tr>
<td>4</td>
<td>JC rel</td>
<td>AJMP page 2</td>
<td>ORL</td>
<td>ORL</td>
<td>ORL</td>
<td>ORL</td>
<td>ORL</td>
<td>ORL</td>
<td>ORL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>dir, A</td>
<td>dir, A</td>
<td>dir, A</td>
<td>dir, #data</td>
<td>dir, #data</td>
<td>dir, #data</td>
<td>dir, #data</td>
<td>dir, #data</td>
</tr>
<tr>
<td>5</td>
<td>JNC rel</td>
<td>ACALL page 2</td>
<td>ANL</td>
<td>ANL</td>
<td>ANL</td>
<td>ANL</td>
<td>ANL</td>
<td>ANL</td>
<td>ANL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>dir, A</td>
<td>dir, A</td>
<td>dir, A</td>
<td>dir, #data</td>
<td>dir, #data</td>
<td>dir, #data</td>
<td>dir, #data</td>
<td>dir, #data</td>
</tr>
<tr>
<td>6</td>
<td>JZ rel</td>
<td>AJMP page 3</td>
<td>XRL</td>
<td>XRL</td>
<td>XRL</td>
<td>XRL</td>
<td>XRL</td>
<td>XRL</td>
<td>XRL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>dir, A</td>
<td>dir, A</td>
<td>dir, A</td>
<td>dir, #data</td>
<td>dir, #data</td>
<td>dir, #data</td>
<td>dir, #data</td>
<td>dir, #data</td>
</tr>
<tr>
<td>7</td>
<td>JNZ rel</td>
<td>ACALL page 3</td>
<td>ORL</td>
<td>ORL</td>
<td>ORL</td>
<td>ORL</td>
<td>ORL</td>
<td>ORL</td>
<td>ORL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>C, bit</td>
<td>dir, #data</td>
<td>dir, #data</td>
<td>dir, #data</td>
<td>dir, #data</td>
<td>dir, #data</td>
<td>dir, #data</td>
<td>dir, #data</td>
</tr>
<tr>
<td>8</td>
<td>SJMP rel</td>
<td>AJMP page 4</td>
<td>ANL</td>
<td>ANL</td>
<td>ANL</td>
<td>ANL</td>
<td>ANL</td>
<td>ANL</td>
<td>ANL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>C, bit</td>
<td>dir, #data</td>
<td>dir, #data</td>
<td>dir, #data</td>
<td>dir, #data</td>
<td>dir, #data</td>
<td>dir, #data</td>
<td>dir, #data</td>
</tr>
<tr>
<td>9</td>
<td>MOV DPTR, #data16</td>
<td>ACALL page 4</td>
<td>MOV</td>
<td>MOV</td>
<td>MOV</td>
<td>MOV</td>
<td>MOV</td>
<td>MOV</td>
<td>MOV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bit, C</td>
<td>A, @A+PC</td>
<td>A, @A+DPTR</td>
<td>A, @A+DPTR</td>
<td>A, @A+DPTR</td>
<td>A, @A+DPTR</td>
<td>A, @A+DPTR</td>
<td>A, @A+DPTR</td>
</tr>
<tr>
<td>10</td>
<td>A ORL C, /bit</td>
<td>AJMP page 5</td>
<td>MOV</td>
<td>INC</td>
<td>MUL</td>
<td>MOV</td>
<td>MOV</td>
<td>MOV</td>
<td>MOV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>C, bit</td>
<td>DPTR</td>
<td>AB</td>
<td>AB</td>
<td>dir, #data</td>
<td>dir, #data</td>
<td>dir, #data</td>
<td>dir, #data</td>
</tr>
<tr>
<td>11</td>
<td>B ANL C, /bit</td>
<td>ACALL page 5</td>
<td>CPL</td>
<td>CPL</td>
<td>CPL</td>
<td>CJNE</td>
<td>CJNE</td>
<td>CJNE</td>
<td>CJNE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bit</td>
<td>bit</td>
<td>bit</td>
<td>C</td>
<td>A, #data, rel</td>
<td>A, #data, rel</td>
<td>A, #data, rel</td>
<td>A, #data, rel</td>
</tr>
<tr>
<td>12</td>
<td>C PUSH dir</td>
<td>AJMP page 6</td>
<td>CLR</td>
<td>CLR</td>
<td>CLR</td>
<td>SWAP</td>
<td>XCH</td>
<td>XCH</td>
<td>XCH</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bit</td>
<td>C</td>
<td>C</td>
<td>A</td>
<td>A, dir</td>
<td>A, @Ri</td>
<td>A, @Ri</td>
<td>A, @Ri</td>
</tr>
<tr>
<td>13</td>
<td>D POP dir</td>
<td>ACALL page 6</td>
<td>SETB</td>
<td>SETB</td>
<td>SETB</td>
<td>DA</td>
<td>DJNZ</td>
<td>XCHD</td>
<td>DJNZ</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bit</td>
<td>C</td>
<td>C</td>
<td>A</td>
<td>dir, rel</td>
<td>A, @Ri</td>
<td>A, @Ri</td>
<td>A, @Ri</td>
</tr>
<tr>
<td>14</td>
<td>E MOVX A, @DPTR</td>
<td>AJMP page 7</td>
<td>MOVX</td>
<td>MOVX</td>
<td>MOVX</td>
<td>CLR</td>
<td>MOV</td>
<td>MOV</td>
<td>MOV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A, @R0</td>
<td>A, @R1</td>
<td>A, @R1</td>
<td>A</td>
<td>A, dir</td>
<td>A, @Ri</td>
<td>A, @Ri</td>
<td>A, @Ri</td>
</tr>
<tr>
<td>15</td>
<td>F MOVX @DPTR, A</td>
<td>ACALL page 7</td>
<td>MOVX</td>
<td>MOVX</td>
<td>MOVX</td>
<td>CPL</td>
<td>MOV</td>
<td>MOV</td>
<td>MOV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>@R0, A</td>
<td>@R1, A</td>
<td>@R1, A</td>
<td>A</td>
<td>dir, A</td>
<td>@Ri, A</td>
<td>@Ri, A</td>
<td>@Ri, A</td>
</tr>
</tbody>
</table>
## Appendix B: Enhanced 8051 Instruction Set

<table>
<thead>
<tr>
<th>L.H.</th>
<th>Instruction</th>
<th>Address</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>NOP</td>
<td>page 0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>JBC</td>
<td>page 0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>JB</td>
<td>page 1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>JNB</td>
<td>page 1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>JC</td>
<td>page 2</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>JNC</td>
<td>page 2</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>JZ</td>
<td>page 3</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>JNZ</td>
<td>page 3</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>SJMP</td>
<td>page 3</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>MOV</td>
<td>addr24</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>ORL</td>
<td>C, #bit</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>ANL</td>
<td>C, #bit</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>PUSH</td>
<td>dir</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>POP</td>
<td>dir</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>MOVX</td>
<td>A, @DPTR</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>MOVX</td>
<td>@DPRTR, A</td>
<td></td>
</tr>
</tbody>
</table>

### Instructions Details:

- **NOP**: No Operation
- **JBC**: Jump if bit condition is set
- **JB**: Jump if bit condition is set
- **JNB**: Jump if bit condition is not set
- **JC**: Jump if condition is set
- **JNC**: Jump if condition is not set
- **JZ**: Jump if zero
- **JNZ**: Jump if not zero
- **SJMP**: Jump short
- **MOV**: Move
- **ORL**: OR Logic
- **ANL**: AND Logic
- **XRL**: XOR Logic
- **CLR**: Clear
- **PUSH**: Push
- **POP**: Pop
## Appendix C: Instruction Execution Time Specification

Table 1. Arithmetic operation execution time.

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Instruction bytes</th>
<th>Execution cycles enhanced 8051 core</th>
<th>Execution cycles standard 8051 core</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD A, Rn</td>
<td>1</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>ADD A, direct</td>
<td>2</td>
<td>2-3</td>
<td>12</td>
</tr>
<tr>
<td>ADD A, @Ri</td>
<td>1</td>
<td>2-3</td>
<td>12</td>
</tr>
<tr>
<td>ADD A, #data</td>
<td>2</td>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>ADDC A, Rn</td>
<td>1</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>ADDC A, direct</td>
<td>2</td>
<td>2-3</td>
<td>12</td>
</tr>
<tr>
<td>ADDC A, @Ri</td>
<td>1</td>
<td>2-3</td>
<td>12</td>
</tr>
<tr>
<td>ADDC A, #data</td>
<td>2</td>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>SUBB A, Rn</td>
<td>1</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>SUBB A, direct</td>
<td>2</td>
<td>2-3</td>
<td>12</td>
</tr>
<tr>
<td>SUBB A, @Ri</td>
<td>1</td>
<td>2-3</td>
<td>12</td>
</tr>
<tr>
<td>SUBB A, #data</td>
<td>2</td>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>INC A</td>
<td>1</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>INC Rn</td>
<td>1</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>INC direct</td>
<td>2</td>
<td>2-3</td>
<td>12</td>
</tr>
<tr>
<td>INC @Ri</td>
<td>1</td>
<td>2-3</td>
<td>12</td>
</tr>
<tr>
<td>DEC A</td>
<td>1</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>DEC Rn</td>
<td>1</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>DEC direct</td>
<td>2</td>
<td>2-3</td>
<td>12</td>
</tr>
<tr>
<td>DEC @Ri</td>
<td>1</td>
<td>2-3</td>
<td>12</td>
</tr>
<tr>
<td>INC DPTR</td>
<td>1</td>
<td>1</td>
<td>24</td>
</tr>
<tr>
<td>MUL AB</td>
<td>1</td>
<td>8</td>
<td>48</td>
</tr>
<tr>
<td>DIV AB</td>
<td>1</td>
<td>8</td>
<td>48</td>
</tr>
<tr>
<td>DA A</td>
<td>1</td>
<td>1</td>
<td>12</td>
</tr>
</tbody>
</table>
Table 2. Logical operation execution time.

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Instruction bytes</th>
<th>Execution cycles enhanced 8051 core</th>
<th>Execution cycles standard 8051 core</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANL A, Rn</td>
<td>1</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>ANL A, direct</td>
<td>2</td>
<td>2-3</td>
<td>12</td>
</tr>
<tr>
<td>ANL A, @Ri</td>
<td>1</td>
<td>2-3</td>
<td>12</td>
</tr>
<tr>
<td>ANL A, @data</td>
<td>2</td>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>ANL direct, A</td>
<td>2</td>
<td>2-3</td>
<td>12</td>
</tr>
<tr>
<td>ANL direct, @data</td>
<td>3</td>
<td>3-4</td>
<td>24</td>
</tr>
<tr>
<td>ORL A, Rn</td>
<td>1</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>ORL A, direct</td>
<td>2</td>
<td>2-3</td>
<td>12</td>
</tr>
<tr>
<td>ORL A, @Ri</td>
<td>1</td>
<td>2-3</td>
<td>12</td>
</tr>
<tr>
<td>ORL A, @data</td>
<td>2</td>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>ORL direct, A</td>
<td>2</td>
<td>2-3</td>
<td>12</td>
</tr>
<tr>
<td>ORL direct, @data</td>
<td>3</td>
<td>3-4</td>
<td>24</td>
</tr>
<tr>
<td>XRL A, Rn</td>
<td>1</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>XRL A, direct</td>
<td>2</td>
<td>2-3</td>
<td>12</td>
</tr>
<tr>
<td>XRL A, @Ri</td>
<td>1</td>
<td>2-3</td>
<td>12</td>
</tr>
<tr>
<td>XRL A, @data</td>
<td>2</td>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>XRL direct, A</td>
<td>2</td>
<td>2-3</td>
<td>12</td>
</tr>
<tr>
<td>XRL direct, @data</td>
<td>3</td>
<td>3-4</td>
<td>24</td>
</tr>
<tr>
<td>CLR A</td>
<td>1</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>CPL A</td>
<td>1</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>RL A</td>
<td>1</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>RLC A</td>
<td>1</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>RR A</td>
<td>1</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>RRC A</td>
<td>1</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>SWAP A</td>
<td>1</td>
<td>1</td>
<td>12</td>
</tr>
</tbody>
</table>
Table 3. Data transfer execution time.

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Instruction bytes</th>
<th>Execution cycles enhanced 8051 core</th>
<th>Execution cycles standard 8051 core</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOV A, Rn</td>
<td>1</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>MOV A, direct</td>
<td>2</td>
<td>2-3</td>
<td>12</td>
</tr>
<tr>
<td>MOV A, @Ri</td>
<td>1</td>
<td>2-3</td>
<td>12</td>
</tr>
<tr>
<td>MOV A, #data</td>
<td>2</td>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>MOV Rn, A</td>
<td>1</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>MOV Rn, direct</td>
<td>2</td>
<td>3</td>
<td>24</td>
</tr>
<tr>
<td>MOV Rn, #data</td>
<td>2</td>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>MOV direct, A</td>
<td>2</td>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>MOV direct, Rn</td>
<td>2</td>
<td>3</td>
<td>24</td>
</tr>
<tr>
<td>MOV direct, direct</td>
<td>3</td>
<td>3</td>
<td>24</td>
</tr>
<tr>
<td>MOV direct, @Ri</td>
<td>2</td>
<td>4</td>
<td>24</td>
</tr>
<tr>
<td>MOV direct, #data</td>
<td>3</td>
<td>3</td>
<td>24</td>
</tr>
<tr>
<td>MOV @Ri, A</td>
<td>1</td>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>MOV @Ri, direct</td>
<td>2</td>
<td>3</td>
<td>24</td>
</tr>
<tr>
<td>MOV @Ri, #data</td>
<td>2</td>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>MOV DPTR, #addr24</td>
<td>4 *</td>
<td>4</td>
<td>24</td>
</tr>
<tr>
<td>MOVC A, @A+DPTR</td>
<td>1</td>
<td>2</td>
<td>24</td>
</tr>
<tr>
<td>MOVC A, @A+PC</td>
<td>1</td>
<td>2</td>
<td>24</td>
</tr>
<tr>
<td>MOVX A, @Ri</td>
<td>1</td>
<td>2</td>
<td>24</td>
</tr>
<tr>
<td>MOVX A, @DPTR</td>
<td>1</td>
<td>2</td>
<td>24</td>
</tr>
<tr>
<td>MOVX @Ri, A</td>
<td>1</td>
<td>2</td>
<td>24</td>
</tr>
<tr>
<td>MOVX @DPTR, A</td>
<td>1</td>
<td>2</td>
<td>24</td>
</tr>
<tr>
<td>PUSH direct</td>
<td>2</td>
<td>4</td>
<td>24</td>
</tr>
<tr>
<td>POP direct</td>
<td>2</td>
<td>3</td>
<td>24</td>
</tr>
<tr>
<td>XCH A, Rn</td>
<td>1</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>XCH A, direct</td>
<td>2</td>
<td>2-3</td>
<td>12</td>
</tr>
<tr>
<td>XCH A, @Ri</td>
<td>1</td>
<td>2-3</td>
<td>12</td>
</tr>
<tr>
<td>XCHD A, @Ri</td>
<td>1</td>
<td>2-3</td>
<td>12</td>
</tr>
</tbody>
</table>

* The standard 8051 core provides the 3-byte instruction MOV DPTR,#addr16.
<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Instruction bytes</th>
<th>Execution cycles enhanced 8051 core</th>
<th>Execution cycles standard 8051 core</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR C</td>
<td>1</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>CLR bit</td>
<td>2</td>
<td>2-3</td>
<td>12</td>
</tr>
<tr>
<td>SETB C</td>
<td>1</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>SETB bit</td>
<td>2</td>
<td>2-3</td>
<td>12</td>
</tr>
<tr>
<td>CPL C</td>
<td>1</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>CPL bit</td>
<td>2</td>
<td>2-3</td>
<td>12</td>
</tr>
<tr>
<td>ANL C, bit</td>
<td>2</td>
<td>2-3</td>
<td>24</td>
</tr>
<tr>
<td>ANL C, /bit</td>
<td>2</td>
<td>2-3</td>
<td>24</td>
</tr>
<tr>
<td>ORL C, bit</td>
<td>2</td>
<td>2-3</td>
<td>24</td>
</tr>
<tr>
<td>ORL C, /bit</td>
<td>2</td>
<td>2-3</td>
<td>24</td>
</tr>
<tr>
<td>MOV C, bit</td>
<td>2</td>
<td>2-3</td>
<td>12</td>
</tr>
<tr>
<td>MOV C, /bit</td>
<td>2</td>
<td>2-3</td>
<td>24</td>
</tr>
<tr>
<td>JC rel</td>
<td>2</td>
<td>2</td>
<td>24</td>
</tr>
<tr>
<td>JNC rel</td>
<td>2</td>
<td>2</td>
<td>24</td>
</tr>
<tr>
<td>JB bit, rel</td>
<td>3</td>
<td>3-4</td>
<td>24</td>
</tr>
<tr>
<td>JNB bit, rel</td>
<td>3</td>
<td>3-4</td>
<td>24</td>
</tr>
<tr>
<td>JBC bit, rel</td>
<td>3</td>
<td>3-4</td>
<td>24</td>
</tr>
</tbody>
</table>
Table 5. Program branching execution times.

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Instruction bytes</th>
<th>Execution cycles enhanced 8051 core</th>
<th>Execution cycles standard 8051 core</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACALL page x</td>
<td>2</td>
<td>5</td>
<td>24</td>
</tr>
<tr>
<td>LCALL addr16</td>
<td>3</td>
<td>6</td>
<td>24</td>
</tr>
<tr>
<td>GCALL addr24</td>
<td>4</td>
<td>7</td>
<td>not supported</td>
</tr>
<tr>
<td>RET</td>
<td>1</td>
<td>3-4</td>
<td>24</td>
</tr>
<tr>
<td>RETI</td>
<td>1</td>
<td>3-4</td>
<td>24</td>
</tr>
<tr>
<td>AJMP page x</td>
<td>2</td>
<td>2</td>
<td>24</td>
</tr>
<tr>
<td>LJMP addr16</td>
<td>3</td>
<td>3</td>
<td>24</td>
</tr>
<tr>
<td>GJMP addr24</td>
<td>4</td>
<td>4</td>
<td>not supported</td>
</tr>
<tr>
<td>SJMP rel</td>
<td>2</td>
<td>2</td>
<td>24</td>
</tr>
<tr>
<td>JMP @A+DPTR</td>
<td>1</td>
<td>1</td>
<td>24</td>
</tr>
<tr>
<td>JZ rel</td>
<td>2</td>
<td>2</td>
<td>24</td>
</tr>
<tr>
<td>JNZ rel</td>
<td>2</td>
<td>2</td>
<td>24</td>
</tr>
<tr>
<td>CJNE A, direct, rel</td>
<td>3</td>
<td>3-4</td>
<td>24</td>
</tr>
<tr>
<td>CJNE A, #data, rel</td>
<td>3</td>
<td>3</td>
<td>24</td>
</tr>
<tr>
<td>CJNE Rn, #data, rel</td>
<td>3</td>
<td>3</td>
<td>24</td>
</tr>
<tr>
<td>CJNE @Ri, #data, rel</td>
<td>3</td>
<td>4-5</td>
<td>24</td>
</tr>
<tr>
<td>DJNZ Rn, rel</td>
<td>2</td>
<td>2</td>
<td>24</td>
</tr>
<tr>
<td>DJNZ direct, rel</td>
<td>3</td>
<td>3-4</td>
<td>24</td>
</tr>
<tr>
<td>NOP</td>
<td>1</td>
<td>1</td>
<td>12</td>
</tr>
</tbody>
</table>

Notes on the instruction set and addressing modes:

Rn     : Register addressing (n = 0..7).
@Ri    : Register-indirect addressing (i = 0..1).
direct : Direct addressing.
#data  : 8-bit constant included in instruction.
#data16 : 16-bit constant included in instruction.
#data24 : 24-bit constant included in instruction.
page x : Branching within 1 Kbyte (x = 0..3).
addr16 : Branching within 64 Kbytes.
addr24 : Branching within 16 Mbytes.
rel    : signed 8-bit offset byte for short jumps.
bit    : Direct addressed bit.
Appendix D: First IDaSS Design

The architecture described below is the first design of the enhanced 8051 core. This design is not suitable for synthesis with the ASA silicon compiler because of the following limitations:

- The external memory ready signal, implemented as IDaSS signal, cannot be converted;
- The use of asynchronous read and write RAM is not supported by ASA;
- In ASA the state machines cannot deal with the semaphore flag;
- Several operators are too complex and use prohibited IDaSS functions.

