10555 Scripps Trl, San Diego, CA 92131
858-877-3485 858-408-9550
http://www.pulserain.com

FP51 (FPGA based 1T 8051 core)

Highlights

1T 8051 Core

- ✓ Intel MCS-51 Compatible
- ✓ RISC Implementation
- ✓ Most instructions are single clock cycle execution
- ✓ 16 Bit Stack Pointer
- ✓ Silicon Proven

FPGA / ASIC Integration

- ✓ RTL code in System Verilog
- ✓ OCD (On Chip Debugger), with RS232 connection
- ✓ Wishbone Bus Interface, easy to add new peripherals
- ✓ Clock rate can reach more than 100MHz on Altera MAX10 device (-8 speed grade)
- ✓ Easy to use, Easy to integrate

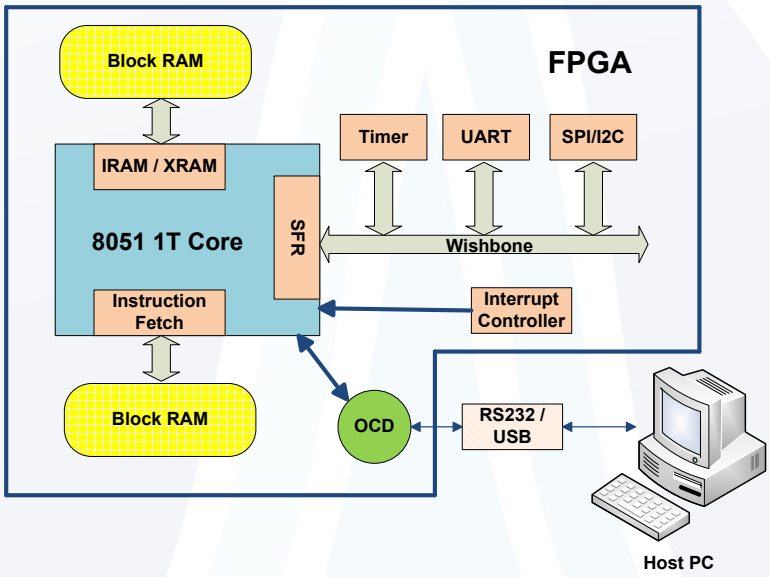
Software

- ✓ SDCC C Compiler
- ✓ Python Based Debug Console
- ✓ Arduino Language Support

Rev 1.03, Sep 2017

Doc # DS-0922-0010

Putting MCU core into FPGA is becoming a universal practice these days. However, most MCU soft cores available today are tied to specific FPGA vendors. To break the status quo, PulseRain Technology has come up with the FP51 - 1T core. It is a high performance 8 bit MCU core compatible with Intel 8051 ISA. With a crafty RISC implementation, the FP51 – 1T core can complete most instructions in just **single clock cycle**, while achieving a clock rate of more than **100MHz**. (Silicon Proven on the low speed grade device (-8 grade) in Altera MAX 10 family).



To facilitate debugging, an **OCD** (On Chip Debugger) is also provided along with the MCU core. The OCD can communicate with host PC through RS232 protocol. And it supports code updating, single step execution, hardware breakpoint etc.

Wishbone bus interface is supported through SFR (Special Function Registers). With great flexibility, new peripherals can be added and customized to target specific applications.

Software Support

- SDCC (Small Device C Compiler)
- Debug Console Software in Python Script
- Library to support the majority of Arduino Language

Memory Space

- Maximum 64KB code space
- Maximum 64KB data space
- 128 byte IRAM
- 128 byte SFR
- Maximum 63.75KB XRAM
- IRAM and XRAM are physically merged together for better memory utilization

Peripherals (Wishbone Bus)

- Timer
- UART
- LED Controller
- PWM
- SPI / I2C Transceiver
- I2S Transceiver

IP Implementation and Quality Metrics

Sample Configuration

- 8051 1T MCU Core
- 16 KB Code Memory, 16 KB Data Memory
- 2 Timer
- 1 Watchdog Timer / LED Controller
- 1 RS-232 UART, 1 JTAG UART
- OCD

