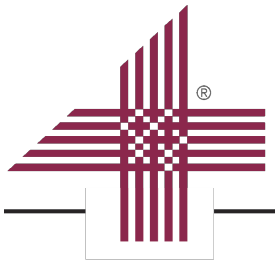
SwiftCore: SC20

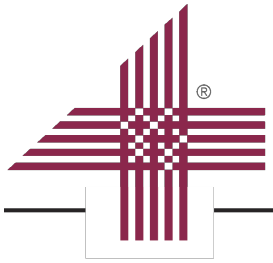
32-bit Soft CPU Core
for Systems-on-a-Chip (SoCs).



FORTH, Inc.

Agenda

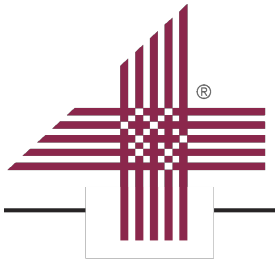
- Rationale
- Size and performance
- ISA (Instruction Set Architecture)
- SwiftX
- Use cases



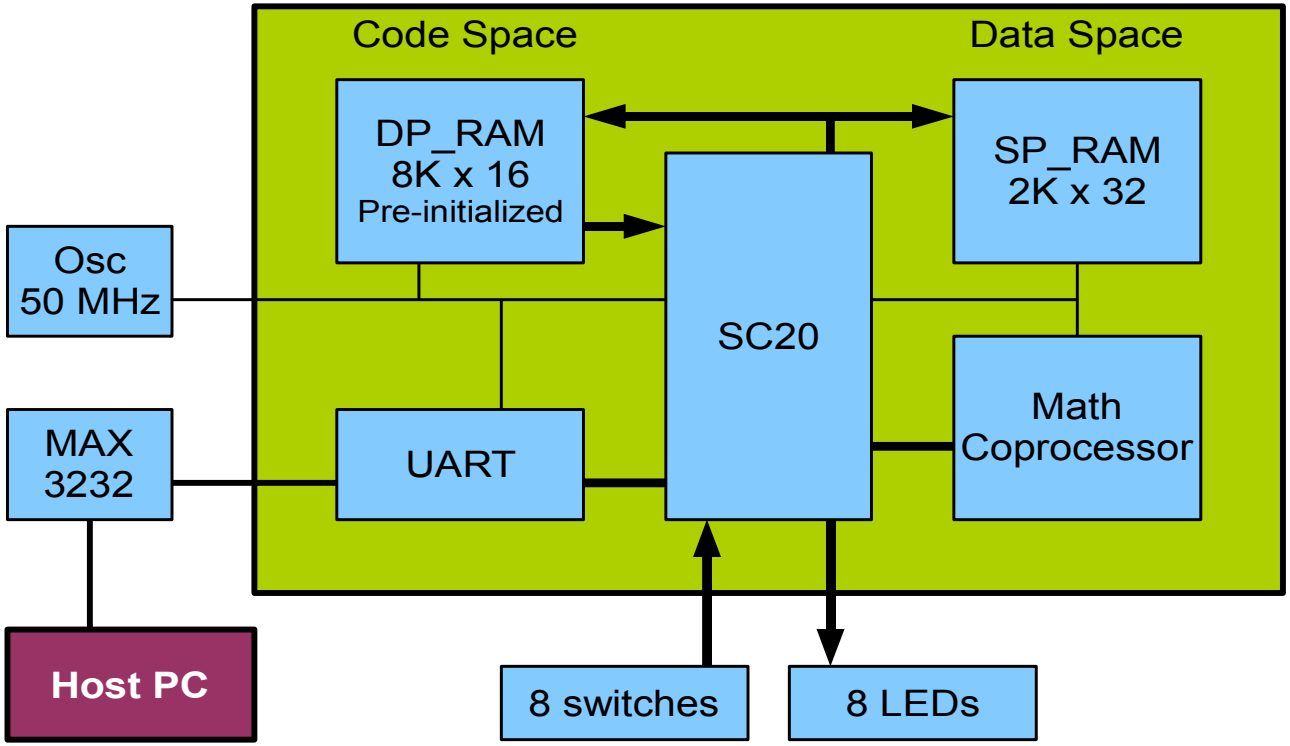
FORTH, Inc.