The enhanced 8051 core is compatible to the standard 8051 core which satisfies the interface specifications for IDaSS library components. Furthermore, the number of clock cycles needed to execute an instruction has been restricted to a minimum. An 8-bit data input and output bus runs through the design as needed for the I/O interface. The presence of these data buses provides the possibility of modifying a data byte within one clock cycle. Therefore, the internal RAM has to be accessed asynchronously when a read or write action is performed.

The instruction code bytes are sequentially read and during instruction execution the next op code is fetched. This method is called prefetching and results in an architecture that uses for instruction execution the same number of clock cycles as the number of instruction bytes. Exceptions are indirect register addressing, subroutine operations and the multiply and divide instruction.

The complete architecture of the processor core is schematically represented in figure 1. All the blocks are controlled by four state machines. Each state machine is activated by a set semaphore flag of specific registers. The semaphore flag is an extra register bit that is automatically set each time a load action occurs and can be tested and reset by controllers.

The four state machines are divided into the addressing modes. The advantages of four state machines above one state machine are smaller states and design surveyability. Testing of the ready signal also occurs in the state machines. The ready signal, generated by the external memories, is simulated by a pulse signal from the signal editor in IDaSS. The state machines functionality is as follows:

- DIRECT (2 states): All the direct addressing operations are handled by this state machine which is linked with register AR.
- BOOLEAN (2 states): This state machine handles the bit-manipulation instructions and is activated by register BOOL.
- REGISTER (3 states): This state machine is linked with the RAMunit temporary register TMP2 and completes the instructions that are associated with registers of the four registers banks. The temporary register TMP2 is loaded with the indirect address. If the TMP2
semaphore flag is not set, a second test is done on bit 3 of the instruction register IR. In this way the difference between register addressing and register indirect addressing mode is detected.

- **DECODER (7 states):** This is the main state machine which controls the total process of instruction fetching and execution depending on the instruction register IR contents. Furthermore the interrupt request, if present in the interrupt number register INo, are handled by this state machine. The first state is used as start up state, so after an asynchronous reset the processor core starts with fetching the first op code. One state serves as wait state to finish the multiplication and division that take eight clock cycles.

The architecture is constituted out of three units: A RAM unit to perform the internal RAM operations; an ALU unit executes the arithmetic and logical operations; and the PC unit to control the external addressing. All the units are interconnected by a switch. The switch is responsible for correct data transport between the units and the external world.
The core contains five special function registers which are placed near the operator that uses the register. The ALU is surrounded by the Accumulator, the Program Status Word and the B register. The temporary register TMP1 is used for storing immediate data and partial results during the multiply and divide operation. The Stack Pointer and Data Pointer accompany the program counter unit.
Appendix E: IDaSS Document of the Enhanced 8051 Core

IDaSS VO.08m document.

'TopLevel\TopLevel\Core80S1' is a schematic.

Bidirectional connector (25 bits) with name 'Addrbus'
Bidirectional connector ( 2 bits) with name 'Ctrlbus'
Bidirectional connector ( 8 bits) with name 'DATibus'
Bidirectional connector ( 8 bits) with name 'DATobus'
Bidirectional connector ( 1 bit) with name 'enable'
Bidirectional connector ( 2 bits) with name 'Iack'
Bidirectional connector ( 3 bits) with name 'Ireq'
Bidirectional connector ( 1 bit) with name 'Ready'
Bidirectional connector (10 bits) with name 'SFRaddr'
Bidirectional connector ( 8 bits) with name 'SFRI'
Bidirectional connector ( 8 bits) with name 'SFRO'

'TopLevel\TopLevel\Core80S1\ALU_SFR' is a schematic.

Bidirectional connector ( 8 bits) with name 'ALUout1'
Bidirectional connector ( 8 bits) with name 'ALUout2'
Bidirectional connector ( 8 bits) with name 'ALUPSW'
Bidirectional connector ( 8 bits) with name 'Ao'
Bidirectional connector ( 8 bits) with name 'Bo'
Bidirectional connector ( 1 bit) with name 'CTRL'
Bidirectional connector ( 8 bits) with name 'DirDat'
Bidirectional connector (10 bits) with name 'IR'
Bidirectional connector ( 8 bits) with name 'PSWo'
Bidirectional connector (10 bits) with name 'SFR'

'TopLevel\TopLevel\Core80S1\ALU_SFR\A' is a register.

This register is 8 bits wide and is controlled by an unnamed control input.
The default function is 'hold'.
This register is loaded with value 0 following system reset.

The value loaded for the 'reset' command is 0.

Control specification:

'TopLevel\TopLevel\Core80S1\ALU_SFR\IMUX' is an operator.

This operator has 3 functions and is controlled by an unnamed control input.
The default function is 'ALUout1'.

Control connector (10 bits) without a name

Control specification:
Section of Digital Information Systems
Enhanced 8051 Core

Output connector (8 bits) with name 'A'
Input connector (8 bits) with name 'ALUout1'
Input connector (8 bits) with name 'ALUout2'
Input connector (8 bits) with name 'Direct'

Text for function 'ALUout1' of 'TopLevel\TopLevel\Core8051\ALU_SFR\AiMUX':

*ALU output, moves included.*
A := ALUout1.

Text for function 'Direct' of 'TopLevel\TopLevel\Core8051\ALU_SFR\AiMUX':

*Direct addressing.*
A := Direct.

Text for function 'XCH' of 'TopLevel\TopLevel\Core8051\ALU_SFR\AiMUX':

*XCH instruction.*
A := ALUout2.

'TopLevel\TopLevel\Core8051\ALU_SFR\Awop' is an operator.
The default function is 'Direct'.

Control connector (10 bits) without a name

Control specification:

*Select register A control.*
\%1011100000. 'MOVX A,@DPTR'
\%10111001x. 'MOVX A,0Ri'
\%1010000011. 'MOVX A,0A-PC'
\%1010001001. 'MOVX A,#0A-PC'
\%0101110100. 'MOVX A,#data'
\%0111100010. 'CLR A'
\%0111100101. 'MOVX A.dire'
\%011110011x. 'MOVX A,#0Ri'
\%01111101xx. 'MOVX A,Rn'
\%1010000011. 'RR A'
\%0100010011. 'RRC A'
\%1010001101. 'RL A'
\%1010001110. 'RLC A'
\%1000000000. 'INCA A'
\%1010010010. 'DECA A'
\%01001001xx. 'ADD A'
\%01010101xx. 'ADD A'
\%10111101xx. 'ADDC A'
\%01011101xx. 'ADDC A'
\%01010001xx. 'ORL A'
\%01010011xx. 'ORL A'
\%01010101xx. 'ANL A'
\%10101101xx. 'ANL A'
\%01011101xx. 'XRL A'
\%10101111xx. 'XRL A'
\%1011000100. 'DIV AB'
\%1010000100. 'DIV AB last cycle'
\%01110101xx. 'SUBB A'
\%0110101xxx. 'SUBB A'
\%0110010100. 'MUL AB'
\%1000101000. 'MUL AB last cycle'
\%0111010100. 'SWAP A'
\%1011110100. 'DA A'
\%1011111010. 'CPL A'
\%0111000101. 'XCH A,dir'
\%110100011x. 'XCH A,#0Ri'
\%0111010111x. 'XCHD A,#0Ri'
\%01111001xxx Wrt. 'XCH A,Rn'

Output connector (1 bit) with name 'AW'
Input connector (1 bit) with name 'CTRL'
Input connector (1 bit) with name 'DirAW'

Text for function 'Direct' of 'TopLevel\TopLevel\Core8051\ALU_SFR\Awop':

*Direct addressing.*
AW := DirAW \CTRL.
Text for function 'Wrt' of 'TopLevel\TopLevel\Core8051\ALU_SFR\AWop':
----------------------v----------------------
"Write register A."
AW := 1 ones \ CTRL.

'TopLevel\TopLevel\Core8051\ALU_SFR\B' is a register.
This register is 8 bits wide and is controlled by an unnamed control input.
The default function is 'hold'.
This register is loaded with value 0 following system reset.
The value loaded for the 'reset' command is 0.

Control specification:
-------------------------------v-------------------------------
%! load. "Write new value in register B."

'TopLevel\TopLevel\Core8051\ALU_SFR\Bmux' is an operator.
This operator has 2 functions and is controlled by an unnamed control input.
The default function is 'Direct'.

Control connector (10 bits) without a name
Control specification:
-------------------------------v-------------------------------
"Select register B input."
i0100x0100 M_D. "Multiplication or division."
Input connector ( 8 bits) with name 'ALUout2'
Output connector ( 8 bits) with name 'B'
Input connector ( 8 bits) with name 'Direct'

Text for function 'Direct' of 'TopLevel\TopLevel\Core8051\ALU_SFR\Bmux':
-------------------------------v-------------------------------
"Direct addressing."
B := Direct.

Text for function 'M_D' of 'TopLevel\TopLevel\Core8051\ALU_SFR\Bmux':
-------------------------------v-------------------------------
"Multiplication or division."
B := ALUout2.

'TopLevel\TopLevel\Core8051\ALU_SFR\Bwop' is an operator.
This operator has 2 functions and is controlled by an unnamed control input.
The default function is 'Normal'.

Control connector (10 bits) without a name
Control specification:
-------------------------------v-------------------------------
"Activate register B access for multiplication or division."
i0100x0100 M_D. "MUL or DIV"
Output connector ( 1 bit ) with name 'BW'
Input connector ( 1 bit ) with name 'OirBW'

Text for function 'M_D' of 'TopLevel\TopLevel\Core8051\ALU_SFR\Bwop':
-------------------------------v-------------------------------
"Multiplication or division."
BW := 1 ones. "Write high byte result register B."

Text for function 'Normal' of 'TopLevel\TopLevel\Core8051\ALU_SFR\Bwop':
-------------------------------v-------------------------------
"Wait for ALU active signal."
BW := OirBW.

'TopLevel\TopLevel\Core8051\ALU_SFR\Dirop' is an operator.
This operator has 2 functions and is controlled by an unnamed control input.
The default function is 'Normal'.

- 41 -
Control connector (10 bits) without a name

Control specification:
-----------------------------------------------------------------------------
*Direct addressing to register A."
(0..7)
%11100000 A.
-----------------------------------------------------------------------------
Output connector ( 1 bit ) with name ‘AW’
Input connector (10 bits) with name ‘SFR’

Text for function ‘A’ of ‘TopLevel\TopLevel\Core8051\ALU_SFR\DirAop’:
-----------------------------------------------------------------------------
*Register A access."
AW := SFR at: 8.
-----------------------------------------------------------------------------

Text for function ‘Normal’ of ‘TopLevel\TopLevel\Core8051\ALU_SFR\DirAop’:
-----------------------------------------------------------------------------
*No direct addressing to register A."
AW := 1 zeroes.
-----------------------------------------------------------------------------

‘TopLevel\TopLevel\Core8051\ALU_SFR\DirBop’ is an operator.
This operator has 2 functions and is controlled by an unnamed control input. The default function is ‘Normal’.

Control connector (10 bits) without a name

Control specification:
-----------------------------------------------------------------------------
*Direct addressing to register B."
(0..7)
%11110000 B.
-----------------------------------------------------------------------------
Output connector ( 1 bit ) with name ‘BW’
Input connector (10 bits) with name ‘SFR’

Text for function ‘B’ of ‘TopLevel\TopLevel\Core8051\ALU_SFR\DirBop’:
-----------------------------------------------------------------------------
*Register B access."
BW := SFR at: 8.
-----------------------------------------------------------------------------

Text for function ‘Normal’ of ‘TopLevel\TopLevel\Core8051\ALU_SFR\DirBop’:
-----------------------------------------------------------------------------
*No direct addressing to register B."
BW := 1 zeroes.
-----------------------------------------------------------------------------

‘TopLevel\TopLevel\Core8051\ALU_SFR\DirPSWop’ is an operator.
This operator has 2 functions and is controlled by an unnamed control input. The default function is ‘Normal’.

Control connector (10 bits) without a name

Control specification:
-----------------------------------------------------------------------------
*Direct addressing to register PSW."
(0..7)
%11010000 PSW.
-----------------------------------------------------------------------------
Output connector ( 1 bit ) with name ‘PSWW’
Input connector (10 bits) with name ‘SFR’

Text for function ‘Normal’ of ‘TopLevel\TopLevel\Core8051\ALU_SFR\DirPSWop’:
-----------------------------------------------------------------------------
*No direct addressing to register PSW."
PSWW := 1 zeroes.
-----------------------------------------------------------------------------

Text for function ‘PSW’ of ‘TopLevel\TopLevel\Core8051\ALU_SFR\DirPSWop’:
-----------------------------------------------------------------------------
*Register PSW access."
PSWW := SFR at: 8.
-----------------------------------------------------------------------------

‘TopLevel\TopLevel\Core8051\ALU_SFR\P’ is a register.
This register is 1 bit wide. The default function is 'load'.
The default value loaded with value 0 following system reset.
The value loaded for the 'reset' command is 0.

'TopLevel\TopLevel\Core8051\ALU_SFR\Pop' is an operator.

This operator has 1 function. The default function is 'Parity'.

Text for function 'Parity' of 'TopLevel\TopLevel\Core8051\ALU_SFR\Pop':

"Generate parity bit."

\[ P := (A \text{ at: } 0) \ll (A \text{ at: } 1) \ll (A \text{ at: } 2) \ll (A \text{ at: } 3) \ll (A \text{ at: } 4) \ll (A \text{ at: } 5) \ll (A \text{ at: } 6) \ll (A \text{ at: } 7). \]

'TopLevel\TopLevel\Core8051\ALU_SFR\Pop' is a register.

This register is 7 bits wide. The default function is 'load'.
The default value loaded with value 0 following system reset.
The value loaded for the 'reset' command is 0.

'TopLevel\TopLevel\Core8051\ALU_SFR\PSWiMUX' is an operator.

This operator has 2 functions and is controlled by an unnamed control input. The default function is 'ALU'.

Control connector (1 bit) without a name

Control specification:

*Select register PSW input.*

\%1 Direct. "Direct addressing mode"

Input connector (8 bits) with name 'ALU'
Input connector (8 bits) with name 'Direct'

Output connector (8 bits) with name 'PSW'

Text for function 'ALU' of 'TopLevel\TopLevel\Core8051\ALU_SFR\PSWiMUX':

"ALU input."

\[ \text{PSW} := \text{ALU}. \]

Text for function 'Direct' of 'TopLevel\TopLevel\Core8051\ALU_SFR\PSWiMUX':

"Direct addressing."

\[ \text{PSW} := \text{Direct}. \]

'TopLevel\TopLevel\Core8051\ALU_SFR\PSWWop' is an operator.

This operator has 1 function. The default function is 'PSWmod'.

Input connector (7 bits) with name 'in'
Input connector (7 bits) with name 'out'
Input connector (1 bit) with name 'P'
Input connector (8 bits) with name 'PSWi'
Input connector (8 bits) with name 'PSWo'
Input connector (1 bit) with name 'PSW'

Text for function 'PSWmod' of 'TopLevel\TopLevel\Core8051\ALU_SFR\PSWWop':

*Register PSW modify.*

\[ \text{out} := \text{PSW} \text{ if0: in 'Old value.'}
\text{if1: (PSWi from: 1 to: ?). 'New value.'}
\text{PSWo} := \text{in}. \text{P}. \]

'TopLevel\TopLevel\Core8051\ALU_SFR\PSWWop' is an operator.
This operator has 2 functions and is controlled by an unnamed control input. The default function is 'Direct'.

Control connector (10 bits) without a name

Control specification:

<table>
<thead>
<tr>
<th>Specification</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%0110100000</td>
<td>ORL C, /bit</td>
</tr>
<tr>
<td>%0110110000</td>
<td>ANL C, /bit</td>
</tr>
<tr>
<td>%0110000010</td>
<td>ORL C,bit</td>
</tr>
<tr>
<td>%0110100110</td>
<td>ANL C,bit</td>
</tr>
<tr>
<td>%0100010011</td>
<td>MOV C,bit</td>
</tr>
<tr>
<td>%0100011011</td>
<td>RRC A</td>
</tr>
<tr>
<td>%0100110011</td>
<td>RLC A</td>
</tr>
<tr>
<td>%0110110011</td>
<td>CPL C</td>
</tr>
<tr>
<td>%0111000011</td>
<td>CLR C</td>
</tr>
<tr>
<td>%0111010011</td>
<td>SETB C</td>
</tr>
<tr>
<td>%01110101xx</td>
<td>ADD</td>
</tr>
<tr>
<td>%01100111xx</td>
<td>ADDC</td>
</tr>
<tr>
<td>%01110111xx</td>
<td>SUBB</td>
</tr>
<tr>
<td>%01110111xx</td>
<td>SUBB</td>
</tr>
<tr>
<td>%01111011xx</td>
<td>CJNNE</td>
</tr>
<tr>
<td>%01111011xx</td>
<td>CJNE</td>
</tr>
<tr>
<td>%1010000100</td>
<td>DIV AB and MUL AB last cycle</td>
</tr>
</tbody>
</table>

Text for function 'Direct' of 'TopLevel\TopLevel\Core8051\ALU_SFR\PSWWop':

```
  No register PSW operation.
  PSWW := DirPSWW \ CTRL.
```

Text for function 'Wrt' of 'TopLevel\TopLevel\Core8051\ALU_SFR\PSWWop':

```
  Write register PSW.
  PSWW := 1 ones \ CTRL.
```

'TopLevel\TopLevel\Core8051\ALU_unit' is a schematic.

This operator has 3 functions and is controlled by an unnamed control input. The default function is 'A'.

Control connector (10 bits) without a name

Control specification:

<table>
<thead>
<tr>
<th>Specification</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%0100010000</td>
<td>JBC bit.rel</td>
</tr>
<tr>
<td>%0100010000</td>
<td>JB bit.rel</td>
</tr>
<tr>
<td>%0100011000</td>
<td>JNB bit.rel</td>
</tr>
<tr>
<td>%0110100000</td>
<td>ORL C,/bit</td>
</tr>
<tr>
<td>%0111010000</td>
<td>ANL C, /bit</td>
</tr>
<tr>
<td>%0101110010</td>
<td>ORL C,bit</td>
</tr>
<tr>
<td>%0101000010</td>
<td>ANL C,bit</td>
</tr>
<tr>
<td>%0111100010</td>
<td>MOV C,bit</td>
</tr>
<tr>
<td>%0111000010</td>
<td>ANL C,bit</td>
</tr>
<tr>
<td>%0110101010</td>
<td>MOV bit,C</td>
</tr>
</tbody>
</table>

Enhanced 8051 Core
Section of Digital Information Systems
Enhanced 8051 Core

%0110100010, "MOV C.bit"
%0111000010, "CPL bit"
%0111010010, "CLR bit"
%0101000011, "ORL dir,.#data"
%0101010011, "ANL dir,.#data"
%0110110011, "XRL dir,.#data"
%0100000010, "INC dir"
%0100010101, "DEC dir"
%0101110101, "MOV dir,.#data"
%0111010101 DIR, "DJNZ dir,.rel"
%0100000011, "INC @Ri"
%0100010101, "DEC @Ri"
%0101110101, "MOV @Ri,.#data"
%0111011000, "CJNE @Ri,.#data,.rel"
%01011110xx, "MOV Rn,.#data"
%01110111xx, "CJNE Rn,.#data,.rel"
%01101110xx RAM, "DJNZ Rn,.rel"

----------------------v----------------------

Input connector ( 8 bits) with name ‘A’
Input connector ( 8 bits) with name ‘Direct’
Output connector ( 8 bits) with name ‘in1’
Input connector ( 8 bits) with name ‘RAM’

Text for function ‘A’ of ‘TopLevel\TopLevel\Core8051\ALUunit\ALUIn1MUX’:
in1 := A.

Text for function ‘DIR’ of ‘TopLevel\TopLevel\Core8051\ALUunit\ALUIn1MUX’:
in1 := Direct.

Text for function ‘RAM’ of ‘TopLevel\TopLevel\Core8051\ALUunit\ALUIn1MUX’:
in1 := RAM.

This operator has 4 functions and is controlled by an unnamed control input.
The default function is ‘RAM’.