Max Clock Rate (FMax)	*101.6 MHZ*
Core Size	*3875 Logic Elements*
FPGA Device	*Altera 10M08SAE144C8G*
Synthesis	*Quartus Prime 15.1*
Place and Route	*Quartus Prime 15.1*
Total Logic Elements	*5,541 / 8,064 (69%)*
Total Comb functions	*4,933 / 8,064 (61%)*
Dedicated logic registers	*2,361 / 8,064 (29%)*
Total Registers	*2363*
Total Memory Bits	*262,144 / 387,073 (5%)*
Embedded Multiplier (9 bit)	*2 / 48 (4%)*

Device Implementation Matrix - **Altera MAX 10 Family**

Instructions

Opcode	Mnemonic	Bytes	Cycle	Comment
0x00	NOP	1	1	No Operation
0x01	AJMP code	2	9	Absolute Jump
0x02	LJMP code	3	9	Long Jump
0x03	RR A	1	1	Rotate Right
0x04	INC A	1	1	Accumulator Increment
0x05	INC data	2	1	Direct Data Increment
0x06	INC @R0	1	1	Indirect Data Increment
0x07	INC @R1	1	1	Indirect Data Increment
0x08	INC R0	1	1	Register Increment
0x09	INC R1	1	1	Register Increment
0x0A	INC R2	1	1	Register Increment
0x0B	INC R3	1	1	Register Increment
0x0C	INC R4	1	1	Register Increment
0x0D	INC R5	1	1	Register Increment
0x0E	INC R6	1	1	Register Increment
0X0F	INC R7	1	1	Register Increment
0x10	JBC bit code	3	2/9	Jump if bit is set, and clear bit
0x11	ACALL code	2	9	Absolute Call
0x12	LCALL code	3	9	Long Call
0x13	RRC A	1	1	Rotate Right with Carry bit
0x14	DEC A	1	1	Accumulator Decrement
0x15	DEC data	2	1	Direct Data Decrement
0x16	DEC @R0	1	1	Indirect Data Decrement
0x17	DEC @R1	1	1	Indirect Data Decrement
0x18	DEC R0	1	1	Register Decrement
0x19	DEC R1	1	1	Register Decrement
0x1A	DEC R2	1	1	Register Decrement
0x1B	DEC R3	1	1	Register Decrement
0x1C	DEC R4	1	1	Register Decrement
0x1D	DEC R5	1	1	Register Decrement
0x1E	DEC R6	1	1	Register Decrement
0x1F	DEC R7	1	1	Register Decrement
0x20	JB bit code	3	2/9	Jump if bit is set
0x21	AJMP code	2	9	Absolute Jump
0x22	RET	1	10	Return from Subroutine Call
0x23	RL A	1	1	Rotate Left
0x24	ADD const	2	1	Add immediate data
0x25	ADD data	2	1	Add direct data
0x26	ADD A @R0	1	1	Add indirect data
0x27	ADD A @R1	1	1	Add indirect data
0x28	ADD A R0	1	1	Add Register
0x29	ADD A R1	1	1	Add Register
0x2A	ADD A R2	1	1	Add Register
0x2B	ADD A R3	1	1	Add Register
0x2C	ADD A R4	1	1	Add Register
0x2D	ADD A R5	1	1	Add Register
0x2E	ADD A R6	1	1	Add Register
0x2F	ADD A R7	1	1	Add Register

(Table Continues on next page)