Rationale

- 40% of FPGA designs in 2010 have a built-in CPU (Gartner), just getting started.
- A Forth-friendly 32-bit CPU (generic RTL) built for modern FPGA and ASIC is future proof.
- On-chip Forth, in simulation and real silicon, is a great tool for SoC verification and validation.
- SwiftX – soft CPU combo leverages a mature and well documented tool flow and code base.



Test setup (small MCU)



FORTH, Inc.

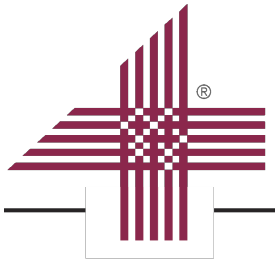
Logic Size (small MCU)

	FPGA	Elements	%	Speed	Special Features	Part Cost [1]	Logic Cost
45 nm	Spartan 6 xc6slx25	1982 LUTs	12	50 MHz		30.90	3.71
	Virtex 6 xc6vlx75T	1977 LUTs	4	150 MHz		764.00	30.56
	Virtex 6 xc6vlx75T	3088 LUTs	6	100 MHz	Math Coprocessor	764.00	45.84
	Cyclone IV EP4CE22E	3105 LEs	14	50 MHz	Shallow stacks	35.52	4.97
	Cyclone III EP3C25E	5227 LEs	21	50 MHz	Math Coprocessor	39.50	8.30
	Stratix III EP3SE50F	1939 ALUTs	7	125 MHz	Shallow stacks	761.00	53.27
	Stratix III EP3SE50F	2178 ALUTs	9	125 MHz		761.00	68.49
90 nm	XP2 LFXP2-30	2361 LUTs	12	65 MHz		59.08	7.09
	XP2 LFXP2-30	3096 LUTs	16	65 MHz	Math Coprocessor	59.08	9.45
	LFE2-20E-5QN208C	2790 LUTs	16	65 MHz		32.49	5.20

[1]: Part cost is low volume distributor webstore price as of Q4 2010

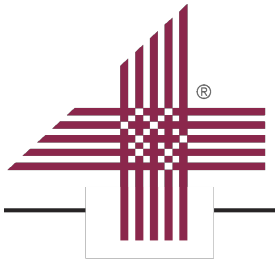
Performance Summary

- 50 to 150 MIPs, depending on the FPGA.
- Easy to add custom coprocessor hardware to make up for lower performance relative to a hard CPU.
- Stack-based architecture efficiently supports both Forth and C without heavy optimization.



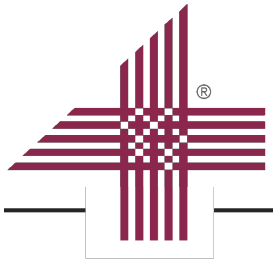
Hardware Goals

- Use single-port, synchronous R/W RAM/ROM (block RAM) for large memories.
- Use dual-port RAMs (LUT RAM) for stack caches and register arrays.
- Don't use large dual-port RAM for stacks.
- Wide muxes are cheap in ASIC, expensive in FPGA.
- Adders are cheap in FPGA, expensive in ASIC.
- SC20 is balanced for FPGA and ASIC use.