Control connector (10 bits) without a name

Control specification:

*************************
%1010000011, "MOVC A,@A+PC"
%1010010011, "MOVC A,@A+DPTR"
%01110111xx, "MOV @Ri,.#data"
%01101111xx, "CJNE @Ri,.#data,.rel"
%0101111xxx, "MOV Rn,.#data"
%0111011xxx RAM, "DJNZ Rn,.rel"
%01101110xx, "CJNE Rn,.#data,.rel"
%0110010010, "ORL dir,.#data"
%0101010010, "ANL dir,.#data"
%0101100010, "XRL dir,.#data"

- 45 -
%0101010101, "ANL A, dir"
%0110010101, "CJNE A, dir, rel"
%0111000101, "XCH A, dir"
%0111100101, "MOV A, dir"
%010101011x, "MOV Rn,dir (second cycle)"

Input connector (8 bits) with name 'B'
Input connector (8 bits) with name 'Direct'
Input connector (8 bits) with name 'IM'
Output connector (8 bits) with name 'in2'
Input connector (8 bits) with name 'RAM'

Text for function 'B' of 'TopLevel\TopLevel\Core8051\ALUunit\ALUin2MUX':
in2 := B.

Text for function 'DIR' of 'TopLevel\TopLevel\Core8051\ALUunit\ALUin2MUX':
in2 := Direct.

Text for function '1M' of 'TopLevel\TopLevel\Core8051\ALUunit\ALUin2MUX':
in2 := 1M.

Text for function 'RAM' of 'TopLevel\TopLevel\Core8051\ALUunit\ALUin2MUX':
in2 := RAM.

'TopLevel\TopLevel\Core8051\ALUunit\ALUs' is a schematic.

Bidirectional connector (8 bits) with name 'in1'
Bidirectional connector (8 bits) with name 'in2'
Bidirectional connector (10 bits) with name 'IR'
Bidirectional connector (8 bits) with name 'Jstl'
Bidirectional connector (8 bits) with name 'Jst2'
Bidirectional connector (8 bits) with name 'out1'
Bidirectional connector (8 bits) with name 'out2'
Bidirectional connector (8 bits) with name 'FSWI'
Bidirectional connector (8 bits) with name 'PSWi'

'TopLevel\TopLevel\Core8051\ALUunit\ALUs\ALU1' is an operator.

This operator has 8 functions and is controlled by an unnamed control input.
The default function is 'DEC'.

Control connector (10 bits) without a name

Control specification:

"Select ALU function."
%0101110000 JZ, "JZ rel"
%0111100000 JNZ, "JNZ rel"
%0100000000 SJMP, "SJMP rel"
%010000001x, "INC"
%01001001xx, "ADD"
%0100101xxx ADDC.
%01001101xx, "SUBB"
%0100111xxx SUBB.

Input connector (8 bits) with name 'in1'
Input connector (8 bits) with name 'in2'
Output connector (1 bit) with name 'Jst1'
Output connector (8 bits) with name 'out1'
Input connector (8 bits) with name 'FSWi'
Output connector (8 bits) with name 'PSWo'

Text for function 'ADD' of 'TopLevel\TopLevel\Core8051\ALUunit\ALUs\ALU1':

"ADD A,in2."
_sum := (1 zeroes, in1, 1 zeroes) + (1 zeroes, in2, 1 zeroes).
out1 := (_sum from: 1 to: 8).
PSWo := (_sum at: 9), ((in1 at: 4) >> (in2 at: 4) >> (_sum at: 5)).
Section of Digital Information Systems

Enhanced 8051 Core

(PSWi from: 3 to: 5), ((in1 at: 7) < (in2 at: 7)) \ ((in1 at: 7) > (Sum at: 8)),
(PSWi from: 0 to: 1).

----------------------------------------

Text for function 'ADDe' of 'TopLevel\TopLevel\Core8051\ALUunit\ALUs\ALU1':

• ADDC A, in2.
  _Sum := (1 zeroes, in1, (PSWi at: 7)) * (1 zeroes, in2, (PSWi at: 7)).
  out1 := (_Sum from: 1 to: 8).
  PSWo := (_Sum at: 9), ((in1 at: 4) < (in2 at: 4) > (_Sum at: 5)),
  (PSWi from: 3 to: 5), ((in1 at: 7) < (in2 at: 7)) \ ((in1 at: 7) > (_Sum at: 8)),
  (PSWi from: 0 to: 1).

----------------------------------------

Text for function 'DEC' of 'TopLevel\TopLevel\Core8051\ALUunit\ALUs\ALU1':

• DEC in1.
  _Sum := (1 zeroes, in1, 1 ones) + (1 zeroes, 7 ones, 1 zeroes, 1 ones).
  out1 := _Sum from: 1 to: 8.

   Activate jump for DJNZ.

   Jtst := (_Sum at: 1) \ (_Sum at: 2) \ (_Sum at: 3) \ (_Sum at: 4) \ (_Sum at: 5) \ (_Sum at: 6) \ (_Sum at: 7) \ (_Sum at: 8).

----------------------------------------

Text for function 'INC' of 'TopLevel\TopLevel\Core8051\ALUunit\ALUs\ALU1':

• INC in1.
  _Sum := (1 zeroes, in1, 1 zeroes) + (8 zeroes, 1 ones, 1 zeroes).
  out1 := _Sum from: 1 to: 8.

----------------------------------------

Text for function 'JNZ' of 'TopLevel\TopLevel\Core8051\ALUunit\ALUs\ALU1':

• JNZ
  Jtst := (in1 at: 0) \ (in1 at: 1) \ (in1 at: 2) \ (in1 at: 3) \ (in1 at: 4) \ (in1 at: 5) \ (in1 at: 6) \ (in1 at: 7).

----------------------------------------

Text for function 'JZ' of 'TopLevel\TopLevel\Core8051\ALUunit\ALUs\ALU1':

• JZ
  Jtst := (in1 at: 0) \ (in1 at: 1) \ (in1 at: 2) \ (in1 at: 3) \ (in1 at: 4) \ (in1 at: 5) \ (in1 at: 6) \ (in1 at: 7) not.

----------------------------------------

Text for function 'SJMP' of 'TopLevel\TopLevel\Core8051\ALUunit\ALUs\ALU1':

• SJMP
  Jtst := 1 ones.

----------------------------------------

Text for function 'SUBB' of 'TopLevel\TopLevel\Core8051\ALUunit\ALUs\ALU1':

• SUBB A, in2.
  _Sum := (1 zeroes, in1, (PSWi at: 7) not) + (1 zeroes, in2 not, (PSWi at: 7) not).
  out1 := (_Sum from: 1 to: 8).
  PSWo := (_Sum at: 9) not, ((in1 at: 4) < (in2 at: 4) > (_Sum at: 5)),
  (PSWi from: 3 to: 5), ((in1 at: 7) < (in2 at: 7) not) \ ((in1 at: 7) > (_Sum at: 8)),
  (PSWi from: 0 to: 1).

===========================================================================

'TopLevel\TopLevel\Core8051\ALUunit\ALUs\ALU2' is an operator.

This operator has 7 functions and is controlled by an unnamed control input.
The default function is ‘MOV’.

Control connector (10 bits) without a name

Control specification:

• Select ALU function.
  %0110000100 DIV. "DIV AB"
  %0100000100 DIV. "DIV AB last cycle"
  %0110000100 MUL. "MUL AB"
  %0100010011 ADD. "ADD A"
  %0110010110 MUL. "MUL AB last cycle"
  %0100010011 RRC. "RRC A"
  %0110010011 RLC. "RLC A"
  %0100000011 RR. "RR A"
  %0100010011 RL. "RL A"

Input connector (8 bits) with name ‘in1’
Input connector (8 bits) with name 'in2'
Output connector (8 bits) with name 'out1'
Output connector (8 bits) with name 'out2'
Input connector (8 bits) with name 'PSWi'
Output connector (8 bits) with name 'PSWo'
Input connector (8 bits) with name 'TMP'

Text for function 'DIV' of 'TopLevel\TopLevel\Core8051\ALUunit\ALUs\ALU2':

```
DIV AB
_Sum := (TMP, (inl at: 7), 1 ones) + (1 zeroes, in2 not, 1 ones).
out1 := (inl from: 0 to: 6), (_Sum at: 9).
out2 := (_Sum at: 9) if1: (_Sum from: 1 to: 8)
       1f0: (TMP from: 0 to: 6), (in1 at: 7).

"Update PSW only in the last cycle"
PSWo := 1 zeroes, (PSWi from: 3 to: 6), ((in2 at: 0) /
     (in2 at: 1) /
     (in2 at: 2) /
     (in2 at: 3) /
     (in2 at: 4) /
     (in2 at: 5) /
     (in2 at: 6) /
     (in2 at: 7)) not,
     (PSWi from: 0 to: 1).
```

Text for function 'MOV' of 'TopLevel\TopLevel\Core8051\ALUunit\ALUs\ALU2':

```
MOV operation.
out1 := in2.
```

Text for function 'MUL' of 'TopLevel\TopLevel\Core8051\ALUunit\ALUs\ALU2':

```
MUL AB
_Test := (inl at: 0) if0: 8 zeroes
         if1: inl.
_Sum := (1 zeroes, TMP, 1 zeroes) + (1 zeroes, _Test, 1 zeroes).
out1 := (_Sum at: 1), (inl from: 1 to: 7).
out2 := (_Sum from: 2 to: 9).

"Update PSW only in the last cycle"
PSWo := 1 zeroes, (PSWi from: 3 to: 6), (_Sum at: 2) /
     (_Sum at: 3) /
     (_Sum at: 4) /
     (_Sum at: 5) /
     (_Sum at: 6) /
     (_Sum at: 7) /
     (_Sum at: 8) /
     (_Sum at: 9),
     (PSWi from: 0 to: 1).
```

Text for function 'RL' of 'TopLevel\TopLevel\Core8051\ALUunit\ALUs\ALU2':

```
RL A
out1 := (inl from: 0 to: 6), (inl at: 7).
```

Text for function 'RLC' of 'TopLevel\TopLevel\Core8051\ALUunit\ALUs\ALU2':

```
RLC A
out1 := (inl from: 0 to: 6), (PSWi at: 7).
PSWo := (inl at: 7), (PSWi from: 0 to: 6).
```

Text for function 'RR' of 'TopLevel\TopLevel\Core8051\ALUunit\ALUs\ALU2':

```
RR A
out1 := (inl at: 0), (inl from: 1 to: 7).
```

Text for function 'RRC' of 'TopLevel\TopLevel\Core8051\ALUunit\ALUs\ALU2':

```
RRC A
out1 := (PSWi at: 7), (inl from: 1 to: 7).
PSWo := (inl at: 0), (PSWi from: 0 to: 6).
```

This operator has 9 functions and is controlled by an unnamed control input. The default function is 'XCH'.

Control connector (10 bits) without a name

Control specification:

'Select ALU function.'
%011010100 DA. "DA A"
%010100001x. "ORL dir"
%01010001xx. "ORL A"
%0101001xxx ORL. "ORL A,Rn"
%0101001001x. "ANL dir"
%01010101xxx. "ANL A"
%01010101001x. "ANL A,Rn"
%01011000101x. "XRL dir"
%01011000100x. "XRL A"
%010110001010x. "XRL A,Rn"
%01011000100x. "XRL A,Rn"
%011100010x. "SWAP. "SWAP A"
%0111100100x. "CLR. "CLR A"
%0111100100x. "CLR A"
%0111110100x. "CPL. "CPL A"
%01110001000x. "XCHD. "XCHD A,@Ri"

Input connector ( 8 bits) with name 'in1'
Input connector ( 8 bits) with name 'in2'
Output connector ( 8 bits) with name 'out1'
Output connector ( 8 bits) with name 'out2'
Input connector ( 8 bits) with name 'PSWi'
Output connector ( 8 bits) with name 'PSWo'

Text for function 'ANL' of 'TopLevel\TopLevel\Core8051\ALUunit\ALUs\ALU3':
"Logical AND operation."
out1 := in1 \ in2.

Text for function 'CLR' of 'TopLevel\TopLevel\Core8051\ALUunit\ALUs\ALU3':
"CLR A."
out1 := 8 zeroes.

Text for function 'CPL' of 'TopLevel\TopLevel\Core8051\ALUunit\ALUs\ALU3':
"CPL A."
out1 := in1 not.

Text for function 'DA' of 'TopLevel\TopLevel\Core8051\ALUunit\ALUs\ALU3':
"DA A."
_Test1a := (in1 at: 3) if0: 1 zeroes
if1: ((in1 at: 1) \ (in1 at: 2)).
_Test1b := (PSWi at: 6) \ (in1 at: 2).
_Add1 := (in1 from: 0 to: 3) \ (PSWi at: 6).
_Low := _Test1b if1: _Add1
if0: (in1 from: 0 to: 3).
_Test2a := (in1 at: 3) if0: 1 zeroes
if1: ((in1 at: 5) \ (in1 at: 6)).
_Test2b := (PSWi at: 7) \ (in1 at: 6).
_Add2 := (in1 from: 4 to: 7) \ (PSWi at: 7).
_High := _Test2b if1: _Add2
if0: (in1 from: 4 to: 7).
out1 := _High, _Low.
PSWo := (_Test2a \ (PSWi at: 7)) \ (PSWi from: 0 to: 6).

Text for function 'ORL' of 'TopLevel\TopLevel\Core8051\ALUunit\ALUs\ALU3':
"Logic OR operation."
out1 := in1 \ in2.

Text for function 'SWAP' of 'TopLevel\TopLevel\Core8051\ALUunit\ALUs\ALU3':
"SWAP A."
out1 := (in1 from: 0 to: 3), (in1 from: 4 to: 7).

Text for function 'XCH' of 'TopLevel\TopLevel\Core8051\ALUunit\ALUs\ALU3':
"XCH operation."
out1 := in1.
out2 := in2.

Text for function 'XCHD' of 'TopLevel\TopLevel\Core8051\ALUunit\ALUs\ALU3':
"XCHD A,@Ri."
out1 := (in2 from: 4 to: 7), (in1 from: 0 to: 3).
out2 := (in1 from: 4 to: 7), (in2 from: 0 to: 3).
Text for function 'XRL' of 'TopLevel\TopLevel\Core8051\ALUunit\ALUs\ALU3':

-Exclusive OR operation.-

out1 := in1 XOR in2.

'TopLevel\TopLevel\Core8051\ALUunit\ALUs\ALU3' is an operator.

This operator has 1 function.
The default function is 'CJNE'.

Input connector (8 bits) with name 'in1'
Input connector (8 bits) with name 'in2'
Output connector (1 bit) with name 'Jtst'
Input connector (8 bits) with name 'JtstMUX'
Output connector (8 bits) with name 'PSWo'

Text for function 'CJNE' of 'TopLevel\TopLevel\Core8051\ALUunit\ALUs\ALU4':

'CJNE in1, in2, Jtst := ((in1 at: 0) XOR (in2 at: 0)) \ ((in1 at: 1) XOR (in2 at: 1)) \ ((in1 at: 2) XOR (in2 at: 2)) \ ((in1 at: 3) XOR (in2 at: 3)) \ ((in1 at: 4) XOR (in2 at: 4)) \ ((in1 at: 5) XOR (in2 at: 5)) \ ((in1 at: 6) XOR (in2 at: 6)) \ ((in1 at: 7) XOR (in2 at: 7))

'Carry is set if in1<in2 else cleared.'

PSWo := (in1<in2), (PSWi from: 0 to: 6).

'TopLevel\TopLevel\Core8051\ALUunit\ALUs\ALUJtstMUX' is an operator.

This operator has 2 functions and is controlled by an unnamed control input.
The default function is 'ALU1'.

Control connector (10 bits) without a name

Control specification:

-Select ALU output.-
%01000001xx, • INC'
%0100001xxx, • DEC'
%01000101xx, • DJNZ'
%0100011xxx, • ADD'
%01100101xx, • SUBB'
%01001001xx, • ORL dir'
%0100101xxx, • ADDC'
%01001101xx, • ADDC'

Text for function 'ALU1' of 'TopLevel\TopLevel\Core8051\ALUunit\ALUs\ALUJtstMUX':

'ALU1 operation.'
Jtst := ALU1.

Text for function 'ALU4' of 'TopLevel\TopLevel\Core8051\ALUunit\ALUs\ALUJtstMUX':

'ALU3 operation.'

'TopLevel\TopLevel\Core8051\ALUunit\ALUs\ALUoutMUX' is an operator.

This operator has 3 functions and is controlled by an unnamed control input.
The default function is 'ALU2'.

Control connector (10 bits) without a name

Control specification:

-Select ALU output.-
%010000001xx, • INC'
%01000001xxx, • INC'
%010000100xx, • ADD'
%010000110xx, • ORL dir'
%01010001xx, "ORL A"
%0101001xxx, "ORL A,Rn"
%010101001x, "ANL dir"
%01010101xx, "ANL A"
%0101011xxx, "ANL A, Rn"
%01110000100, "SWAP A"
%01111000100, "CLR A"
%0111110100, "CPL A"
%01110000101, "XCH A,dir"
%0111000011x, "XCH A,@Ri"
%0111001xxx, "XCH A,Rn"
%011101011x ALU3. 'XCHD

Input connector (8 bits) with name 'ALU1'
Input connector (8 bits) with name 'ALU2'
Input connector (8 bits) with name 'ALU3'
Output connector (8 bits) with name 'out1'

Text for function 'ALU1' of 'TopLevel\TopLevel\Core8051\ALUunit\ALUs\ALUoutlMUX':

"ALU1 operation."
out1 := ALU1.

Text for function 'ALU2' of 'TopLevel\TopLevel\Core8051\ALUunit\ALUs\ALUout2MUX':

"ALU2 operation."
out2 := ALU2.

Text for function 'ALU3' of 'TopLevel\TopLevel\Core8051\ALUunit\ALUs\ALUout3MUX':

"ALU3 operation."
out3 := ALU3.

Control connector (10 bits) without a name

Control specification:

"Select ALU output."
$01010000100, "DIV AB last cycle"
$0101010000100 ALU2. "MUL AB last cycle"

Text for function 'ALU2' of 'TopLevel\TopLevel\Core8051\ALUunit\ALUs\ALUout2MUX':

"ALU2 operation."
out2 := ALU2.

Text for function 'ALU3' of 'TopLevel\TopLevel\Core8051\ALUunit\ALUs\ALUout3MUX':

"ALU3 operation."
out3 := ALU3.

Control connector (10 bits) without a name

Control specification:

"Select ALU output."
$01010000100, "DIV AB last cycle"
$0101010000100 ALU2. "MUL AB last cycle"
$0111010100 ALU3. "DA A"
%01101101xx
%0110111xxx ALU4."CJNE"
----------------------~----------------------
Enhanced 8051 Core
Input connector (8 bits) with name 'ALU1'
Input connector (8 bits) with name 'ALU2'
Input connector (8 bits) with name 'ALU3'
Input connector (8 bits) with name 'ALU4'
Output connector (8 bits) with name 'PSW0'

Text for function 'ALU1' of 'TopLevel\TopLevel\Core8051\ALUunit\ALUs\ALUPSWoMUX':
------------------------------v-------------------------------
"ALU1 operation."
PSW0 := ALU1.
------------------------------v-------------------------------

Text for function 'ALU2' of 'TopLevel\TopLevel\Core8051\ALUunit\ALUs\ALUPSWoMUX':
------------------------------v-------------------------------
"ALU2 operation."
PSW0 := ALU2.
------------------------------v-------------------------------

Text for function 'ALU3' of 'TopLevel\TopLevel\Core8051\ALUunit\ALUs\ALUPSWoMUX':
------------------------------v-------------------------------
"ALU3 operation."
PSW0 := ALU3.
------------------------------v-------------------------------

Text for function 'ALU4' of 'TopLevel\TopLevel\Core8051\ALUunit\ALUs\ALUPSWoMUX':
------------------------------v-------------------------------
"ALU4 operation."
PSW0 := ALU4.
------------------------------v-------------------------------

'TopLevel\TopLevel\Core8051\ALUunit\ALUs\CNTR' is a register.
This register is 3 bits wide.
The default function is 'hold'.
This register is loaded with value 0 following system reset.
The value loaded for the 'reset' command is 0.

'TopLevel\TopLevel\Core8051\ALUunit\ALUs\TMP' is a register.
This register is 8 bits wide.
The default function is 'hold'.
This register is loaded with value 0 following system reset.
The value loaded for the 'reset' command is 0.

'TopLevel\TopLevel\Core8051\ALUunit\BOOL' is an operator.
This operator has 12 functions and is controlled by an unnamed control input.
The default function is 'JB'.