Opcode	Mnemonic	Bytes	Cycle	Comment
0x30	JNB bit code	3	2/9	Jump if bit is not set
0x31	ACALL code	1	9	Absolute Subroutine Call
0x32	RETI	1	10	Return from ISR
0x33	RLC A	1	1	Rotate Left with Carry Bit
0x34	ADDC const	2	1	Add Immediate data plus Carry Bit
0x35	ADDC A data	2	1	Add Direct Data plus Carry Bit
0x36	ADDC A @R0	1	1	Add Indirect Data plus Carry Bit
0x37	ADDC A @R1	1	1	Add Indirect Data plus Carry Bit
0x38	ADDC A R0	1	1	Add Register plus Carry Bit
0x39	ADDC A R1	1	1	Add Register plus Carry Bit
0x3A	ADDC A R2	1	1	Add Register plus Carry Bit
0x3B	ADDC A R3	1	1	Add Register plus Carry Bit
0x3C	ADDC A R4	1	1	Add Register plus Carry Bit
0x3D	ADDC A R5	1	1	Add Register plus Carry Bit
0x3E	ADDC A R6	1	1	Add Register plus Carry Bit
0x3F	ADDC A R7	1	1	Add Register plus Carry Bit
0x40	JC code	2	2/9	Jump if carry bit is set
0x41	AJMP code	2	9	Absolute Jump
0x42	ORL data A	2	1	Logical OR Accumulator to direct data
0x43	ORL data const	3	1	Logical OR immediate data to direct data
0x44	ORL const	2	1	Logical OR immediate data to Accumulator
0x45	ORL A data	2	1	Logical OR Direct Data to Accumulator
0x46	ORL A @R0	1	1	Logical OR indirect data to Accumulator
0x47	ORL A @R1	1	1	Logical OR indirect data to Accumulator
0x48	ORL A R0	1	1	Logical OR Register to Accumulator
0x49	ORL A R1	1	1	Logical OR Register to Accumulator
0x4A	ORL A R2	1	1	Logical OR Register to Accumulator
0x4B	ORL A R3	1	1	Logical OR Register to Accumulator
0x4C	ORL A R4	1	1	Logical OR Register to Accumulator
0x4D	ORL A R5	1	1	Logical OR Register to Accumulator
0x4E	ORL A R6	1	1	Logical OR Register to Accumulator
0x4F	ORL A R7	1	1	Logical OR Register to Accumulator
0x50	JNC code	2	2/9	Jump if carry bit is not set
0x51	ACALL code	2	9	Absolute Subroutine Call
0x52	ANL data A	2	1	Logical AND Accumulator to direct data
0x53	ANL data const	3	1	Logical AND immediate data to direct data
0x54	ANL const	2	1	Logical AND immediate data to Accumulator

Opcode	Mnemonic	Bytes	Cycle	Comment
0x55	ANL A data	2	1	Logical AND Direct Data to Accumulator
0x56	ANL A @R0	1	1	Logical AND indirect data to Accumulator
0x57	ANL A @R1	1	1	Logical AND indirect data to Accumulator
0x58	ANL A R0	1	1	Logical AND Register to Accumulator
0x59	ANL A R1	1	1	Logical AND Register to Accumulator
0x5A	ANL A R2	1	1	Logical AND Register to Accumulator
0x5B	ANL A R3	1	1	Logical AND Register to Accumulator
0x5C	ANL A R4	1	1	Logical AND Register to Accumulator
0x5D	ANL A R5	1	1	Logical AND Register to Accumulator
0x5E	ANL A R6	1	1	Logical AND Register to Accumulator
0x5F	ANL A R7	1	1	Logical AND Register to Accumulator
0x60	JZ code	2	2/9	Jump if Accumulator is zero
0x61	AJMP code	2	9	Absolute Jump
0x62	XRL data A	2	1	Logical XOR Accumulator to direct data
0x63	XRL data const	3	1	Logical XOR immediate data to direct data
0x64	XRL const	2	1	Logical XOR immediate data to Accumulator
0x65	XRL A data	2	1	Logical XOR Direct Data to Accumulator
0x66	XRL A @R0	1	1	Logical XOR indirect data to Accumulator
0x67	XRL A @R1	1	1	Logical XOR indirect data to Accumulator
0x68	XRL A R0	1	1	Logical XOR Register to Accumulator
0x69	XRL A R1	1	1	Logical XOR Register to Accumulator
0x6A	XRL A R2	1	1	Logical XOR Register to Accumulator
0x6B	XRL A R3	1	1	Logical XOR Register to Accumulator
0x6C	XRL A R4	1	1	Logical XOR Register to Accumulator
0x6D	XRL A R5	1	1	Logical XOR Register to Accumulator
0x6E	XRL A R6	1	1	Logical XOR Register to Accumulator
0x6F	XRL A R7	1	1	Logical XOR Register to Accumulator
0x70	JNZ code	2	2/9	Jump if Accumulator is not zero
0x71	ACALL code	2	9	Absolute Subroutine Call
0x72	ORL C bit	2	1	Logical OR bit to carry
0x73	JMP @A+DPTR	1	9	Indirect Jump
0x74	MOV A const	2	1	Move immediate data into Accumulator
0x75	MOV data const	3	1	Move immediate data into direct data
0x76	MOV @R0 const	2	1	Move immediate data into indirect data
0x77	MOV @R1 const	2	1	Move immediate data into indirect data