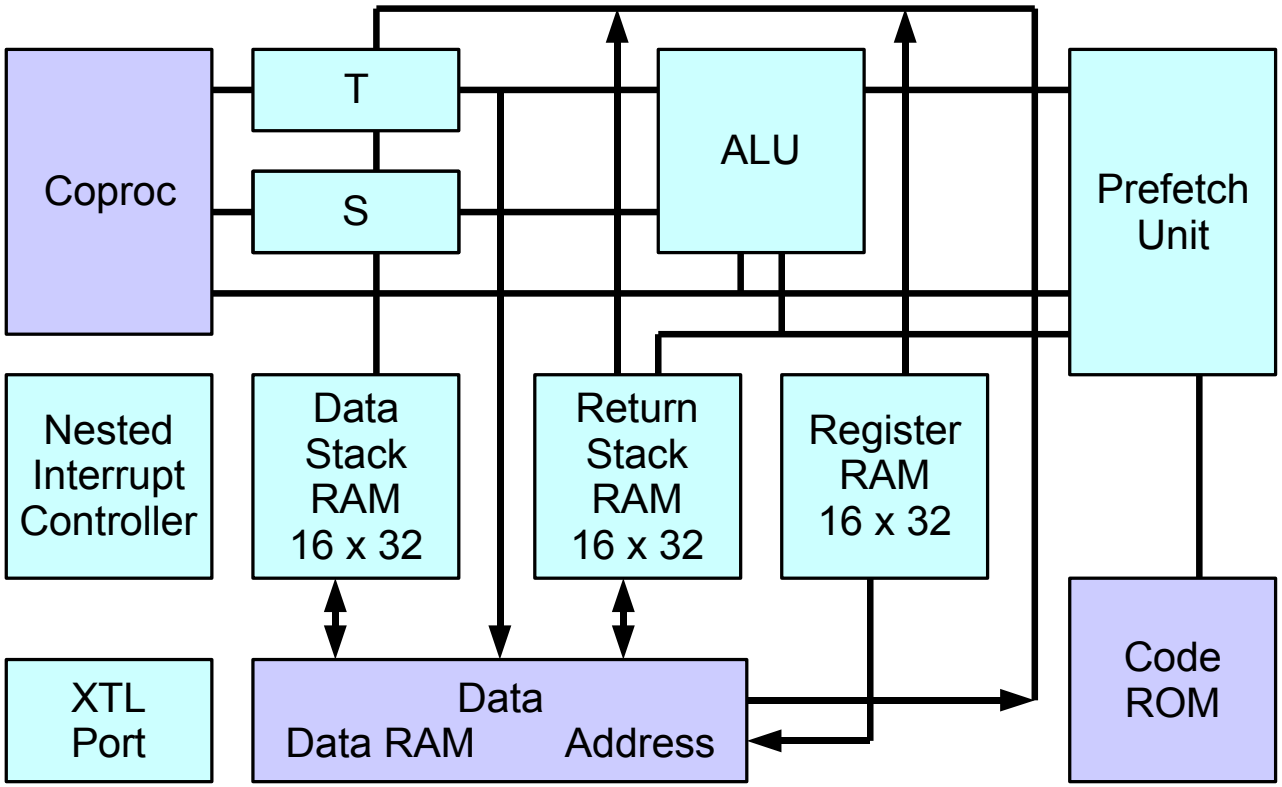


Software (ISA) Goals

- Mimic the PC environment: 32-bit, byte-addressed, little-endian, Von Neuman model.
- Use automatic stack spill and refill to allow unlimited (within RAM limits) stack depth.
- Optimized for 16-bit code memory but implementation can scale up or down in bus width.



Block Diagram



FORTH, Inc.

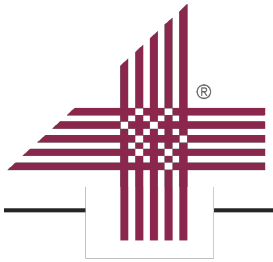
Registers

U	U	U	U
W	W	W	W
A	A	A	A
B	B	B	B
SP	U	U	U
RP	W	W	W
X	A	A	A
Y	B	B	B

U & W are general purpose regs
A, B, X, Y are also address regs

A, B, X, Y support indirect addressing
X, Y also support base+offset

ISRs may select any BANK of U,W,A,B



FORTH, Inc.