Control connector (10 bits) without a name
Control specification:
------------------------------v-------------------------------
"Select boolean operation."
%0100010000 JB.
%0100100000 JB.
%0100110000 JNB.
%0101100000 JC.
%0101110000 JNC.
%0110000000 RLnot.
%0110010000 ANLnot.
%0110100000 ORL.
%0110110000 ANL.
%0111010000 MOV.
%0111010010 CPL.
%0111100010 CLR.
%0111100101 SETB.
------------------------------v-------------------------------

Input connector (1 bit) with name 'Biti'
Output connector (1 bit) with name 'Bito'
Input connector (8 bits) with name 'PSTO'
Output connector (8 bits) with name 'PSW0'

Text for function 'ANL' of 'TopLevel\TopLevel\Core8051\ALUunit\BOOL':

- 52 -
This operator has 8 functions and is controlled by an unnamed control input. The default function is 'UNK'.

'Enhanced 8051 Core'
Control connector ( 8 bits) without a name

Control specification:

- Select bit operator.

\[(0..2)\]

\$0 Bit0.
\$1 Bit1.
\$2 Bit2.
\$3 Bit3.
\$4 Bit4.
\$5 Bit5.
\$6 Bit6.
\$7 Bit7.

Input connector ( 1 bit ) with name 'Biti'
Output connector ( 1 bit ) with name 'Bito'
Input connector ( 8 bits) with name 'inl'
Output connector ( 8 bits) with name 'outl'

Text for function 'Bit0' of 'TopLevel\TopLevel\Core8051\ALUunit\BYTEop':

- Bit 0 operation.

\[\text{Bito := inl at: 0.}\]
\[\text{outl := (inl from: 1 to: 7), Biti.}\]

Text for function 'Bit1' of 'TopLevel\TopLevel\Core8051\ALUunit\BYTEop':

- Bit 1 operation.

\[\text{Bito := inl at: 1.}\]
\[\text{outl := (inl from: 2 to: 7), Biti, (inl at: 0).}\]

Text for function 'Bit2' of 'TopLevel\TopLevel\Core8051\ALUunit\BYTEop':

- Bit 2 operation.

\[\text{Bito := inl at: 2.}\]
\[\text{out1 := (inl from: 3 to: 7), Biti, (inl from: 0 to: 1).}\]

Text for function 'Bit3' of 'TopLevel\TopLevel\Core8051\ALUunit\BYTEop':

- Bit 3 operation.

\[\text{Bito := inl at: 3.}\]
\[\text{out1 := (inl from: 4 to: 7), Biti, (inl from: 0 to: 2).}\]

Text for function 'Bit4' of 'TopLevel\TopLevel\Core8051\ALUunit\BYTEop':

- Bit 4 operation.

\[\text{Bito := inl at: 4.}\]
\[\text{out1 := (inl from: 5 to: 7), Biti, (inl from: 0 to: 3).}\]

Text for function 'Bit5' of 'TopLevel\TopLevel\Core8051\ALUunit\BYTEop':

- Bit 5 operation.

\[\text{Bito := inl at: 5.}\]
\[\text{out1 := (inl from: 6 to: 7), Biti, (inl from: 0 to: 4).}\]

Text for function 'Bit6' of 'TopLevel\TopLevel\Core8051\ALUunit\BYTEop':

- Bit 6 operation.

\[\text{Bito := inl at: 6.}\]
\[\text{out1 := (inl from: 7 to: 7), Biti, (inl from: 0 to: 5).}\]

Text for function 'Bit7' of 'TopLevel\TopLevel\Core8051\ALUunit\BYTEop':

- Bit 7 operation.

\[\text{Bito := inl at: 7.}\]
\[\text{out1 := Biti, (inl from: 0 to: 7).}\]

'TopLevel\TopLevel\Core8051\ALUunit\DV_CTRL' is an operator.

This operator has 5 functions and is controlled by an unnamed control input.
The default function is 'NOP'.
Control connector (10 bits) without a name

**Control specification:**

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%0100000000</td>
<td>&quot;NOP&quot;</td>
</tr>
<tr>
<td>%0101000000</td>
<td>&quot;JC rel&quot;</td>
</tr>
<tr>
<td>%0101010000</td>
<td>&quot;JNC rel&quot;</td>
</tr>
<tr>
<td>%0101100000</td>
<td>&quot;JZ rel&quot;</td>
</tr>
<tr>
<td>%0101110000</td>
<td>&quot;JNZ rel&quot;</td>
</tr>
<tr>
<td>%0110000000</td>
<td>&quot;SJMP rel&quot;</td>
</tr>
<tr>
<td>%0110010000</td>
<td>&quot;HOV DPTR third byte&quot;</td>
</tr>
<tr>
<td>%1010010000</td>
<td>&quot;HOV DPTR second byte&quot;</td>
</tr>
<tr>
<td>%1110010000</td>
<td>&quot;HOV DPTR first byte&quot;</td>
</tr>
<tr>
<td>%1111000000</td>
<td>&quot;PUSH dir second cycle&quot;</td>
</tr>
<tr>
<td>%1111010000</td>
<td>&quot;PUSH dir third cycle&quot;</td>
</tr>
<tr>
<td>%1011100000</td>
<td>&quot;MOVX @DPTR second cycle&quot;</td>
</tr>
</tbody>
</table>

- 55 -

Enhanced 8051 Core
%010110111x, ·CJNE \@Ri,*data.dir·
%011100011x, ·XCH A,@Ri·
%011101011x, ·XCHD A,@Ri·
%011110011x, ·MOV A,@Ri"
%0100001xxx, "INC Rn"
%0100011xxx, "DEC Rn"
%010101011x, "ADD A,Rn"
%0101011xxx, "ADDC A,Rn"
%010101011x, "O RL A,Rn"
%0101011xxx, "ANL A,Rn"
%0110111xxx, "SUBB A,Rn"
%0111001xxx, "XCH A,Rn"
%0111011xxx, "DJNZ Rn,rel"
%0111101xxx RAMread, "MOV A,Rn"
%010111011x, "MOV \@Ri,\text{Mdata}"
%101010011x, "MOV \@Ri,dir"
%011111011x, "MOV \@Ri,A"
%0101111xxx, "MOV Rn,\text{Mdata}"
%1010101xxx, "MOV Rn,dir"
%0111111xxx RAMwrite, "MOV Rn,A"

Text for function 'Direct' of 'TopLevel\TopLevel\Core8051\ALUunit\DV_CTRL':
----------------------v----------------------
"Direct data valid."
CTRL := DIRDV.
----------------------v----------------------

Text for function 'NOP' of 'TopLevel\TopLevel\Core8051\ALUunit\DV_CTRL':
----------------------v----------------------
"Data not valid."
CTRL := 1 zeroes.
----------------------v----------------------

Text for function 'Normal' of 'TopLevel\TopLevel\Core8051\ALUunit\DV_CTRL':
----------------------v----------------------
"Data valid."
CTRL := 1 ones.
----------------------v----------------------

Text for function 'RAMread' of 'TopLevel\TopLevel\Core8051\ALUunit\DV_CTRL':
----------------------v----------------------
"RAM read data valid."
CTRL := RAMDV at: 0.
----------------------v----------------------

Text for function 'RAMwrite' of 'TopLevel\TopLevel\Core8051\ALUunit\DV_CTRL':
----------------------v----------------------
"RAM write data valid."
CTRL := RAMDV at: 1.
----------------------v----------------------

'TopLevel\TopLevel\Core8051\ALUunit\JtstMUX' is an operator.

This operator has 2 functions and is controlled by an unnamed control input.
The default function is 'ALU'.

Control connector (10 bits) without a name
Control specification:
----------------------v----------------------
"Select boolean operation."
%0100010000, "JBC"
%0100100000, "JB"
%0100110000, "JNB"
%0101000000, "JC"
%0101010000 Bool. "JNC"
----------------------v----------------------

Input connector ( 1 bit ) with name 'ALU'
Input connector ( 1 bit ) with name 'BOOL'
Output connector ( 1 bit ) with name 'Jtst'

Text for function 'ALU' of 'TopLevel\TopLevel\Core8051\ALUunit\JtstMUX':
----------------------v----------------------
"ALU operation."
Jtst := ALU.
Text for function 'Bool' of 'TopLevel\TopLevel\Core8051\ALUunit\JtstMUX':

"Boolean operation."

Jtst := BOOL.

'TopLevel\TopLevel\Core8051\ALUunit\outlMUX' is an operator.

This operator has 2 functions and is controlled by an unnamed control input. The default function is 'ALU'.

Control connector (10 bits) without a name

Control specification:

'Select boolean operation.'

%0100010000, "JBC bit,rel"
%0110010010, "MOV bit,C"
%0110110010, "CLR bit"
%0111000010, "CLR bit"
%0111010010 Bool. "SETB bit"

Input connector (8 bits) with name 'ALU'

Output connector (8 bits) with name 'out1'

Text for function 'ALU' of 'TopLevel\TopLevel\Core8051\ALUunit\outlMUX':

"ALU operation."

out1 := ALU.

'TopLevel\TopLevel\Core8051\ALUunit\PSWoMUX' is an operator.

This operator has 2 functions and is controlled by an unnamed control input. The default function is 'ALU'.

Control connector (10 bits) without a name

Control specification:

'Select boolean operation.'

%0110100000, "ORL C,bit"
%0110110000, "ANL C,bit"
%0101110010, "ORL C,bit"
%0110000010, "ANL C,bit"
%0110100010, "MOV C. bit"
%0110110011, "CPL C"
%0111000011, "CLR C"
%0111010011 Bool. "SETB C"

Input connector (8 bits) with name 'ALU'

Input connector (8 bits) with name 'BOOL'

Output connector (8 bits) with name 'PSWo'

Text for function 'ALU' of 'TopLevel\TopLevel\Core8051\ALUunit\PSWoMUX':

"ALU operation."

PSWo := ALU.

'TopLevel\TopLevel\Core8051\ARBITER' is a state machine controller.

This state machine controller has 8 states. No stack is available for 'subroutine' calls. This controller is enabled following system reset.
Text for state number 1 (reset state) of 'TopLevel\TopLevel\Core8051\ARBITER':

WAIT: "Wait until multiplication or division is completed."

[ALUUnit\ALUs\CNTR]
$I0 IR_SEL\IRop First; *Start up*
$I1..$6 ALUUnit\ALUs\CNTR inc;
ALUUnit\ALUs\TMP load;;
$I7 ALUUnit\ALUs\CNTR inc;
ALUUnit\ALUs\TMP reset;
IR_SEL\IRop Second;
;
------------------------------------------------------------------------

Text for state number 2 of 'TopLevel\TopLevel\Core8051\ARBITER':

STAGE1: "Decode instruction register or load next instruction code."

[PCUnit]
$I1 PCUnit\PCbufop INTR;
SP_CTRL\STACK PUSH;->STACK1

[BUSIFACE\RDY 11 "Wait until instruction code is loaded."

[BUSIFACE\Data_in\IR]
$I73 PCUnit\PCgen MOVCF_Dp;
$I53..$20,$25,$30,$35,$42,$43,$45,
$I2,$55,$92,$42,$63,$65,$72,$75,$82,$85..$8F,
$I9,$95,$A0,$A2,$A6..$AF,$B0,$B2,$B5,
$C0,$C2,$C5,$D0,$D2,$D5,$E5,$F5
BUSIFACE\RSop AR: "Load direct addressing register AR"
PCUnit\WR Read;
PCUnit\PCop INC;

$I71..$21,$24,$25,$34,$44,$54,$64,$74,
$81,$84,$86..$8F
BUSIFACE\RSop IM: "Load immediate data register IM"
PCUnit\WR Read;
PCUnit\PCop INC;

$I80..$60,$70,$80,$86..$DF
BUSIFACE\RSop REL: "Load relative jump address register REL"
PCUnit\WR Read;
PCUnit\PCop INC;

$I82..$32 SP_CTRL\STACK POP;
BUSIFACE\RSop IM;
RAMUnit\Busop RET;
SP_CTRL\SP_reg from: 5 to: 15
$50 PCUnit\PCbufop PCERam;
 )->STACK1

$I80 BUSIFACE\RSop IM;
PCUnit\PCadgen DATA_DP;
PCUnit\WR Read;>

$I82..$33 IR_SEL\IRop First;
BUSIFACE\RSop IM;
PCUnit\PCadgen DATA_RAM;
PCUnit\WR Read;>

$I83 PCUnit\PCgen MOVCF_PC;
PCUnit\PC hold;
IR_SEL\IRop First;
BUSIFACE\RSop IM;
PCUnit\WR Read;>

$I93 PCUnit\PCgen MOVCF_DP;
PCUnit\PC hold;
IR_SEL\IRop First;
BUSIFACE\RSop IM;
PCUnit\WR Read;>

$I70 IR_SEL\IRop First;
PCUnit\PCadgen DATA_DP;
PCUnit\WR Read;>

$I72..$F3 IR_SEL\IRop First;
PCUnit\PCadgen DATA_RAM;
PCUnit\WR Read;>

$I84..$A4 IR_SEL\IRop First;
ALUUnit\ALUs\TMP load;
ALUUnit\ALUs\CNTR inc;->WAIT

];
IR_SEL\IRop First;
);
------------------------------------------------------------------------

Text for state number 3 of 'TopLevel\TopLevel\Core8051\ARBITER':

STAGE2: "Decode instruction register or load next instruction code."

[BUSIFACE\RDY 11 "Wait until instruction code is loaded."
[BUSIFACE\Data_in\IR
|$10,$20,$30,$40,$53,$63,$75
BUSIFACE\Rsop IM: Load immediate data register IM
PCunit\WR Read;
Pcunit\PcCop INC;>
|$43,$53,$63,$75
BUSIFACE\Rsop IM: Load immediate data register IM
PCunit\WR Read;
Pcunit\PcCop INC;>
|$2,$12 PCunit\Pcbufop FChim;
BUSIFACE\Rsop IM: PCunit\WR Read;
Pcunit\PcCop INC;>
|$81,$91 PCunit\Pcbufop FChim;
BUSIFACE\Rsop IM: PCunit\WR Read;
Pcunit\PcCop INC;>
|$90 IR_Sel\IRop First;
BUSIFACE\Rsop IM: PCunit\WR Read;
Pcunit\PcCop INC;>
|$80 SP_CTRL\STACK POP;
BUSIFACE\Rsop IM: SP_CTRL\STACK POP;
IR_Sel\IRop First;
PCunit\WR Read;
Pcunit\PcCop INC;>
|$88..S93..SFO..SFA..SAF IR_Sel\IRop Second;->STAGE1
|$83..S93..SFO..SFA..SAF IR_Sel\IRop Second;->STAGE1
|%0xx0001 PCunit\Pcbufop FClm;
IR_Sel\IRop First;
PCunit\Pcgen ABS;->STAGE1
|%0xx1001 PCunit\Pcbufop FClm;
SP_CTRL\STACK PUSH;->STAGE1
];
IR_Sel\IRop First;->STAGE1
]

Text for state number 4 of 'TopLevel\TopLevel\Core8051\ARBITER':
------------------------------------------------------------------------------------------------------------------
STAGE3: Decode instruction register or load next instruction code.

[BUSIFACE\Data_in\IR
1$86..S93..SFO..SFA..SAF IR_Sel\IRop Second;->STAGE1
1$53..SFO..SFA..SAF IR_Sel\IRop Second;->STAGE1
1$53..SFO..SFA..SAF IR_Sel\IRop Second;->STAGE1
|5D0 SP_CTRL\STACK PUSH;->STAGE1
];

[BUSIFACE\RDY
10 [BUSIFACE\Data_in\IR
1$85 IR_Sel\IRop NOP;
];
11 [BUSIFACE\Data_in\IR Wait until instruction code is loaded.
|$10,$20,$30,$40,$53,$63,$75,$84..S93..SFO..SFA..SAF IR_Sel\IRop First;
|$85 IR_Sel\IRop Second;
|$2 PCunit\Pcgen LONG;
IR_Sel\IRop First;
Pcunit\Pcbufop FClm;
|590 IR_Sel\IRop Second;
BUSIFACE\Rsop IM: Load immediate data register IM
PCunit\WR Read;
Pcunit\PcCop INC;>
|$81,$91 PCunit\Pcbufop FChim;
BUSIFACE\Rsop IM: Load immediate data register IM
PCunit\WR Read;
Pcunit\PcCop INC;>
|$12 PCunit\Pcbufop FChim;

- 59 -
Text for state number 5 of 'TopLevel\TopLevel\Core80S1\ARB1TER':

```
| BUSIFACE\Data.in\IR |
| $86, $87, $A6, $A7 IR Sel\IRop Second;->STAGE1 |
| $CO [SP_CTRL\SV |
| $I IR Sel\IRop Third;->STAGE1 |
| ];->STAGE1 |
```

Text for state number 6 of 'TopLevel\TopLevel\Core80S1\ARB1TER':

```
| BUSIFACE\RDY |
| $90 IR Sel\IRop Second;->STAGE1 |
| $91 PCUnit\PCbufop PCLim; |
| SP_CTRL\STACK PUSH; » |
| ];->STAGE1 |
```

Text for state number 7 of 'TopLevel\TopLevel\Core80S1\ARB1TER':

```
| BUSIFACE\Data.in\IR |
| $22, $32 |
| SP_CTRL\POReg |
| $I SP_CTRL\STACK POP; |
| $2 RAMUnit\BusyoP RET; |
| SP_CTRL\STACK POP; |
| PCUnit\PCbufop CH; |
| $I3 PCUnit\PCbufop PCE; |
| $I4 SP_CTRL\STACK POP; |
| PCUnit\PCbufop PCE; |
| $I5 SP_CTRL\STACK POP; |
| PCUnit\PCbufop PCE; |
| BUSIFACE\RSop IM; |
| ];->STAGE1 |
```

- 60 -
Text for state number 8 of 'TopLevel\TopLevel\Core8051\ARBITER':

STACK3:

{SP_CTRL\SV S1
{PCUnit\I

10 [BUSIFACE\Data_in\IR
12 Ir Sel\IRop First;
13 PUnit\Iackop RETI;
BUSIFACE\RSop IR;
PUnit\WR Read;
PUnit\PCop INC;
12,13

{SP_CTRL\POPreg
12 SP_CTRL\STACK POP;

13 [RAMUnit\WR
10 SP_CTRL\STACK POP;
]
PUnit\PCbufop PCLram;
PUnit\PCgen GLOBAL;
10
11,31,51,71 PUnit\PCgen AES;
Ir Sel\IRop First;
12 PUnit\PCgen LONG;
Ir Sel\IRop First;
10
11 PUnit\PCgen GLOBAL;
PUnit\I reset;
BUSIFACE\RSop IR;
PUnit\WR Read;
PUnit\PCop INC;
] ->STAGE1
}

=======================================================================

'TopLevel\TopLevel\Core8051\BUSIFACE' is a schematic.

Bidirectional connector (8 bits) with name 'A'
Bidirectional connector (25 bits) with name 'Addrbus'
Bidirectional connector (8 bits) with name 'ALUout1'
Bidirectional connector (8 bits) with name 'ARC'
Bidirectional connector (8 bits) with name 'DATibus'
Bidirectional connector (1 bit) with name 'enable'
Bidirectional connector (8 bits) with name 'IMo'
Bidirectional connector (10 bits) with name 'IR'
Bidirectional connector (3 bits) with name 'IRlr'
Bidirectional connector (8 bits) with name 'IRo'
Bidirectional connector (25 bits) with name 'PCaddr'
Bidirectional connector (8 bits) with name 'PCdat'
Bidirectional connector (2 bits) with name 'POWR'
Bidirectional connector (1 bit) with name 'Ready'
Bidirectional connector (8 bits) with name 'RELO'
Bidirectional connector (1 bit) with name 'RS'
Bidirectional connector (16 bits) with name 'SPaddr'
Bidirectional connector (2 bits) with name 'SPWR'
Bidirectional connector (2 bits) with name 'WR'

=======================================================================

'TopLevel\TopLevel\Core8051\BUSIFACE\ADDR_op' is an operator.

This operator has 2 functions and is controlled by an unnamed control input.
The default function is 'PC'.

Control connector (1 bit) without a name

Control specification:
*Select external address.*

%1 SP. "Stack operation."

---

Output connector (25 bits) with name 'ADDR'
Input connector (25 bits) with name 'PC'
Input connector (16 bits) with name 'SP'

---

Text for function 'PC' of 'TopLevel\TopLevel\Core8051\BUSIFACE\ADDR_op':

"Program counter address."
ADDR := PC.