Opcode	Mnemonic	Bytes	Cycle	Comment
0x78	MOV R0 const	2	1	Move immediate data into Register
0x79	MOV R1 const	2	1	Move immediate data into Register
0x7A	MOV R2 const	2	1	Move immediate data into Register
0x7B	MOV R3 const	2	1	Move immediate data into Register
0x7C	MOV R4 const	2	1	Move immediate data into Register
0x7D	MOV R5 const	2	1	Move immediate data into Register
0x7E	MOV R6 const	2	1	Move immediate data into Register
0x7F	MOV R7 const	2	1	Move immediate data into Register
0x80	SJMP code	2	1	Short Jump
0x81	AJMP code	2	9	Absolute Jump
0x82	ANL C bit	2	1	Logical AND bit to carry
0x83	MOVC @A+PC	1	6	Move code into Accumulator
0x84	DIV AB	1	9	Division
0x85	MOV data data	3	1	Move direct data into direct data
0x86	MOV data @R0	1	1	Move indirect data into direct data
0x87	MOV data @R1	1	1	Move indirect data into direct data
0x88	MOV data R0	2	1	Move register into direct data
0x89	MOV data R1	2	1	Move register into direct data
0x8A	MOV data R2	2	1	Move register into direct data
0x8B	MOV data R3	2	1	Move register into direct data
0x8C	MOV data R4	2	1	Move register into direct data
0x8D	MOV data R5	2	1	Move register into direct data
0x8E	MOV data R6	2	1	Move register into direct data
0x8F	MOV data R7	2	1	Move register into direct data
0x90	MOV DPTR const	3	1	Move immediate data into DPTR
0x91	ACALL code	2	9	Absolute Subroutine Call
0x92	MOV bit C	2	1	Move carry into bit
0x93	MOVC @A+DPTR	1	6	Move code into Accumulator
0x94	SUBB A const	2	1	Subtract immediate data and carry bit
0x95	SUBB A data	2	1	Subtract direct data and carry bit
0x96	SUBB A @R0	1	1	Subtract indirect data and carry bit
0x97	SUBB A @R1	1	1	Subtract indirect data and carry bit
0x98	SUBB A R0	1	1	Subtract register and carry bit
0x99	SUBB A R1	1	1	Subtract register and carry bit
0x9A	SUBB A R2	1	1	Subtract register and carry bit
0x9B	SUBB A R3	1	1	Subtract register and carry bit
0x9C	SUBB A R4	1	1	Subtract register and carry bit
0x9D	SUBB A R5	1	1	Subtract register and carry bit
0x9E	SUBB A R6	1	1	Subtract register and carry bit
0x9F	SUBB A R7	1	1	Subtract register and carry bit
0xA0	ORL C /bit	2	1	(C) = (C) or (not bit)
0xA1	AJMP code	2	9	Absolute Jump
0xA2	MOV C bit	2	1	Move bit into carry
0xA3	INC DPTR	1	1	DPTR Increment
0xA4	MUL AB	1	3	Multiplication
0xA5	Invalid Op	NA	NA	NA
0xA6	MOV @R0 data	2	1	Move direct data into indirect data
0xA7	MOV @R1 data	2	1	Move direct data into indirect data
0xA8	MOV R0 data	2	1	Move direct data into register
0xA9	MOV R1 data	2	1	Move direct data into register
0xAA	MOV R2 data	2	1	Move direct data into register
0xAB	MOV R3 data	2	1	Move direct data into register
0xAC	MOV R4 data	2	1	Move direct data into register
0xAD	MOV R5 data	2	1	Move direct data into register
0xAE	MOV R6 data	2	1	Move direct data into register
0xAF	MOV R7 data	2	1	Move direct data into register
0xB0	ANL C /bit	2	1	(C) = (C) and (not bit)
0xB1	ACALL code	2	9	Absolute Subroutine Call
0xB2	CPL bit	2	1	(bit) = (not bit)
0xB3	CPL C	1	1	(C) = (not C)
0xB4	CJNE const code	3	2/9	Subtract immediate data from Accumulator, modify carry bit, and jump if they are not equal
0xB5	CJNE data code	3	2/9	Subtract direct data from Accumulator, modify carry bit, and jump if they are not equal
0xB6	CJNE @R0 const code	3	2/9	Subtract immediate data from indirect data, modify carry bit, and jump if they are not equal
0xB7	CJNE @R1 const code	3	2/9	Subtract immediate data from indirect data, modify carry bit, and jump if they are not equal
0xB8	CJNE R0 const code	3	2/9	Subtract immediate data from register, modify carry bit, and jump if they are not equal