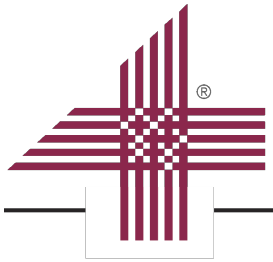
Opcode Map

BRA

ALU

LD/ST

+	0	20	40	60	80	A0	C0	E0
00	SEQ	SCS	SMI	SVS	STREG			
01	SHI	SGE	SGT	SAL	LDREG			
02	SNE	SCC	SPL	SVC	LIT			
03	SLS	SLT	SLE	SNV	COP			
04	TJMP	RET	PSHPC	MXM	LDIB			
05	SXTB	LSL	SXTH	LSR	LDIH			
06	COM	LSL4	NEG	ASR4	LDI			
07	ZXTB	LSLXT	ZXTH	ASR	LINK			
08	LDXB	LDXB+	LDAB	LDAB+	LDXIB			
09	LDXH	LDXH+	LDAH	LDAH+	LDXIH			
0A	LDX	LDX+	LDA	LDA+	LDXI			
0B	GETX	LDX-	GETA	LDA-	NEXT			
0C	LDYB	LDYB+	LDBB	LDBB+	LDYIB			
0D	LDYH	LDYH+	LDBH	LDBH+	LDYIH			
0E	LDY	LDY+	LDB	LDB+	LDYI			
0F	GETY	LDY-	GETB	LDB-	ZBRAN			
10	ADD	ADDC	SUB	SUBB	ADDI			
11	ORL	POPPSR	THRD	SETR	ORI			
12	XRL	GETRP	GETU	TDUP	XRI			
13	ANL	SETRP	SETU	PUSH	ANI			
14	DEC	INC	TROT	TSWAP	STIB			
15	FLUSH	PSHPSR	NOP	GETR	STIH			
16	TOVER	GETSP	GETW	POP	STI			
17	TDROP	SETSP	SETW	TNIP	SWI			
18	STXB	STXB+	STAB	STAB+	STXIB			
19	STXH	STXH+	STAH	STAH+	STXIH			
1A	STX	STX+	STA	STA+	STXI			
1B	SETX	STX-	SETA	STA-	BSR			
1C	STYB	STYB+	STBB	STBB+	STYIB			
1D	STYH	STYH+	STBH	STBH+	STYIH			
1E	STY	STY+	STB	STB+	STYI			
1F	SETY	STY-	SETB	STB-	BRA			



FORTH, Inc.

Instruction Format

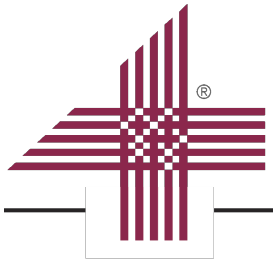
Instruction length is encoded in the opcode:

- An opcode with immediate data must reside at an even byte address.
- If a group has a second opcode, it is executed regardless of whether a branch was taken.

b7	b6:b5	Imm Bits
0	xx	0
1	00	8
1	01	16
1	10	24
1	11	32

Instruction memory is 16-bit, containing one of the following instruction groups:

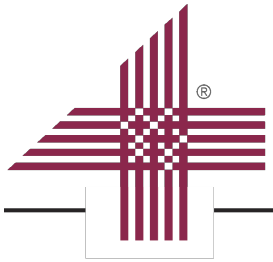
- Two 8-bit instructions
- One 16-bit instruction
- One 24-bit instruction and an 8-bit instruction
- One 32-bit instruction
- One 40-bit instruction and an 8-bit instruction



FORTH, Inc.

Interrupts

- Interrupts have a 2-bit priority level, allowing higher priority interrupts to nest into lower ones.
- Hardware saves status such as BANK, IMR, carry and overflow flags as it services the IRQ. It also selects the BANK (of U/W/A/B) used by the ISR.
- The ISR ends with a “POPPSR RET” instruction group. “RETI” macro.
- Pulse-triggered or Hi-triggered, expect synced inputs.
- User logic is responsible for converting falling-edge or rising-edge to trigger pulse.
- Each interrupt source has its own bank select, priority and enable bits.



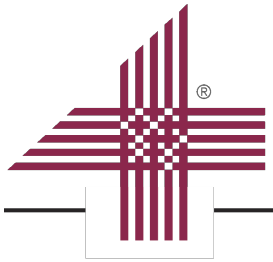
User I/O bits

- Up to 64 output bits.
- Up to 32 input bits.
- Read with LDREG.
- Write with STREG.