---

Text for function 'SP' of 'TopLevel\TopLevel\Core8051\BUSIFACE\ADDR_op':

"Stack address."
ADDR := 1 ones, 8 zeroes, SP.

---

'TopLevel\TopLevel\Core8051\BUSIFACE\Addr_out' is a schematic.

Bidirectional connector (25 bits) with name 'ADDRi'
Bidirectional connector (25 bits) with name 'ADDRo'
Bidirectional connector (2 bits) with name 'WRi'
Bidirectional connector (2 bits) with name 'WRo'

---

'TopLevel\TopLevel\Core8051\BUSIFACE\Addr_out\AddrReg' is a register.

This register is 25 bits wide and is controlled by control input 'WRo'.
The default function is 'hold'.
This register is loaded with value 0 following system reset.

The value loaded for the 'reset' command is 0.

Control specification:

---

%01 load. "Continue reading."
%10 load. "Continue writing."

---

'TopLevel\TopLevel\Core8051\BUSIFACE\Addr_out\AddrSel' is an operator.

This operator has 2 functions and is controlled by control input 'WRi'.
The default function is 'OldAddr'.

---

Input connector (25 bits) with name 'ADDRi'
Input connector (25 bits) with name 'in'
Output connector (25 bits) with name 'out'
Control connector (2 bits) with name 'WRi'

Control specification:

---

%00 OldAddr. "Idle."
%01 NewAddr. "Start read cycle."
%10 NewAddr. "Start write cycle."

---

Text for function 'NewAddr' of 'TopLevel\TopLevel\Core8051\BUSIFACE\Addr_out\AddrSel':

"New read or write cycle."
out := ADDRi.

---

Text for function 'OldAddr' of 'TopLevel\TopLevel\Core8051\BUSIFACE\Addr_out\AddrSel':

"Continue read or write cycle."
out := in.

---

'TopLevel\TopLevel\Core8051\BUSIFACE\Data_in' is a schematic.

Bidirectional connector (8 bits) with name 'ARi'
Bidirectional connector (8 bits) with name 'DATi'
Bidirectional connector (8 bits) with name 'IMo'
Bidirectional connector (8 bits) with name 'IRO'
Bidirectional connector (1 bit) with name 'Ready'
Bidirectional connector (8 bits) with name 'RELo'
Bidirectional connector (3 bits) with name 'RSi'
Bidirectional connector (2 bits) with name 'WRo'
'TopLevel\TopLevel\Core8051\BUSIFACE\Data_in\AR' is a register.

This register is 8 bits wide and is controlled by an unnamed control input. The default function is 'hold'. This register is loaded with value 0 following system reset.

The value loaded for the 'reset' command is 0.

Control specification:

- Load new direct address.
  %010 load.

'TopLevel\TopLevel\Core8051\BUSIFACE\Data_in\IM' is a register.

This register is 8 bits wide and is controlled by an unnamed control input. The default function is 'hold'. This register is loaded with value 0 following system reset.

The value loaded for the 'reset' command is 0.

Control specification:

- Load new immediate data.
  %011 load.

'TopLevel\TopLevel\Core8051\BUSIFACE\Data_in\IR' is a register.

This register is 8 bits wide and is controlled by an unnamed control input. The default function is 'hold'. This register is loaded with value 0 following system reset.

The value loaded for the 'reset' command is 0.

Control specification:

- Load new instruction code.
  %001 load.

'TopLevel\TopLevel\Core8051\BUSIFACE\Data_in\RegSel' is an operator.

This operator has 2 functions and is controlled by control input 'RDYi'. The default function is 'Idle'.

Output connector (3 bits) with name 'out'
Control connector (1 bit) with name 'RDYi'

Control specification:

- Idle.
- RegLoad.

Input connector (3 bits) with name 'RSi'
Input connector (2 bits) with name 'WRi'

Text for function 'Idle' of 'TopLevel\TopLevel\Core8051\BUSIFACE\Data_in\RegSel':

out := 3 zeroes.

Text for function 'RegLoad' of 'TopLevel\TopLevel\Core8051\BUSIFACE\Data_in\RegSel':

- Select register to load if reading is ready.
  out := (WRi at: 0) if0: 3 zeroes
  if1: RSi.

'TopLevel\TopLevel\Core8051\BUSIFACE\Data_in\REL' is a register.

This register is 8 bits wide and is controlled by an unnamed control input. The default function is 'hold'. This register is loaded with value 0 following system reset.

The value loaded for the 'reset' command is 0.

Control specification:

- Load new relative jump address.
Bidirectional connector (8 bits) with name 'A'
Bidirectional connector (8 bits) with name 'ALUout1'
Bidirectional connector (8 bits) with name 'DATo'
Bidirectional connector (10 bits) with name 'IR'
Bidirectional connector (5 bits) with name 'PCdat'
Bidirectional connector (2 bits) with name 'WRi'

'Toplevel\TopLevel\Core8051\BUSIFACE\Data_out\DataBuf' is a register.

This register is 8 bits wide.
The default function is 'load'.
This register is loaded with value 0 following system reset.

'Toplevel\TopLevel\Core8051\BUSIFACE\Data_out\DataSel' is an operator.

This operator has 2 functions and is controlled by control input 'WRi'.
The default function is 'OldData'.

%1011000000 PUSH, 'PUSH direct.'
%0111100010x MOVX @Ri,A, 'MOVX @Ri,A'
%0111100010x MOVX @DPTR,A 'MOVX @DPTR,A'

Input connector (8 bits) with name 'PCdat'

Text for function 'MOVX' of 'TopLevel\TopLevel\Core8051\BUSIFACE\Data_out\DATo_Sel':
----------------------v----------------------
'MOVX @DPTR or @Ri,A.'
DATA := A.

Text for function 'PUSH' of 'TopLevel\TopLevel\Core8051\BUSIFACE\Data_out\DATo_Sel':
----------------------v----------------------
'PUSH program counter onto stack.'
DATA := PCdat.

Text for function 'PCstacK' of 'TopLevel\TopLevel\Core8051\BUSIFACE\Data_out\DATo_Sel':
----------------------v----------------------
'PUSH program counter onto stack.'
DATA := A.

Control specification:

Input connector (8 bits) with name 'A'
Input connector (8 bits) with name 'ALUout1'
Output connector (8 bits) with name 'DATo'
Control connector (10 bits) with name 'IR'

Text for function 'MOVX' of 'TopLevel\TopLevel\Core8051\BUSIFACE\Data_out\DATo_Sel':
----------------------v----------------------
'MOVX @DPTR or @Ri,A.'
DATA := A.

Text for function 'PUSH' of 'TopLevel\TopLevel\Core8051\BUSIFACE\Data_out\DATo_Sel':
----------------------v----------------------
'PUSH program counter onto stack.'
DATA := PCdat.
"PUSH dir."
DATA := ALUout1.

'TopLevel\TopLevel\Core8051\BUSIFACE\RDY' is a register.

This register is 1 bit wide.
The default function is 'load'.
This register is loaded with value 0 following system reset.

The value loaded for the 'reset' command is 0.

'TopLevel\TopLevel\Core8051\BUSIFACE\RSbuf' is a register.

This register is 3 bits wide and is controlled by an unnamed control input.
The default function is 'load'.
This register is loaded with value 0 following system reset.

The value loaded for the 'reset' command is 0.

Control specification:

This operator has 5 functions.
The default function is 'Normal'.

Input connector (3 bits) with name 'in'
Input connector (1 bit) with name 'RS'
Output connector (3 bits) with name 'RSo'

Text for function 'AR' of 'TopLevel\TopLevel\Core8051\BUSIFACE\RSop':

-Load direct addressing register-
RSo := 1 zeroes, 1 ones, 1 zeroes.

Text for function 'IM' of 'TopLevel\TopLevel\Core8051\BUSIFACE\RSop':

-Load immediate data register-
RSo := 1 zeroes, 2 ones.

Text for function 'IR' of 'TopLevel\TopLevel\Core8051\BUSIFACE\RSop':

-Load instruction code register-
RSo := 2 zeroes, 1 ones.

Text for function 'Normal' of 'TopLevel\TopLevel\Core8051\BUSIFACE\RSop':

-Continue data input-
RSo := RS if0: in
if1: (2 zeroes, 1 ones).

Text for function 'Rel' of 'TopLevel\TopLevel\Core8051\BUSIFACE\RSop':

-Load relative jump address register-
RSo := 1 ones, 2 zeroes.

'TopLevel\TopLevel\Core8051\BUSIFACE\SPwrop' is an operator.

This operator has 2 functions and is controlled by an unnamed control input.
The default function is 'SP'.

Control connector (16 bits) without a name

Control specification:

-Detect internal stack acces-
(8..15)
$0\ PC.

Output connector (1 bit) with name 'SPc'
Input connector (2 bits) with name 'SPWR'

Text for function 'PC' of 'TopLevel\TopLevel\Core8051\BUSIFACE\SPwrop':

*No external stack acces.*

SPc := 0 zeroes.

Text for function 'SP' of 'TopLevel\TopLevel\Core8051\BUSIFACE\SPwrop':

*External stack acces.*

SPc := (SPWR at: 0) \ (SPWR at: 1).

Bidirectional connector (1 bit) with name 'enable'

Bidirectional connector (1 bit) with name 'RDY1'

Bidirectional connector (2 bits) with name 'WRi'

Bidirectional connector (2 bits) with name 'WRo'

This operator has 2 functions and is controlled by an unnamed control input.
The default function is 'Continue'.

Control connector (2 bits) without a name

Control specification:

%00 Continue. 'Continue cycle.'
%01 NewWR. 'Start read cycle.'
%10 NewWR. 'Start write cycle.'

Text for function 'Continue' of 'TopLevel\TopLevel\Core8051\BUSIFACE\WR_Ctrl\WRop':

*Continue previous cycle.*

WRo := in.
enable := in at: 1.

Text for function 'NewWR' of 'TopLevel\TopLevel\Core8051\BUSIFACE\WR_Ctrl\WRop':

*Start read or write cycle.*

%0 load. 'Continue.'
%1 reset. 'End of read-write cycle.'

This register is 2 bits wide and is controlled by an unnamed control input.
The default function is 'hold'.

This register is loaded with value 0 following system reset.
The value loaded for the 'reset' command is 0.

Control specification:

%0 load. 'Continue.'
%1 reset. 'End of read-write cycle.'

This operator has 2 functions and is controlled by an unnamed control input.
The default function is 'PC'.

Control connector (1 bit) without a name

Control specification:

%1 STACK.

Input connector (2 bits) with name 'PC'
Input connector (2 bits) with name 'SPWR'
Section of Digital Information Systems

Enhanced 8051 Core

Output connector (2 bits) with name 'WRo'

Text for function 'PC' of 'TopLevel\TopLevel\Core8051\BUSIFACE\WR_op':
"Program counter unit operation."

WRO := PC.

Text for function 'Stack' of 'TopLevel\TopLevel\Core8051\BUSIFACE\WR_op':
"Stack operation."

WRO := SPWR.

'Bidirectional connector (8 bits) with name 'Ai' is a schematic.

Bidirectional connector (8 bits) with name 'ALUout1'
Bidirectional connector (8 bits) with name 'AR'
Bidirectional connector (8 bits) with name 'B_i'
Bidirectional connector (8 bits) with name 'DirDat'
Bidirectional connector (1 bit) with name 'DIRDV'
Bidirectional connector (8 bits) with name 'DIRc'
Bidirectional connector (10 bits) with name 'IR'
Bidirectional connector (8 bits) with name 'FSW1'
Bidirectional connector (8 bits) with name 'RAMDATi'
Bidirectional connector (2 bits) with name 'RAMDV'
Bidirectional connector (10 bits) with name 'SFR'
Bidirectional connector (10 bits) with name 'SFRaddr'
Bidirectional connector (8 bits) with name 'SRF1'
Bidirectional connector (8 bits) with name 'SPI'

'TopLevel\TopLevel\Core8051\Direct\AR_SEL' is an operator.

This operator has 2 functions and is controlled by an unnamed control input. The default function is 'Direct'.

Control connector (10 bits) without a name

Control specification:

'Select boolean operations (Direct operation is default).'

%0100010000 BOOL. "JBC bit,rel"%0100100000 BOOL. "JB bit,rel"%0100110000 BOOL. "JNB bit,rel"%0110100000 BOOL. "ORL C,/bit"%0110110000 BOOL. "ANL C, /bit"%0101110010 BOOL. "ORL C .bit'"%0110000010 BOOL. "ANL C .bit'"%0110010010 BOOL. "MOV bit,C'"%0110100010 BOOL. "MOV C.bit"%0110110010 BOOL. "CFL bit"%0111000010 BOOL. "CLR bit"%0111010010 BOOL. "SETB bit"

Input connector (8 bits) with name 'AR'
Input connector (2 bits) with name 'SW'

Output connector (10 bits) with name 'SFR'

Text for function 'BOOL' of 'TopLevel\TopLevel\Core8051\Direct\AR_SEL':
"Boolean operation."

SFR := (AR at: 7) if0: RW, 2 zeroes, 1 ones, (AR from: 3 to: 7)
ifi: RW, (AR from: 3 to: 7), 3 zeroes.

Text for function 'Direct' of 'TopLevel\TopLevel\Core8051\Direct\AR_SEL':
"Direct addressing."

SFR := RW, AR.

'TopLevel\TopLevel\Core8051\Direct\Dir_Sel' is an operator.

This operator has 7 functions and is controlled by an unnamed control input. The default function is 'SFR'.

Control connector (10 bits) without a name

Control specification:
Select direct addressing mode.
(0..7)
%0xxxxxxx RAM.
%1000001x DP.
%10000100 DP.
%110100000 PSW.
%11100000 A.
%11110000 B.

---

Input connector (8 bits) with name 'Ai'.
Input connector (8 bits) with name 'Bin'.
Input connector (1 bit) with name 'DIRDV'.
Input connector (8 bits) with name 'DIRo'.
Input connector (8 bits) with name 'DPi'.
Input connector (2 bits) with name 'RAMi'.
Input connector (10 bits) with name 'SPI'.
Output connector (10 bits) with name 'SFRI'.
Input connector (8 bits) with name 'SFRi'.
Input connector (8 bits) with name 'SPI'.

Text for function 'A' of 'TopLevel\TopLevel\Core8051\Direct\Dir_Sel':

```
* SFR operation on register A.*
DIRO := Ai.
SFRaddr := 2 zeroes, (SFR from: 0 to: 7).
DIRDV := (SFR at: 9) \ (SFR at: 8).
```

Text for function 'B' of 'TopLevel\TopLevel\Core8051\Direct\Dir_Sel':

```
* SFR operation on register B.*
DIRO := Bin.
SFRaddr := 2 zeroes, (SFR from: 0 to: 7).
DIRDV := (SFR at: 9) \ (SFR at: 8).
```

Text for function 'DP' of 'TopLevel\TopLevel\Core8051\Direct\Dir_Sel':

```
* SFR operation on byte in register DP.*
DIRO := DPi.
SFRaddr := 2 zeroes, (SFR from: 0 to: 7).
DIRDV := (SFR at: 9) \ (SFR at: 8).
```

Text for function 'PSW' of 'TopLevel\TopLevel\Core8051\Direct\Dir_Sel':

```
* SFR operation on register PSW.*
DIRO := pswi.
SFRaddr := 2 zeroes, (SFR from: 0 to: 7).
DIRDV := (SFR at: 9) \ (SFR at: 8).
```

Text for function 'RAM' of 'TopLevel\TopLevel\Core8051\Direct\Dir_Sel':

```
* SFR operation on internal RAM byte.*
DIRO := RAMi.
SFRaddr := 2 zeroes, (SFR from: 0 to: 7).
DIRDV := ((SFR at: 8) \ (RAMDV at: 1)) \ ((SFR at: 9) \ (RAMDV at: 0)).
```

Text for function 'SFR' of 'TopLevel\TopLevel\Core8051\Direct\Dir_Sel':

```
* SFR operation on external SFR.*
DIRO := SFRI.
SFRaddr := SFR.
DIRDV := (SFR at: 9) \ (SFR at: 8).
```

Text for function 'SP' of 'TopLevel\TopLevel\Core8051\Direct\Dir_Sel':

```
* SFR operation on byte in register SP.*
DIRO := SPI.
SFRaddr := 2 zeroes, (SFR from: 0 to: 7).
DIRDV := (SFR at: 9) \ (SFR at: 8).
```

---

'STopLevel\TopLevel\Core8051\Direct\DIRbuffer' is a schematic.
Bidirectional connector (1 bit) with name 'DIRDV'
Bidirectional connector (8 bits) with name 'DIRI'
Bidirectional connector (8 bits) with name 'DIRO'
Bidirectional connector (8 bits) with name 'in'
Bidirectional connector (10 bits) with name 'IR'
Bidirectional connector (8 bits) with name 'out'
Bidirectional connector (8 bits) with name 'RAMdatI'
Bidirectional connector (2 bits) with name 'RAMDV'

'TopLevel\TopLevel\Core8051\Direct\DIRbuffer\DIRbuf' is a register.

This register is 8 bits wide and is controlled by an unnamed control input.
The default function is 'hold'.
This register is loaded with unknown values after a system reset.

The value loaded for the 'reset' command is 0.

Control specification:

Output buffer for instructions:
MOV direct, direct & MOV direct, Rn & MOV direct, Rn
%1 load.

'TopLevel\TopLevel\Core8051\Direct\DIRbuffer\DIRbufop' is an operator.

This operator has 3 functions and is controlled by control input 'IR'.
The default function is 'Normal'.

Input connector (8 bits) with name 'Buf'
Input connector (1 bit) with name 'DIRDV'
Input connector (8 bits) with name 'DIRI'
Output connector (8 bits) with name 'DIRO'
Input connector (8 bits) with name 'in'
Control connector (10 bits) with name 'IR'

Control specification:

Direct addressing output buffer.
%01110000 BufferIn. "PUSH dir first cycle"
%01110000 BufferOut. "PUSH dir second cycle"
%01000010 BufferIn. "MOV dir, dir first cycle"
%01000010 BufferOut. "MOV dir, dir second cycle"
%01101001 BufferIn. "MOV @R0, dir first cycle"
%01101001 BufferOut. "MOV @R0, dir second cycle"
%10101001 BufferIn. "MOV Rn, dir first cycle"
%10101001 BufferOut. "MOV Rn, dir second cycle"

Output connector (1 bit) with name 'load'
Output connector (8 bits) with name 'out'
Input connector (2 bits) with name 'RAMDV'
Input connector (8 bits) with name 'RAMi'

Text for function 'BufferIn' of 'TopLevel\TopLevel\Core8051\Direct\DIRbuffer\DIRbufop':

Buffer direct addressing output.
load := DIRDV.
DIRO := in.

Text for function 'BufferOut' of 'TopLevel\TopLevel\Core8051\Direct\DIRbuffer\DIRbufop':

Output buffer if not read for ASA RAM.
out := (RAMDV at: 0) if0: Buf
         if1: RAMi.
load := RAMDV at: 0.
DIRO := (RAMDV at: 0) if0: Buf
         if1: RAMi.

Text for function 'Normal' of 'TopLevel\TopLevel\Core8051\Direct\DIRbuffer\DIRbufop':

Normal direct addressing output.
load := 1 zeroes.
DIRO := in.
out := DIRI.
The default function is "NOP".

Control connector (10 bits) without a name

Control specification:

Select SFR mode.