Opcode	Mnemonic	Bytes	Cycle	Comment
0xB9	CJNE R1 const code	3	2/9	Subtract immediate data from register, modify carry bit, and jump if they are not equal
0xBA	CJNE R2 const code	3	2/9	Subtract immediate data from register, modify carry bit, and jump if they are not equal
0xBB	CJNE R3 const code	3	2/9	Subtract immediate data from register, modify carry bit, and jump if they are not equal
0xBC	CJNE R4 const code	3	2/9	Subtract immediate data from register, modify carry bit, and jump if they are not equal
0xBD	CJNE R5 const code	3	2/9	Subtract immediate data from register, modify carry bit, and jump if they are not equal
0xBE	CJNE R6 const code	3	2/9	Subtract immediate data from register, modify carry bit, and jump if they are not equal
0xBF	CJNE R7 const code	3	2/9	Subtract immediate data from register, modify carry bit, and jump if they are not equal
0xC0	PUSH data	2	1	push direct data onto stack
0xC1	AJMP code	2	9	Absolute Jump
0xC2	CLR bit	2	1	Clear bit
0xC3	CLR C	1	1	Clear Carry Bit
0xC4	SWAP A	1	3	Swap higher 4 bits with the lower 4 bits in Accumulator
0xC5	XCH A data	2	1	Exchange between direct data and Accumulator
0xC6	XCH A @R0	1	1	Exchange between indirect data and Accumulator
0xC7	XCH A @R1	1	1	Exchange between indirect data and Accumulator
0xC8	XCH A R0	1	1	Exchange between register and Accumulator
0xC9	XCH A R1	1	1	Exchange between register and Accumulator
0xCA	XCH A R2	1	1	Exchange between register and Accumulator
0xCB	XCH A R3	1	1	Exchange between register and Accumulator
0xCC	XCH A R4	1	1	Exchange between register and Accumulator
0xCD	XCH A R5	1	1	Exchange between register and Accumulator
0xCE	XCH A R6	1	1	Exchange between register and Accumulator
0xCF	XCH A R7	1	1	Exchange between register and Accumulator
0xD0	POP data	2	1	Pop data from stack
0xD1	ACALL code	2	9	Absolute Subroutine Call
0xD2	SETB bit	2	1	Set bit to one
0xD3	SETB C	2	1	Set carry bit to one
0xD4	DA A	1	5	Decimal Adjust
0xD5	DJNZ data code	3	2/9	Decrement direct data, and jump if not zero
0xD6	XCHD A @R0	1	3	Exchange the lower 4 bits between indirect data and Accumulator
0xD7	XCHD A @R1	1	3	Exchange the lower 4 bits between indirect data and Accumulator
0xD8	DJNZ R0 data	2	2/9	Decrement register, and jump if not zero
0xD9	DJNZ R1 data	2	2/9	Decrement register, and jump if not zero
0xDA	DJNZ R2 data	2	2/9	Decrement register, and jump if not zero
0xDB	DJNZ R3 data	2	2/9	Decrement register, and jump if not zero
0xDC	DJNZ R4 data	2	2/9	Decrement register, and jump if not zero
0xDD	DJNZ R5 data	2	2/9	Decrement register, and jump if not zero
0xDE	DJNZ R6 data	2	2/9	Decrement register, and jump if not zero
0xDF	DJNZ R7 data	2	2/9	Decrement register, and jump if not zero
0xE0	MOVX A @DPTR	1	1	Move data in XRAM into Accumulator
0xE1	AJMP code	2	9	Absolute Jump
0xE2	MOVX A, @R0	1	1	Move data in XRAM into Accumulator
0xE3	MOVX A, @R1	1	1	Move data in XRAM into Accumulator
0xE4	CLR A	1	1	Clear Accumulator
0xE5	MOV A data	1	1	Move direct data into Accumulator
0xE6	MOV A @R0	1	1	Move indirect data into Accumulator
0xE7	MOV A @R1	1	1	Move indirect data into Accumulator
0xE8	MOV A R0	1	1	Move register into Accumulator
0xE9	MOV A R1	1	1	Move register into Accumulator
0xEA	MOV A R2	1	1	Move register into Accumulator
0xEB	MOV A R3	1	1	Move register into Accumulator
0xEC	MOV A R4	1	1	Move register into Accumulator
0xED	MOV A R5	1	1	Move register into Accumulator
0xEE	MOV A R6	1	1	Move register into Accumulator
0xEF	MOV A R7	1	1	Move register into Accumulator