<table>
<thead>
<tr>
<th>Binary Code</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0010010000</td>
<td>&quot;JB bit, rel&quot;</td>
</tr>
<tr>
<td>0010100000</td>
<td>&quot;JNB bit, rel&quot;</td>
</tr>
<tr>
<td>0011000000</td>
<td><em>PUSH dir</em></td>
</tr>
<tr>
<td>0011010000</td>
<td><em>ORL C, bit</em></td>
</tr>
<tr>
<td>0011100000</td>
<td><em>ANL C, bit</em></td>
</tr>
<tr>
<td>0011110000</td>
<td><em>ADD A, dir</em></td>
</tr>
<tr>
<td>0010000010</td>
<td>&quot;ADD A, dir&quot;</td>
</tr>
<tr>
<td>0010101010</td>
<td>&quot;XRL A, dir'</td>
</tr>
<tr>
<td>0011000010</td>
<td><em>PUSH dir</em></td>
</tr>
<tr>
<td>0101010101</td>
<td>&quot;ORL dir, A'</td>
</tr>
<tr>
<td>0110010101</td>
<td>&quot;ANL dir, A'</td>
</tr>
<tr>
<td>0111100101</td>
<td>&quot;MOV dir, dir (first cycle)'</td>
</tr>
<tr>
<td>0100101010</td>
<td>&quot;SUBB A, dir&quot;</td>
</tr>
<tr>
<td>0101110010</td>
<td><em>POP dir</em></td>
</tr>
<tr>
<td>0101010000</td>
<td><em>POP dir</em></td>
</tr>
<tr>
<td>1011000010</td>
<td>&quot;MOV dir, dir (second cycle)'</td>
</tr>
<tr>
<td>0111110101</td>
<td>&quot;MOV dir, dir, dir (second cycle)'</td>
</tr>
<tr>
<td>1100100101</td>
<td>&quot;MOV dir, dir (third cycle)'</td>
</tr>
<tr>
<td>0101010000</td>
<td>&quot;MOV dir, dir (third cycle)'</td>
</tr>
<tr>
<td>0101010101</td>
<td>&quot;MOV dir, dir (third cycle)'</td>
</tr>
<tr>
<td>1010000100</td>
<td>&quot;MOV dir, dir (third cycle)'</td>
</tr>
<tr>
<td>0111010001</td>
<td>&quot;MOV dir, dir (third cycle)'</td>
</tr>
<tr>
<td>0111010101</td>
<td>&quot;MOV dir, dir (third cycle)'</td>
</tr>
<tr>
<td>0111100010</td>
<td>&quot;MOV dir, dir (third cycle)'</td>
</tr>
<tr>
<td>0111110010</td>
<td>&quot;MOV dir, dir (third cycle)'</td>
</tr>
<tr>
<td>0010000010</td>
<td>&quot;ORL dir, A'</td>
</tr>
<tr>
<td>0011100010</td>
<td>&quot;ANL dir, A'</td>
</tr>
<tr>
<td>0010100000</td>
<td>&quot;XRL dir, A'</td>
</tr>
<tr>
<td>0011000010</td>
<td>&quot;ORL bit, C'</td>
</tr>
<tr>
<td>0100101010</td>
<td>&quot;CLR bit&quot;</td>
</tr>
<tr>
<td>0110110101</td>
<td>&quot;SETB bit&quot;</td>
</tr>
<tr>
<td>01010100011</td>
<td>&quot;ORL A, dir'</td>
</tr>
<tr>
<td>00101010011</td>
<td>&quot;ANL A, dir'</td>
</tr>
<tr>
<td>0010000010</td>
<td>&quot;INC dir&quot;</td>
</tr>
<tr>
<td>0100100101</td>
<td>&quot;DEC dir&quot;</td>
</tr>
<tr>
<td>0111010001</td>
<td>&quot;XCH A, dir'</td>
</tr>
<tr>
<td>0111100101</td>
<td>&quot;DJNZ dir, rel'</td>
</tr>
</tbody>
</table>

Output connector (2 bits) with name 'RW'

Text for function 'Modify' of 'TopLevel\TopLevel\Core8051\Direct\RWop':

* Modify SFR. *
RW := 2 ones.

Text for function 'NOP' of 'TopLevel\TopLevel\Core8051\Direct\RWop':

* No SFR access. *
RW := 2 zeroes.

Text for function 'Read' of 'TopLevel\TopLevel\Core8051\Direct\RWop':

* Read SFR. *
RW := 1 ones, 1 zeroes.

Text for function 'Wrt' of 'TopLevel\TopLevel\Core8051\Direct\RWop':

* Write SFR. *
RW := 1 zeroes, 1 ones.

"TopLevel\TopLevel\Core8051\DP_reg" is a schematic.

Bidirectional connector (8 bits) with name 'DirDat'
Bidirectional connector (24 bits) with name 'DP'
Bidirectional connector (8 bits) with name 'IM'
Bidirectional connector (10 bits) with name ‘IR’
Bidirectional connector (10 bits) with name ‘SFR’

'TopLevel\TopLevel\Core8051\DP_reg\DP_CTRL' is an operator.

This operator has 4 functions and is controlled by an unnamed control input.
The default function is ‘SFR’.

Control connector (10 bits) without a name

Control specification:

Control specification:

Control specification:

Control specification:

Control specification:

Control specification:

'TopLevel\TopLevel\Core8051\DP_reg\DP_CTRL' is an operator.

This operator has 3 functions and is controlled by an unnamed control input.
The default function is ‘DPL’.

Control connector (10 bits) without a name
DP240 := (DP24i from: 16 to: 23), DP1, (DP24i from: 0 to: 7).

Text for function 'DPL' of 'TopLevel\TopLevel\Core8051\DP_reg\DP_Sel':
---------------------------------------------
"Select first byte from data pointer."
DP0 := DP24i from: 0 to: 7.
DP240 := (DP24i from: 8 to: 23), DP1.
---------------------------------------------

This operator has 2 functions and is controlled by an unnamed control input. The default function is 'OLD_DP'.

Control connector (10 bits) without a name

Control specification:
----------------------v----------------------
"Control of new data pointer is written."
%1100000000 NEW_DP.
%1100000000 NEW_DP.
----------------------v----------------------
Input connector (24 bits) with name 'DP'
Input connector (24 bits) with name 'in'
Output connector (24 bits) with name 'out'

Text for function 'NEW_DP' of 'TopLevel\TopLevel\Core8051\DP_reg\DP0p':
---------------------------------------------
"Load data pointer with new value."
out := DP.
---------------------------------------------

Text for function 'OLD_DP' of 'TopLevel\TopLevel\Core8051\DP_reg\DP0p':
---------------------------------------------
"Load data pointer with same value."
out := in.
---------------------------------------------

This register is 24 bits wide and is controlled by an unnamed control input. The default function is 'load'. This register is loaded with value 0 following system reset.

The value loaded for the 'reset' command is 0.

Control specification:
----------------------v----------------------
"Load data pointer with same or new value unless increase data pointer instruction."
%0110000001 inc.
---------------------------------------------

This operator has 1 function.
The default function is 'Clear_IR'.

Input connector (2 bits) with name 'lack'
Input connector (3 bits) with name 'IRclr'
Bidirectional connector (8 bits) with name 'IR1'
Bidirectional connector (10 bits) with name 'IR0'

Text for function 'Clear_IR' of 'TopLevel\TopLevel\Core8051\IR_Sel\Clr_op':
---------------------------------------------
"Reset instruction code."
out := IR \ (l zeroes, lack).
---------------------------------------------

This register is 2 bits wide and is controlled by an unnamed control input. The default function is 'load'. This register is loaded with value 0 following system reset.
The value loaded for the 'reset' command is 0.

Control specification:

---
"Clear register while loading new instruction code."
%001 reset.
---

'TopLevel\TopLevel\Core8051\IR_Sel\IRop' is an operator.

This operator has 5 functions.
The default function is 'Normal'.

- **Input connector (2 bits) with name 'in':**
- **Input connector (8 bits) with name 'IRi':**
- **Output connector (10 bits) with name 'IRO':**

Text for function 'First' of 'TopLevel\TopLevel\Core8051\IR_Sel\IRop':

---
"First instruction cycle."
IRO := 1 zeroes, 1 ones, IRi.
out := 1 zeroes, 1 ones.
---

Text for function 'NOP' of 'TopLevel\TopLevel\Core8051\IR_Sel\IRop':

---
"No instruction cycle."
IRO := 2 zeroes, IRi.
out := 2 zeroes.
---

Text for function 'Normal' of 'TopLevel\TopLevel\Core8051\IR_Sel\IRop':

---
"Continue instruction code."
IRO := in, IRi.
out := in.
---

Text for function 'Second' of 'TopLevel\TopLevel\Core8051\IR_Sel\IRop':

---
"Second instruction cycle."
IRO := 1 ones, 1 zeroes, IRi.
out := 1 zeroes, 1 zeroes.
---

Text for function 'Third' of 'TopLevel\TopLevel\Core8051\IR_Sel\IRop':

---
"Third instruction cycle."
IRO := 2 ones, IRi.
out := 2 ones.
---

Bidirectional connector (25 bits) with name 'ADDRo'
Bidirectional connector (8 bits) with name 'Ai'
Bidirectional connector (1 bit) with name 'CTRL'
Bidirectional connector (8 bits) with name 'Data'
Bidirectional connector (24 bits) with name 'DP'
Bidirectional connector (2 bits) with name 'JstS'
Bidirectional connector (8 bits) with name 'IR'
Bidirectional connector (10 bits) with name 'IREq'
Bidirectional connector (3 bits) with name 'IREq'
Bidirectional connector (1 bit) with name 'JstS'
Bidirectional connector (8 bits) with name 'RAMDATi'
Bidirectional connector (8 bits) with name 'Rel'
Bidirectional connector (1 bit) with name 'RS'
Bidirectional connector (2 bits) with name 'WRO'
---

'TopLevel\TopLevel\Core8051\PCUnit\CTRLop' is an operator.

This operator has 3 functions and is controlled by an unnamed control input.
The default function is 'Normal'.

Control connector (10 bits) without a name

Control specification:

---
"Select last cycle program counter control function."
%0100010000, "JBC bit.rel"
enhanced 8051 core

section of digital information systems

%0100100000, "JB bit.rel"
%0100110000, "JNB bit.rel"
%0101000000, "JC rel"
%0101010000, "JNC rel"
%0101100000, "JZ rel"
%0101110000, "JNZ rel"
%0110000000, "SJMP rel"
%01101101xx, "CJNE"
%0111000000, "PUSH dir"
%1011000000, "PUSH dir'
%0110010000, "MOY DPTR first byte"
%1010010000, "MOY DPTR second byte"
%0111000000, "FIRE dir"
%0110000101, "MOY dir,dir first cycle"
%011000011x, "MOY dir,@Ri first cycle"
%0111010101, "DJNZ dir, re1"
%0111011xxx Re1Jmp. "DJNZ Rn, re1"
%011010011x, "MOY @Ri,dir first cycle"
%0110101xxx NOP. "MOY Rn,dir first cycle"
%0110101xxx NOP. "MOV Rn,dir first cycle"

---------------------------------------------------------------
Input connector (1 bit) with name 'CTRLi'
Output connector (1 bit) with name 'CTRLo'
Input connector (1 bit) with name 'Jtsti'
Output connector (1 bit) with name 'Jtsto'

Text for function 'NOP' of 'TopLevel\TopLevel\Core8051\PCunit\CTRLop':
---------------------------------------------------------------
'Suppress last cycle signal.'
CTRLo := 1 zeroes.
Jtsto := 1 zeroes.

Text for function 'Normal' of 'TopLevel\TopLevel\Core8051\PCunit\CTRLop':
---------------------------------------------------------------
'Detect last cycle signal and don't jump relative.'
Jtsto := 1 zeroes.
CTRLo := CTRLi.

Text for function 'RelJmp' of 'TopLevel\TopLevel\Core8051\PCunit\CTRLop':
---------------------------------------------------------------
'Detect last cycle signal and jump relative conditionally.'
Jtsto := CTRLi if 0: 1 zeroes
if 1: Jtsti.
CTRLo := CTRLi.

'TopLevel\TopLevel\Core8051\PCunit\I' is a register.
This register is 1 bit wide.
The default function is 'load'.
This register is loaded with value 0 following system reset.
The value loaded for the 'reset' command is 0.
'TopLevel\TopLevel\Core8051\PCunit\Jackop' is an operator.

This operator has 2 functions.
The default function is 'Jack'.

Input connector (2 bits) with name 'Jacki'
Output connector (2 bits) with name 'Jacko'

Text for function 'Jack' of 'TopLevel\TopLevel\Core8051\PCunit\Jackop':
---------------------------------------------------------------
'Acknowledge requested interrupt.'
Jacko := Jacki.

Text for function 'RETI' of 'TopLevel\TopLevel\Core8051\PCunit\Jackop':
---------------------------------------------------------------
'End of interrupt.'
Jacko := 1 ones, 1 zeroes.

'TopLevel\TopLevel\Core8051\PCunit\Ino' is a register.
This register is 3 bits wide and is controlled by an unnamed control input.
The default function is 'load.'
This register is loaded with value 7 following system reset.

The value loaded for the 'reset' command is 0.

Control specification:

---

"Hold interrupt vector if interrupt request is accepted."

$01 hold. 'Interrupt handling'  

---

'TopLevel\TopLevel\Core8051\PCunit\INTR' is an operator.

This operator has 2 functions and is controlled by an unnamed control input.
The default function is 'INTR'.

Control connector (3 bits) without a name

Control specification:

---

'Detect interrupt request.'

---

'TopLevel\TopLevel\Core8051\PCunit\INTR' is an operator.

This operator has 2 functions and is controlled by an unnamed control input.
The default function is 'Normal'.

Control connector (2 bits) without a name

Control specification:

---

'Set register I for stack state machine.'

---

'TopLevel\TopLevel\Core8051\PCunit\Iop' is an operator.

This operator has 2 functions and is controlled by an unnamed control input.
The default function is 'Normal'.

Control connector (1 bit) without a name

Control specification:

---

'Detect relative jump.'

---

'TopLevel\TopLevel\Core8051\PCunit\JUMP' is an operator.

This operator has 2 functions and is controlled by an unnamed control input.
The default function is 'PC'.

Control connector (1 bit) without a name

Control specification:

---

"Hold interrupt vector if interrupt request is accepted."

$01 hold. 'Interrupt handling'  

---

'TopLevel\TopLevel\Core8051\PCunit\INTR' is an operator.

This operator has 2 functions and is controlled by an unnamed control input.
The default function is 'INTR'.

Control connector (3 bits) without a name

Control specification:

---

'Detect interrupt request.'

---

'TopLevel\TopLevel\Core8051\PCunit\INTR' is an operator.

This operator has 2 functions and is controlled by an unnamed control input.
The default function is 'Normal'.

Control connector (2 bits) without a name

Control specification:

---

'Set register I for stack state machine.'

---

'TopLevel\TopLevel\Core8051\PCunit\Iop' is an operator.

This operator has 2 functions and is controlled by an unnamed control input.
The default function is 'Normal'.

Control connector (1 bit) without a name

Control specification:

---

'Detect relative jump.'

---

'TopLevel\TopLevel\Core8051\PCunit\JUMP' is an operator.

This operator has 2 functions and is controlled by an unnamed control input.
The default function is 'PC'.

Control connector (1 bit) without a name

Control specification:
Input connector (24 bits) with name ‘PC’
Output connector (24 bits) with name ‘PC’
Input connector (24 bits) with name ‘REL’

Text for function ‘PC’ of ‘TopLevel\TopLevel\Core8051\PCunit\JUMP’:
"Normal throughput program counter."
PC := PC.

Text for function ‘REL’ of ‘TopLevel\TopLevel\Core8051\PCunit\JUMP’:
"Relative jump."
PC := REL.

This register is 24 bits wide.
The default function is ‘load’.
This register is loaded with value 0 following system reset.
The value loaded for the ‘reset’ command is 0.

‘TopLevel\TopLevel\Core8051\PCunit\PCadgen’ is an operator.
This operator has 3 functions.
The default function is ‘PROGRAM’.

Input connector (25 bits) with name ‘ADDR’
Input connector (24 bits) with name ‘DP’
Input connector (24 bits) with name ‘FC’
Input connector (8 bits) with name ‘RAM’

Text for function ‘DATA_DP’ of ‘TopLevel\TopLevel\Core8051\PCunit\PCadgen’:
"MOVX DPTR, ADDR"
ADDR := 1 ones, DP.

Text for function ‘DATA_RAM’ of ‘TopLevel\TopLevel\Core8051\PCunit\PCadgen’:
"External data memory."
ADDR := 1 ones, 16 zeroes, RAM.

Text for function ‘PROGRAM’ of ‘TopLevel\TopLevel\Core8051\PCunit\PCadgen’:
"External program memory."
ADDR := 1 zeroes, PCI.

This register is 24 bits wide.
The default function is ‘load’.
This register is loaded with value 0 following system reset.
The value loaded for the ‘reset’ command is 0.

‘TopLevel\TopLevel\Core8051\PCunit\PCbufop’ is an operator.
This operator has 8 functions.
The default function is ‘Normal’.

Input connector (8 bits) with name ‘IM’
Input connector (24 bits) with name ‘IN’
Input connector (3 bits) with name ‘INR’
Output connector (24 bits) with name ‘OUT’
Output connector (24 bits) with name ‘FC’
Input connector (8 bits) with name ‘RAM’

Text for function ‘INTR’ of ‘TopLevel\TopLevel\Core8051\PCunit\PCbufop’:
"Store interrupt address."
out := 18 zeroes, INR, 1 zeroes, 2 ones.

Text for function ‘Normal’ of ‘TopLevel\TopLevel\Core8051\PCunit\PCbufop’:
"Hold old register contents."
Section of Digital Information Systems

Enhanced 8051 Core

PCo := in.
out := in.

Text for function 'PCEim' of 'TopLevel\TopLevel\Core8051\PCunit\PCbufop':

-- STORE third byte of PC in PCbuf.
out := IM. (in from: 0 to: 15).
PCo := IM. (in from: 0 to: 15).

Text for function 'PCEram' of 'TopLevel\TopLevel\Core8051\PCunit\PCbufop':

-- STORE third byte of PC in PCbuf.
out := RAM. (in from: 0 to: 15).
PCo := RAM. (in from: 0 to: 15).

Text for function 'PCHim' of 'TopLevel\TopLevel\Core8051\PCunit\PCbufop':

-- STORE new second byte of PC in PCbuf.
out := (in from: 16 to: 23), IM. (in from: 0 to: 7).
PCo := (in from: 16 to: 23), IM. (in from: 0 to: 7).

Text for function 'PCHRam' of 'TopLevel\TopLevel\Core8051\PCunit\PCbufop':

-- STORE new second byte of PC in PCbuf.
out := (in from: 16 to: 23), RAM. (in from: 0 to: 7).
PCo := (in from: 16 to: 23), RAM. (in from: 0 to: 7).

Text for function 'PCLim' of 'TopLevel\TopLevel\Core8051\PCunit\PCbufop':

-- STORE new first byte of PC in PCbuf.
out := (in from: 8 to: 23), IM.
PCo := (in from: 8 to: 23), IM.

Text for function 'PCLram' of 'TopLevel\TopLevel\Core8051\PCunit\PCbufop':

-- STORE new first byte of PC in PCbuf.
out := (in from: 8 to: 23), RAM.
PCo := (in from: 8 to: 23), RAM.

This operator has 3 functions.
The default function is 'PCL'.

Output connector (8 bits) with name 'PCdat'
Input connector (24 bits) with name 'PCi'

Text for function 'PCE' of 'TopLevel\TopLevel\Core8051\PCunit\PCdatop':

-- OUTPUT third byte of PC.
PCdat := PCi from: 16 to: 23.

Text for function 'PCH' of 'TopLevel\TopLevel\Core8051\PCunit\PCdatop':

-- OUTPUT second byte of PC.
PCdat := PCi from: 8 to: 15.

Text for function 'PCL' of 'TopLevel\TopLevel\Core8051\PCunit\PCdatop':

-- OUTPUT first byte of PC.
PCdat := PCi from: 0 to: 7.

This operator has 6 functions.
The default function is 'Normal'.

Input connector (8 bits) with name 'A'
Input connector (24 bits) with name 'CP'
Input connector (10 bits) with name 'JR'
Input connector (24 bits) with name 'PCbuf'
Input connector (24 bits) with name 'PCi'
Output connector (24 bits) with name 'PCo'
Text for function 'ABS' of 'TopLevel\TopLevel\Core8051\PCunit\PCgen':

"Absolute call or jump."

PC0 := (PCI from: 11 to: 23), (IR from: 5 to: 7), (PCbuf from: 0 to: 7).

Text for function 'GLOBAL' of 'TopLevel\TopLevel\Core8051\PCunit\PCgen':

"Global call or jump."

PC0 := PCbuf.

Text for function 'LONG' of 'TopLevel\TopLevel\Core8051\PCunit\PCgen':

"Long call or jump."

PC0 := (PCI from: 16 to: 23), (PCbuf from: 0 to: 15).

Text for function 'MOVC_DP' of 'TopLevel\TopLevel\Core8051\PCunit\PCgen':

"MOVC A,@A+DPTR or JMP @A+DPTR"

PC0 := (16 zeroes, A) + DP.