Opcode	Mnemonic	Bytes	Cycle	Comment
0xF0	MOVX @DPTR A	1	1	Move Accumulator into XRAM
0xF1	ACALL code	2	9	Absolute Subroutine Call
0xF2	MOVX @R0 A	1	1	Move Accumulator into XRAM
0xF3	MOVX @R0 A	1	1	Move Accumulator into XRAM
0xF4	CPL A	1	1	Invert Accumulator
0xF5	MOV data A	2	1	Move Accumulator into direct data
0xF6	MOV @R0 A	1	1	Move Accumulator into indirect data
0xF7	MOV @R1 A	1	1	Move Accumulator into indirect data
0xF8	MOV R0 A	1	1	Move Accumulator into register
0xF9	MOV R1 A	1	1	Move Accumulator into register
0xFA	MOV R2 A	1	1	Move Accumulator into register
0xFB	MOV R3 A	1	1	Move Accumulator into register
0xFC	MOV R4 A	1	1	Move Accumulator into register
0xFD	MOV R5 A	1	1	Move Accumulator into register
0xFE	MOV R6 A	1	1	Move Accumulator into register
0xFF	MOV R7 A	1	1	Move Accumulator into register

Deliverable

This IP core is available in the form of:

- MCU core in RTL source code (System Verilog)

And the release package will include:

- Other peripheral IP cores in certain form
- OCD in source code
- Release notes
- ICD (Interface Control Document)
- Modelsim simulation library
- Testbench
- BIST Test Vectors
- Scripts for simulation or synthesis
- Debug Console in Python
- Library (source code) to support the majority of Arduino Language
- Sample code in C language (SDCC)