Text for function 'MOVC_PC' of 'TopLevel\TopLevel\Core8051\PCunit\PCgen':

"MOVC A,@A+PC"

PC0 := (16 zeroes, A) + PCI.

Text for function 'Normal' of 'TopLevel\TopLevel\Core8051\PCunit\PCgen':

"Read instruction code."

PC0 := PCI.

'TopLevel\TopLevel\Core8051\PCunit\PCinc' is an operator.

This operator has 2 functions and is controlled by an unnamed control input. The default function is 'Normal'.

Control connector (1 bit) without a name

Control specification:

"Select program counter increment."

$1 AddOne.

Input connector (24 bits) with name 'in'

Output connector (24 bits) with name 'out'

Text for function 'AddOne' of 'TopLevel\TopLevel\Core8051\PCunit\PCinc':

"Increment program counter before storing."

out := in + (23 zeroes, 1 ones).

Text for function 'Normal' of 'TopLevel\TopLevel\Core8051\PCunit\PCinc':

"Throughput program counter."

out := in.

'TopLevel\TopLevel\Core8051\PCunit\PCop' is an operator.

This operator has 2 functions. The default function is 'CTRL'.

Input connector (1 bit) with name 'CTRL'

Output connector (1 bit) with name 'PC'

Text for function 'CTRL' of 'TopLevel\TopLevel\Core8051\PCunit\PCop':

"Control program counter."

PC := CTRL.

Text for function 'INC' of 'TopLevel\TopLevel\Core8051\PCunit\PCop':

"Load and increment program counter."

PC := 1 ones.
"TopLevel\TopLevel\Core80S1\PCunit\RELop" is an operator.

This operator has 2 functions and is controlled by an unnamed control input. The default function is 'FORWARD'.

Control connector (8 bits) without a name

Control specification:
- Detect forward or backward jump.
- %0 FORWARD.
- %1 BACKWARD.

Input connector (24 bits) with name 'PCI'
Output connector (24 bits) with name 'PCo'
Input connector (8 bits) with name 'REL'

Text for function 'BACKWARD' of 'TopLevel\TopLevel\Core80S1\PCunit\RELop':
- Backward relative jump.
- PCo := PCI + (16 ones, Rel).

Text for function 'FORWARD' of 'TopLevel\TopLevel\Core80S1\PCunit\RELop':
- Forward relative jump.
- PCo := PCI + (16 zeroes, Rel).

'TopLevel\TopLevel\Core80S1\PCunit\WR' is an operator.

This operator has 3 functions. The default function is 'Normal'.

Input connector (1 bit) with name 'CTRL'
Output connector (2 bits) with name 'WRo'

Text for function 'Normal' of 'TopLevel\TopLevel\Core80S1\PCunit\WR':
- Read new instruction during last cycle.
- WRo := 1 zeroes, CTRL.

Text for function 'Read' of 'TopLevel\TopLevel\Core80S1\PCunit\WR':
- Read control signal.
- WRo := 1 zeroes, 1 ones.

Text for function 'Write' of 'TopLevel\TopLevel\Core80S1\PCunit\WR':
- Write control signal.
- WRo := 1 ones, 1 zeroes.

'TopLevel\TopLevel\Core80S1\RAM_SEL' is a schematic.

Bidirectional connector (10 bits) with name 'IR'
Bidirectional connector (8 bits) with name 'PSWj'
Bidirectional connector (8 bits) with name 'RAMADDR'
Bidirectional connector (3 bits) with name 'RAM1WR'
Bidirectional connector (10 bits) with name 'SFR'
Bidirectional connector (16 bits) with name 'SPaddr'
Bidirectional connector (2 bits) with name 'SPWR'

'TopLevel\TopLevel\Core80S1\RAM_SEL\IWR_SEL' is an operator.

This operator has 3 functions and is controlled by an unnamed control input. The default function is 'Stack'.

Control connector (10 bits) without a name

Control specification:
- Select RAM access mode.
- %0100010000, 'JBC bit, rel'
- %0100010000, 'JB bit, rel'
- %01000110000, 'JNB bit, rel'
Section of Digital Information Systems

Enhanced 8051 Core

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0111000000</td>
<td>ORL C, /bit</td>
<td>Direct</td>
</tr>
<tr>
<td>0111000001</td>
<td>ANL C, /bit</td>
<td>Direct</td>
</tr>
<tr>
<td>0111000000</td>
<td>PUSH dir</td>
<td>Direct</td>
</tr>
<tr>
<td>0111000010</td>
<td>POP dir</td>
<td>Direct</td>
</tr>
<tr>
<td>0110100010</td>
<td>ORL dir, A</td>
<td>Direct</td>
</tr>
<tr>
<td>0110100010</td>
<td>ANL dir, A</td>
<td>Direct</td>
</tr>
<tr>
<td>0110110010</td>
<td>ORL C, bit</td>
<td>Direct</td>
</tr>
<tr>
<td>0110000010</td>
<td>MOV bit, C</td>
<td>Direct</td>
</tr>
<tr>
<td>0110110010</td>
<td>MOV C, bit</td>
<td>Direct</td>
</tr>
<tr>
<td>0110110010</td>
<td>MOV dir, #data</td>
<td>Direct</td>
</tr>
<tr>
<td>0110110010</td>
<td>MOV dir, #data</td>
<td>Direct</td>
</tr>
<tr>
<td>0110000100</td>
<td>INC dir</td>
<td>Direct</td>
</tr>
<tr>
<td>0110010010</td>
<td>ADD A, dir</td>
<td>Direct</td>
</tr>
<tr>
<td>0110010010</td>
<td>ADDC A, dir</td>
<td>Direct</td>
</tr>
<tr>
<td>0110100010</td>
<td>CLR dir</td>
<td>Direct</td>
</tr>
<tr>
<td>0110110010</td>
<td>MOV dir, #data</td>
<td>Direct</td>
</tr>
</tbody>
</table>

---

Output connector (8 bits) with name 'RAM_ADDR'
Output connector (3 bits) with name 'RAM_WR'
Input connector (8 bits) with name 'REG_ADDR'
Input connector (3 bits) with name 'REG_WR'
Input connector (10 bits) with name 'SFR'
Input connector (16 bits) with name 'SF_ADDR'
Input connector (2 bits) with name 'SPWR'

Text for function 'Direct' of 'TopLevel\TopLevel\Core8051\RAM_SEL\IWR_SEL':
"Direct addressing."
RAMIWR := (1 zeroes, (SFR at: 8) \ (SFR at: 7) not), (SFR at: 9) \ (SFR at: 7) not).
RAMADDR := SFR from: 0 to: 7.

Text for function 'REG' of 'TopLevel\TopLevel\Core8051\RAM_SEL\IWR_SEL':

"Register addressing."
RAMIWR := REGIWR.
RAMADDR := REGADDR.

Text for function 'Stack' of 'TopLevel\TopLevel\Core8051\RAM_SEL\IWR_SEL':

"Stack operation."
RAMIWR := (SPaddr at: 8) \ (SPaddr at: 9) \ (SPaddr at: 10) \ (SPaddr at: 11) \ (SPaddr at: 12) \ (SPaddr at: 13) \ (SPaddr at: 14) \ (SPaddr at: 15)) if0: (1 zeroes, SPWR)
if1: (3 zeroes).
RAMADDR := SPaddr from: 0 to: 7.

Control specification:

"Select register addressing control."
%010010011x, 'ADD A, @Ri'
%010011011x, 'ADD C A, @Ri'
%010100011x, 'ORL A, @Ri'
%010101011x, 'ANL A, @Ri'
%010110111x, 'MOV dir, @Ri'
%011001111x, 'SUBB A, @Ri'
%011101111x, 'CJNE @Ri, Kdata, rel'
%011100111x, 'MOV A, @Ri'
%010001011x, 'DEC @Ri'
%011000111x, 'INC @Ri'
%011001011x, 'SUBB @Ri'
%011100011x, 'XCH A, @Ri'
%011101011x, 'XCHD A, @Ri'
%010000111x, 'INC Rn'
%011001011x, 'DEC Rn'
%011110111x, 'MOV Rn, dir'
%011111111x Wrt. 'MOV Rn, A'
%010000011x, 'INC Rn'
%010100011x, 'DEC Rn'
%011100111x, 'XCH A, Rn'
%011101111x, 'XCHD A, Rn'
%011110111x Modify. 'DJNZ Rn, rel'

Control connector (10 bits) without a name

Input connector (10 bits) with name 'IR'
Input connector (8 bits) with name 'PSWj'
Output connector (8 bits) with name 'RAMADDR'
Output connector (3 bits with name 'RAMIWR'

Text for function 'Modify' of 'TopLevel\TopLevel\Core8051\RAM_SEL\REG_SEL':

"Modify register."
RAMIWR := 3 zeroes, (PSWj from: 3 to: 4), ((IR at: 3) if0: (2 zeroes, (IR at: 0))
if1: (IR from: 0 to: 2)).
RAMADDR := (IR at: 3) not, 2 ones.
Text for function 'MOVX' of 'TopLevel\TopLevel\Core8051\RAM_SEL\REG_SEL':

"Read indirect address."

Text for function 'NOP' of 'TopLevel\TopLevel\Core8051\RAM_SEL\REG_SEL':

"No register addressing."
RAMIWR := 0 zeroes.

Text for function 'Read' of 'TopLevel\TopLevel\Core8051\RAM_SEL\REG_SEL':

"Read register."
RAMADDR := 3 zeroes, (PSW: from: 3 to: 4), (IR: at: 3) if 0: (2 zeroes, (IR: at: 0)) if 1: (IR: from: 0 to: 2)).
RAMIWR := (IR: at: 3) not, 1 zeroes, 1 ones.

Text for function 'Wrt' of 'TopLevel\TopLevel\Core8051\RAM_SEL\REG_SEL':

"Write register."
RAMADDR := 3 zeroes, (PSW: from: 3 to: 4), (IR: at: 3) if 0: (2 zeroes, (IR: at: 0)) if 1: (IR: from: 0 to: 2)).
RAMIWR := (IR: at: 3) not, 1 zeroes, 1 ones.

"TopLevel\TopLevel\Core8051\RAMunit' is a schematic.

Designer comments:

RAM address 0-31 selects asynchronous read and write RAM.
RAM address 32-255 selects a RAM with synchronous read.
RAMDAT is RAM data input bus.
RAMDATO is RAM data output bus.
RAMADDR is RAM address bus.
RAMIWR is RAM read-write control bus, where I (bit 2) indicates indirect addressing mode.
RAMDV is RAM data valid flags bus.
Data valid read flag (bit 0) indicates data can be read from data input bus.
Data valid write flag (bit 1) indicates data from the data input bus will be written in RAM.

Control connector (8 bits) with name 'ADDR'
Control specification:

"Select RAM addressing mode."
S0 REGRAM.
S1,S3,S5,S7 RAM1.
S2, S6 RAM2.
S4 RAM3.

Input connector (8 bits) with name 'ADD'
Output connector (8 bits) with name 'HADDR'
Output connector (2 bits) with name 'HWmod'
Output connector (2 bits) with name 'HWR'
Output connector (7 bits) with name 'LWRaddr'
Input connector (2 bits) with name 'WR'

Text for function 'RAM1' of 'TopLevel\TopLevel\Core8051\RAMunit\ADDRgen':

"Invert bit 5 of address for synchronous read RAM."
HWMS := (WR at: 1) / (WR at: 0) not, (WR at: 0).
HWmod := (WR at: 1) / (WR at: 0), 1 zeroes.

Bidirectional connector (8 bits) with name 'DIRout1'
Bidirectional connector (8 bits) with name 'DI stresses'
Bidirectional connector (10 bits) with name 'IR'
Bidirectional connector (8 bits) with name 'PODat'
Bidirectional connector (8 bits) with name 'RAMADDR'
Bidirectional connector (8 bits) with name 'RAMDAT'
Bidirectional connector (2 bits) with name 'RAMDV'
Bidirectional connector (3 bits) with name 'RAMIWR'

'TopLevel\TopLevel\Core8051\RAMunit\ADDRgen' is an operator.

This operator has 4 functions and is controlled by an unnamed control input.
The default function is 'REGRAM'.

Control connector (8 bits) without a name
LWRaddr := 2 zeroes. (ADDR from: 0 to: 4).
HADDR := (ADDR from: 6 to: 7). (ADDR at: 5) not. (ADDR from: 0 to: 4).

Text for function 'RAM2' of 'TopLevel\TopLevel\Core8051\RAMunit\ADDRgen':

"*Invert bit 5 and 6 for synchronous read RAM."\nHWR := ((WR at: 1) \ (WR at: 0) not), (WR at: 0).
HWmod := ((WR at: 1) \ (WR at: 0)). 1 zeroes.
LWRaddr := 2 zeroes. (ADDR from: 0 to: 4).
HADDR := (ADDR from: 5 to: 7). (ADDR from: 0 to: 4).

Text for function 'RAM3' of 'TopLevel\TopLevel\Core8051\RAMunit\ADDRgen':

"Invert three highest address bits for synchronous read RAM.
HWR := ((WR at: 1) \ (WR at: 0) not), (WR at: 0).
HWmod := ((WR at: 1) \ (WR at: 0)), 1 zeroes.
LWRaddr := 2 zeroes. (ADDR from: 0 to: 4).
HADDR := (ADDR from: 5 to: 7) not. (ADDR from: 0 to: 4).

Text for function 'REGRAM' of 'TopLevel\TopLevel\Core8051\RAMunit\ADDRgen':

"Asynchronous RAM."
LWRaddr := WR, (ADDR from: 0 to: 4).
HWR := 2 zeroes.
HWmod := 2 zeroes.

"TopLevel\TopLevel\Core8051\RAMunit\ASALRAM" is a schematic.

Bidirectional connector ( 7 bits) with name 'LWRaddr'
Bidirectional connector ( 8 bits) with name 'rd'
Bidirectional connector ( 8 bits) with name 'wd'

"TopLevel\TopLevel\Core8051\RAMunit\ASALRAM\BORO" is a register.

This register is 8 bits wide and is controlled by an unnamed control input.
The default function is 'hold'.
This register is loaded with unknown values after a system reset.
The value loaded for the 'reset' command is 0.

Control specification:
%01000000 enable. "Read"
%10000000 load. "Write"
%11000000 load; enable. "Modify"

"TopLevel\TopLevel\Core8051\RAMunit\ASALRAM\BOR2" is a register.

This register is 8 bits wide and is controlled by an unnamed control input.
The default function is 'hold'.
This register is loaded with unknown values after a system reset.
The value loaded for the 'reset' command is 0.

Control specification:
%01000010 enable. "Read"
%10000010 load. "Write"
%11000010 load; enable. "Modify"

"TopLevel\TopLevel\Core8051\RAMunit\ASALRAM\BOR2" is a register.

This register is 8 bits wide and is controlled by an unnamed control input.
The default function is 'hold'.
This register is loaded with unknown values after a system reset.
The value loaded for the 'reset' command is 0.

Control specification:
%01000100 enable. "Read"
%10000100 load. "Write"
%11000100 load; enable. "Modify"
`TopLevel\TopLevel\Core80S1\RAMunit\ASALRAM\BOR3` is a register.

This register is 8 bits wide and is controlled by an unnamed control input. The default function is 'hold'. This register is loaded with unknown values after a system reset.

The value loaded for the 'reset' command is 0.

Control specification:

```plaintext
%0100011 enable. "Read"
%1000011 load; enable. "Modify"

TopLevel\TopLevel\Core80S1\RAMunit\ASALRAM\BOR4` is a register.

This register is 8 bits wide and is controlled by an unnamed control input. The default function is 'hold'. This register is loaded with unknown values after a system reset.

The value loaded for the 'reset' command is 0.

Control specification:

```plaintext
%0100100 enable. "Read"
%1000100 load. "Write"
%1100100 load; enable. "Modify"

TopLevel\TopLevel\Core80S1\RAMunit\ASALRAM\BOR5` is a register.

This register is 8 bits wide and is controlled by an unnamed control input. The default function is 'hold'. This register is loaded with unknown values after a system reset.

The value loaded for the 'reset' command is 0.

Control specification:

```plaintext
%0100101 enable. "Read"
%1000101 load. "Write"
%1100101 load; enable. "Modify"

TopLevel\TopLevel\Core80S1\RAMunit\ASALRAM\BOR6` is a register.

This register is 8 bits wide and is controlled by an unnamed control input. The default function is 'hold'. This register is loaded with unknown values after a system reset.

The value loaded for the 'reset' command is 0.

Control specification:

```plaintext
%0100110 enable. "Read"
%1000110 load. "Write"
%1100110 load; enable. "Modify"

TopLevel\TopLevel\Core80S1\RAMunit\ASALRAM\BOR7` is a register.

This register is 8 bits wide and is controlled by an unnamed control input. The default function is 'hold'. This register is loaded with unknown values after a system reset.

The value loaded for the 'reset' command is 0.

Control specification:

```plaintext
%0100111 enable. "Read"
%1000111 load. "Write"
%1100111 load; enable. "Modify"

TopLevel\TopLevel\Core80S1\RAMunit\ASALRAM\BOR0` is a register.

This register is 8 bits wide and is controlled by an unnamed control input. The default function is 'hold'. This register is loaded with unknown values after a system reset.

The value loaded for the 'reset' command is 0.
The value loaded for the 'reset' command is 0.

Control specification:

0101000 enable. "Read"
1001000 load. "Write"
1101000 load; enable. "Modify"

'TopLevel\TopLevel\Core80S1\RAMunit\ASALRAM\BIR1' is a register.

This register is 8 bits wide and is controlled by an unnamed control input. The default function is 'hold'. This register is loaded with unknown values after a system reset.

The value loaded for the 'reset' command is 0.

Control specification:

0101001 enable. "Read"
1001001 load. "Write"
1101001 load; enable. "Modify"

'TopLevel\TopLevel\Core80S1\RAMunit\ASALRAM\BIR2' is a register.

This register is 8 bits wide and is controlled by an unnamed control input. The default function is 'hold'. This register is loaded with unknown values after a system reset.

The value loaded for the 'reset' command is 0.

Control specification:

0101010 enable. "Read"
1001010 load. "Write"
1101010 load; enable. "Modify"

'TopLevel\TopLevel\Core80S1\RAMunit\ASALRAM\BIR3' is a register.

This register is 8 bits wide and is controlled by an unnamed control input. The default function is 'hold'. This register is loaded with unknown values after a system reset.

The value loaded for the 'reset' command is 0.

Control specification:

0101011 enable. "Read"
1001011 load. "Write"
1101011 load; enable. "Modify"

'TopLevel\TopLevel\Core80S1\RAMunit\ASALRAM\BIR4' is a register.

This register is 8 bits wide and is controlled by an unnamed control input. The default function is 'hold'. This register is loaded with unknown values after a system reset.

The value loaded for the 'reset' command is 0.

Control specification:

0101100 enable. "Read"
1001100 load. "Write"
1101100 load; enable. "Modify"

'TopLevel\TopLevel\Core80S1\RAMunit\ASALRAM\BIR5' is a register.

This register is 8 bits wide and is controlled by an unnamed control input. The default function is 'hold'. This register is loaded with unknown values after a system reset.

The value loaded for the 'reset' command is 0.

Control specification:

0101101 enable. "Read"
1001101 load. "Write"
This register is 8 bits wide and is controlled by an unnamed control input. The default function is 'hold'.
This register is loaded with unknown values after a system reset.

The value loaded for the 'reset' command is 0.

Control specification:

```
%1101101 load; enable. "Modify"
```

'TopLevel\TopLevel\Core8051\RAMunit\ASALRAM\B1R6' is a register.

This register is 8 bits wide and is controlled by an unnamed control input. The default function is 'hold'.
This register is loaded with unknown values after a system reset.

The value loaded for the 'reset' command is 0.

Control specification:

```
%1010110 enable. "Read"
%1001110 load. "Write"
%1101110 load; enable. "Modify"
```

'TopLevel\TopLevel\Core8051\RAMunit\ASALRAM\B1R7' is a register.

This register is 8 bits wide and is controlled by an unnamed control input. The default function is 'hold'.
This register is loaded with unknown values after a system reset.

The value loaded for the 'reset' command is 0.

Control specification:

```
%0110000 enable. "Read"
%1010000 load. "Write"
%1110000 load; enable. "Modify"
```

'TopLevel\TopLevel\Core8051\RAMunit\ASALRAM\B2R0' is a register.

This register is 8 bits wide and is controlled by an unnamed control input. The default function is 'hold'.
This register is loaded with unknown values after a system reset.

The value loaded for the 'reset' command is 0.

Control specification:

```
%0110001 enable. "Read"
%1010001 load. "Write"
%1110001 load; enable. "Modify"
```

'TopLevel\TopLevel\Core8051\RAMunit\ASALRAM\B2R1' is a register.

This register is 8 bits wide and is controlled by an unnamed control input. The default function is 'hold'.
This register is loaded with unknown values after a system reset.

The value loaded for the 'reset' command is 0.

Control specification:

```
%0110010 enable. "Read"
%1010010 load. "Write"
%1110010 load; enable. "Modify"
```

'TopLevel\TopLevel\Core8051\RAMunit\ASALRAM\B2R2' is a register.

This register is 8 bits wide and is controlled by an unnamed control input. The default function is 'hold'.
This register is loaded with unknown values after a system reset.

The value loaded for the 'reset' command is 0.

Control specification:

```
%0110011 enable. "Read"
%1010011 load. "Write"
%1110011 load; enable. "Modify"
```

'TopLevel\TopLevel\Core8051\RAMunit\ASALRAM\B2R3' is a register.

This register is 8 bits wide and is controlled by an unnamed control input.
The default function is 'hold'.
This register is loaded with unknown values after a system reset.

The value loaded for the 'reset' command is 0.

Control specification:
%0110011 enable. "Read"
%010011 load. "Write"
%1110011 load; enable. "Modify"

'TopLevel\TopLevel\Core8051\RAMunit\ASALRAM\B2R4' is a register.
This register is 8 bits wide and is controlled by an unnamed control input.
The default function is 'hold'.
This register is loaded with unknown values after a system reset.

The value loaded for the 'reset' command is 0.

Control specification:
%0110100 enable. "Read"
%1010100 load. "Write"
%1110100 load; enable. "Modify"

'TopLevel\TopLevel\Core8051\RAMunit\ASALRAM\B2R5' is a register.
This register is 8 bits wide and is controlled by an unnamed control input.
The default function is 'hold'.
This register is loaded with unknown values after a system reset.

The value loaded for the 'reset' command is 0.

Control specification:
%0110111 enable. "Read"
%1010111 load. "Write"
%1110111 load; enable. "Modify"

'TopLevel\TopLevel\Core8051\RAMunit\ASALRAM\B2R6' is a register.
This register is 8 bits wide and is controlled by an unnamed control input.
The default function is 'hold'.
This register is loaded with unknown values after a system reset.

The value loaded for the 'reset' command is 0.

Control specification:
%0110110 enable. "Read"
%0101110 load. "Write"
%1110110 load; enable. "Modify"

'TopLevel\TopLevel\Core8051\RAMunit\ASALRAM\B2R7' is a register.
This register is 8 bits wide and is controlled by an unnamed control input.
The default function is 'hold'.
This register is loaded with unknown values after a system reset.

The value loaded for the 'reset' command is 0.

Control specification:
%0110111 enable. "Read"
%1010111 load. "Write"
%1110111 load; enable. "Modify"

'TopLevel\TopLevel\Core8051\RAMunit\ASALRAM\B3R0' is a register.
This register is 8 bits wide and is controlled by an unnamed control input.
The default function is 'hold'.
This register is loaded with unknown values after a system reset.

The value loaded for the 'reset' command is 0.

Control specification:
This register is 8 bits wide and is controlled by an unnamed control input. The default function is 'hold'. This register is loaded with unknown values after a system reset.

The value loaded for the 'reset' command is 0.

Control specification:

%0111011 enable. *Read*
%1011011 load. *Write*
%1111101 load; enable. *Modify*
'TopLevel\TopLevel\Core8051\RAMunit\ASALRAM\B3R6' is a register.

This register is 8 bits wide and is controlled by an unnamed control input. The default function is 'hold'. This register is loaded with unknown values after a system reset.

The value loaded for the 'reset' command is 0.

Control specification:

1011110 enable. "Read"
1011110 load. "Write"
1111110 load; enable. "Modify"

'TopLevel\TopLevel\Core8051\RAMunit\ASALRAM\B3R7' is a register.

This register is 8 bits wide and is controlled by an unnamed control input. The default function is 'hold'. This register is loaded with unknown values after a system reset.

The value loaded for the 'reset' command is 0.

Control specification:

1011111 enable. "Read"
1011111 load. "Write"
1111111 load; enable. "Modify"

'TopLevel\TopLevel\Core8051\RAMunit\BUSIMPL' is a schematic.

Designer comments:

This schematic contains a control input driven RAM, as it should be inserted in a design before transfer to ASA. The synthesized result will have the same behaviour as BUSIMU, which should be used during simulation of the system.

Bidirectional connector (2 bits) with name 'c'
Bidirectional connector (8 bits) with name 'ra'
Bidirectional connector (8 bits) with name 'rd'
Bidirectional connector (8 bits) with name 'wa'
Bidirectional connector (8 bits) with name 'wd'

'TopLevel\TopLevel\Core8051\RAMunit\BUSIMPL\RAM' is a RAM.

This RAM contains 224 words of 8 bits each and is controlled by an unnamed control input. There is no contents file attached. This RAM is loaded with unknown values after a system reset.

Control specification:

%10 Write.
%01 Enable

'TopLevel\TopLevel\Core8051\RAMunit\Busy' is a register.

This register is 1 bit wide and is controlled by an unnamed control input. The default function is 'hold'. This register is loaded with value 0 following system reset.

The value loaded for the 'reset' command is 0.

Control specification:

%00 load.
%01 reset.
%10 reset.
%11 reset.

'TopLevel\TopLevel\Core8051\RAMunit\Busyop' is an operator.

This operator has 2 functions.
The default function is 'Normal'.

Input connector (2 bits) with name 'DV1'
Output connector (2 bits) with name 'DV0'

Text for function 'Normal' of 'TopLevel\TopLevel\Core8051\RAMunit\Busyop':

"Throughput data valid signal."
DV0 := DV1.

Text for function 'RET' of 'TopLevel\TopLevel\Core8051\RAMunit\Busyop':

"Continued stack pop action."
DV0 := 1 zeroes, 1 ones.

'TopLevel\TopLevel\Core8051\RAMunit\DVop' is an operator.
This operator has 2 functions and is controlled by an unnamed control input.
The default function is 'Normal'.

Control connector (3 bits) without a name

Control specification:

'If indirect register read, no data valid flags.'
(2)
% Indirect.

Input connector (2 bits) with name 'c'
Output connector (2 bits) with name 'DV'
Input connector (2 bits) with name 'HWR'
Input connector (7 bits) with name 'LWR'

Text for function 'Indirect' of 'TopLevel\TopLevel\Core8051\RAMunit\DVop':

"Indirect register read. OV = %00."
DV := 2 zeroes.

Text for function 'Normal' of 'TopLevel\TopLevel\Core8051\RAMunit\DVop':

"Compose data valid flags."
DV := (LWR from: 5 to: 6) \ (1 zeroes, (HWR at: 0)) \ ((c at: 1), 1 zeroes).

'TopLevel\TopLevel\Core8051\RAMunit\HADDRbuf' is a register.
This register is 8 bits wide.
The default function is 'hold'.
This register is loaded with unknown values after a system reset.
The value loaded for the 'reset' command is 0.

'TopLevel\TopLevel\Core8051\RAMunit\HRAM' is an operator.
This operator has 2 functions and is controlled by an unnamed control input.
The default function is 'Normal'.

Control connector (2 bits) without a name

Control specification:

(1)
% Normal.
% Modify. "Modify operation."

Output connector (2 bits) with name 'c'
Input connector (8 bits) with name 'HADD'
Input connector (8 bits) with name 'HADDbuf'
Input connector (2 bits) with name 'HWmod'
Input connector (2 bits) with name 'HWR'
Output connector (8 bits) with name 'ra'
Output connector (8 bits) with name 'wa'

Text for function 'Modify' of 'TopLevel\TopLevel\Core8051\RAMunit\HRAM':

wa := HADDbuf.
ra := HADDbuf.
c := HWmod.
Text for function 'Normal' of 'TopLevel\TopLevel\Core8051\RAMunit\HRAM':

\texttt{wa := HADDR.}
\texttt{ra := HADDR.}
\texttt{c := HWR.}

`TopLevel\TopLevel\Core8051\RAMunit\HWmod` is a register.

This register is 2 bits wide.
The default function is 'load'.
This register is loaded with value 0 following system reset.

The value loaded for the 'reset' command is 0.

`TopLevel\TopLevel\Core8051\RAMunit\HWR` is a register.

This register is 2 bits wide.
The default function is 'load'.
This register is loaded with value 0 following system reset.

The value loaded for the 'reset' command is 0.

`TopLevel\TopLevel\Core8051\RAMunit\HWR` is a register.

This register is 8 bits wide and is controlled by an unnamed control input.
The default function is 'hold'.
This register is loaded with value 0 following system reset.

The value loaded for the 'reset' command is 0.

\texttt{Control specification:}
\begin{itemize}
\item \texttt{\textbackslash\textbackslash load.}
\end{itemize}

`TopLevel\TopLevel\Core8051\RAMunit\TMR` is a register.

This register is 3 bits wide.
The default function is 'load'.
This register is loaded with value 0 following system reset.

The value loaded for the 'reset' command is 0.

`TopLevel\TopLevel\Core8051\RAMunit\RAMDATobuf` is a schematic.

Bidirectional connector ( 2 bits) with name 'DV'
Bidirectional connector ( 8 bits) with name 'in'
Bidirectional connector ( 8 bits) with name 'out'

`TopLevel\TopLevel\Core8051\RAMunit\RAMDATobuf\DATobufop` is an operator.

This operator has 2 functions and is controlled by control input 'DV'.
The default function is 'BUF'.

Input connector ( 8 bits) with name 'Buf'
Input connector ( 8 bits) with name 'DAT'
Control connector ( 2 bits) with name 'DV'

\texttt{Control specification:}
\begin{itemize}
\item \texttt{\textbackslash\textbackslash load. \textbackslash\textbackslash load.}
\end{itemize}
Section of Digital Information Systems

Enhanced 8051 Core

%1 RAM.

Output connector (8 bits) with name 'out'

Text for function 'BUF' of 'TopLevel\TopLevel\Core8051\RAMunit\RAMDATobuf\DATobufop' :

"RAM output buffer."
out := Buf.

Text for function 'RAM' of 'TopLevel\TopLevel\Core8051\RAMunit\RAMDATobuf\DATobufop' :

"RAM output."
out := DAT.

This operator has 3 functions and is controlled by an unnamed control input. The default function is 'STACK'.

Control connector (10 bits) without a name

Control specification:

"Select RAM data input."
%1010000000, "PUSH dir"
%100000011x, "INC @Ri"
%100000111x, "DEC @Ri"
%010111011x, "MOV @Ri,#data"
%1011000000, "PUSH dir"
%101000011x, "INC Rn"
%10100111xx, "DEC Rn"
%11011111xx, "MOV Rn,#data"
%1101111000, "MOV Rn.dir"%1101111001, "MOV Rn.dir"
%11011111xx, "DINZ Rn.rel"
%11111111xx ALU. "MOV Rn,A"
%1000010000, "JBC bit.rel"
%1111010000, "POP dir"
%101000001x, "ORL dir,A & #data"
%101010001x, "ANL dir,A & #data"
%101110010x, "XRL dir,A & #data"
%1101010100, "MOV bit,C"
%1101000100, "CPL bit"
%1111000000, "CLR bit"
%1111100010, "SETB bit"
%1100000010, "INC dir"
%1000010101, "DEC dir"
%1011010101, "MOV dir,#data"
%1001000010, "MOV dir,dir"
%1101100101, "XCH A.dir"
%1111010101, "DINZ dir.rel"
%1111110101, "MOV dir,A"
%1010000111x, "MOV dir,#Ri"
%1010001111xx Direct. "MOV dir,Rn"

Input connector (8 bits) with name 'ALOut1'
Input connector (8 bits) with name 'Direct'
Input connector (8 bits) with name 'FCdat'
Output connector (8 bits) with name 'RAM'

Text for function 'ALU' of 'TopLevel\TopLevel\Core8051\RAMunit\RAMiMUX' :

"Data from ALU."
RAM := ALOut1.

Text for function 'Direct' of 'TopLevel\TopLevel\Core8051\RAMunit\RAMiMUX' :

"Direct addressing."
RAM := Direct.

Text for function 'STACK' of 'TopLevel\TopLevel\Core8051\RAMunit\RAMiMUX' :

"PUSH stack data."
Section of Digital Information Systems

Enhanced 8051 Core

RAM := PCdat.

'TopLevel\TopLevel\Core80S1\RAMunit\RAMop' is an operator.

This operator has 2 functions and is controlled by an unnamed control input.
The default function is 'Busy'.

Control connector (1 bit) without a name

Control specification:

"RAM access control."
%0 IWR. 'RAM accessible.'
%1 Busy. 'No RAM access until data valid signal is detected.'

Output connector (1 bit) with name 'Busy'
Input connector (3 bits) with name 'IWR'
Input connector (3 bits) with name 'RAMIWR'
Input connector (2 bits) with name 'WR'

Text for function 'Busy' of 'TopLevel\TopLevel\Core80S1\RAMunit\RAMop':

"RAM is busy; no access."
IWR := 3 zeroes.
WR := 2 zeroes.
Busy := 1 ones.

Text for function 'IWR' of 'TopLevel\TopLevel\Core80S1\RAMunit\RAMop':

"Indirect register addressing."
IWR := (RAMIWR at: 2) if0: 0 zeroes
if1: RAMIWR. 'Store indirect action.'
WR := (RAMIWR at: 2) if0: (RAMIWR from: 0 to: 1)
if1: 1 zeroes, 1 ones. 'Read indirect address.'
Busy := (RAMIWR at: 0) \ (RAMIWR at: 1).

'TopLevel\TopLevel\Core80S1\RAMunit\RAMSelect' is an operator.

This operator has 2 functions and is controlled by an unnamed control input.
The default function is 'Normal'.

Control connector (3 bits) without a name

Control specification:

"Detect indirect addressing."
(2)
%0 Indirect.

Output connector (8 bits) with name 'ADDR'
Input connector (8 bits) with name 'IADDR'
Input connector (3 bits) with name 'IWR'
Input connector (8 bits) with name 'RAMADDR'
Input connector (2 bits) with name 'WR'
Output connector (2 bits) with name 'WRO'

Text for function 'Indirect' of 'TopLevel\TopLevel\Core80S1\RAMunit\RAMSelect':

"Select indirect RAM address."
WRO := IWR from: 0 to: 1.
ADDR := IADDR.

Text for function 'Normal' of 'TopLevel\TopLevel\Core80S1\RAMunit\RAMSelect':

"Select normal RAM address."
WRO := WR.
ADDR := (WR at: 0) \ (WR at: 1) if0: 8 zeroes
if1: RAMADDR.

'TopLevel\TopLevel\Core80S1\SP_CTRL' is a schematic.

Bidirectional connector (8 bits) with name 'DirDat'
Bidirectional connector (1 bit) with name 'Ready'
Bidirectional connector (10 bits) with name 'SPR'
Bidirectional connector (16 bits) with name 'SPAddr'
Bidirectional connector (8 bits) with name 'SPo'
Bidirectional connector (2 bits) with name 'SPWR'

'TopLevel\TopLevel\Core8051\SP_CTRL\POPop' is an operator.

This operator has 5 functions and is controlled by an unnamed control input. The default function is 'five'.

Control specification:

Control connector (16 bits) without a name

Control specification:

Output connector (3 bits) with name 'Qut'

Text for function 'five' of 'TopLevel\TopLevel\Core8051\SP_CTRL\POPop':

Text for function 'four' of 'TopLevel\TopLevel\Core8051\SP_CTRL\POPop':

Text for function 'one' of 'TopLevel\TopLevel\Core8051\SP_CTRL\POPop':

Text for function 'three' of 'TopLevel\TopLevel\Core8051\SP_CTRL\POPop':

Text for function 'two' of 'TopLevel\TopLevel\Core8051\SP_CTRL\POPop':

This register is 3 bits wide and is controlled by an unnamed control input. The default function is 'hold'. This register is loaded with value 0 following system reset.

The value loaded for the 'reset' command is 0.

Control specification:

Enhanced 8051 Core
Text for function 'NoDec' of 'TopLevel\TopLevel\Core8051\SP_CTRL\SP_POP':

*No decrement for POP SPL or SPH.*
WRO := 2 zeroes.

Text for function 'Normal' of 'TopLevel\TopLevel\Core8051\SP_CTRL\SP_POP':

*Increment or decrement stack pointer.*
WRO := WRI.

'TopLevel\TopLevel\Core8051\SP_CTRL\SP_reg' is a register.
The default function is 'load'.
This register is 16 bits wide.
The value loaded for the 'reset' command is 0.
This operator has 2 functions and is controlled by an unnamed control input.

Control connector (10 bits) without a name

Control specification:

Text for function 'SPH' of 'TopLevel\TopLevel\Core8051\SP_CTRL\SP_Sel':

SPo := SP16i from: 8 to: 15.
SP16o := SPi, (SP16i from: 0 to: 7).

Text for function 'SF-' of 'TopLevel\TopLevel\Core8051\SP_CTRL\SP_Sel':

SPo := SP16i from: 0 to: 7.
SP16o := (SP16i from: 8 to: 15), SPi.

'TopLevel\TopLevel\Core8051\SP_CTRL\SPop' is an operator.

This operator has 2 functions and is controlled by an unnamed control input.
The default function is 'OLD_SP'.

Control connector (10 bits) without a name

Control specification:

Text for function 'NEW_SP' of 'TopLevel\TopLevel\Core8051\SP_CTRL\SPop':

'Load new stack pointer.'
out := SP.

Text for function 'OLD_SP' of 'TopLevel\TopLevel\Core8051\SP_CTRL\SPop':

'No stack pointer change.'
out := in.

'TopLevel\TopLevel\Core8051\SP_CTRL\STACK' is an operator.

This operator has 3 functions.
The default function is 'Normal'.

- 95 -
Text for function 'Normal' of 'TopLevel\TopLevel\Core8051\SP_CTRL\STACK':

---v-----------------------------------
"No stack operation."
WR := 2 zeroes.

Text for function 'POP' of 'TopLevel\TopLevel\Core8051\SP_CTRL\STACK':

---v-----------------------------------
"POP from stack."
WR := 1 zeroes, 1 ones.

Text for function 'PUSH' of 'TopLevel\TopLevel\Core8051\SP_CTRL\STACK':

---v-----------------------------------
"PUSH onto stack."
WR := 1 ones, 1 zeroes.

This operator has 3 functions and is controlled by an unnamed control input.
The default function is 'Normal'.

Control connector (2 bits) without a name

Control specification:

%00 POP.
%10 PUSH.

Output connector (16 bits) with name 'ADDRo'
Input connector (16 bits) with name 'SPI'
Output connector (16 bits) with name 'SPo'

Text for function 'Normal' of 'TopLevel\TopLevel\Core8051\SP_CTRL\STACKop'::

---v-----------------------------------
"Throughput stack pointer."
SPo := SPI.
ADDRo := SPI.

Text for function 'TOP' of 'TopLevel\TopLevel\Core8051\SP_CTRL\STACKop':

---v-----------------------------------
"POP from stack."
SPo := SPI + 16 ones.
ADDRo := SPI.

Text for function 'PUSH' of 'TopLevel\TopLevel\Core8051\SP_CTRL\STACKop':

---v-----------------------------------
"Push onto stack."
_inc := SPI + (15 zeroes, 1 ones).
SPo := _inc.
ADDRo := _inc.

This register is 1 bit wide.
The default function is 'load'.
This register is loaded with value 0 following system reset.
The value loaded for the 'reset' command is 0.

'TopLevel\TopLevel\Core8051\SF_CTRL\SVop' is an operator.

This operator has 2 functions and is controlled by an unnamed control input.
The default function is 'Ready'.

Control connector (16 bits) without a name

Control specification:

(8..15) "Internal RAM."
%00000000 SET.

Output connector (1 bit) with name 'Flag'
Input connector (1 bit) with name 'Ready'
Text for function 'Ready' of 'TopLevel\TopLevel\Core8051\SP_CTRL\SVop':
Flag := Ready.

Text for function 'SET' of 'TopLevel\TopLevel\Core8051\SP_CTRL\SVop':
Flag := 1 ones.