Example:

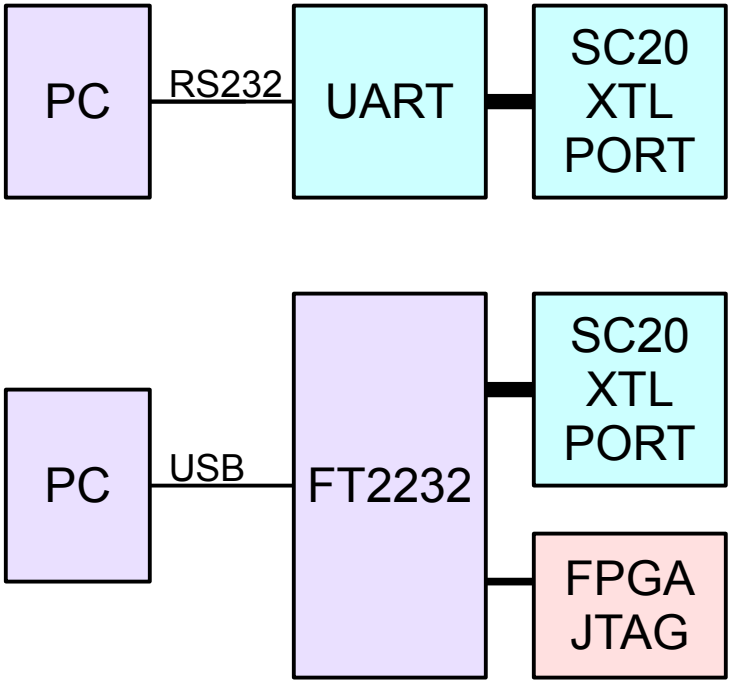
```
CODE SPI  ( c -- c' )
   7 LIT PUSH BEGIN
      TDUP  0 BOUT STREG  \ data out
      0 LIT 1 BOUT STREG  \ raise clk
      1 LIT 1 BOUT STREG  \ lower clk
      LSL  0 BIN LDREG    \ data in
      SUB
   NEXT RET
END-CODE
```

15 clocks per iteration
3.3 MHz @ 50M



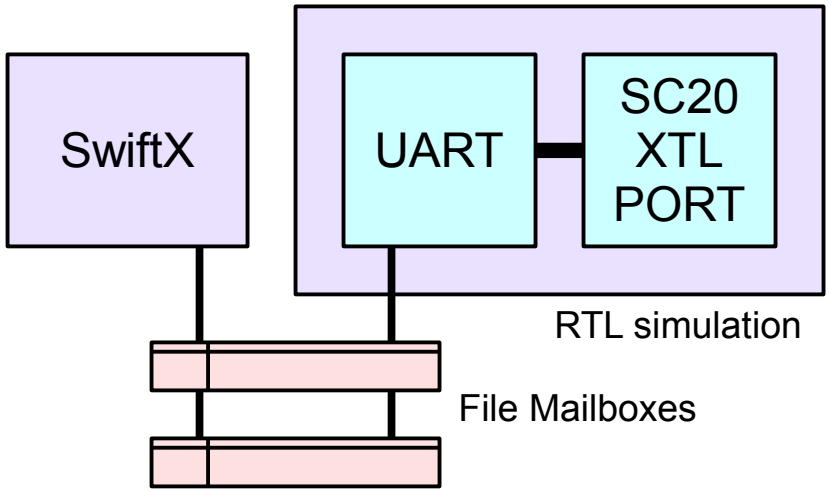
FORTH, Inc.

SwiftX



- SwiftX uses a XTL (cross-target link) to connect a thin-client debugger to a rich Windows or Linux based development environment.
- The SC20's XTL port connects to a UART, which SwiftX accesses via a COM port.
- XTL port is designed to connect to a FIFO, so bridging to a FTDI USB chip would be easy. FT245R supports transfer rate of 1M byte/sec. FT2232H has transfer rate of >10M byte/sec.

ModelSim XTL

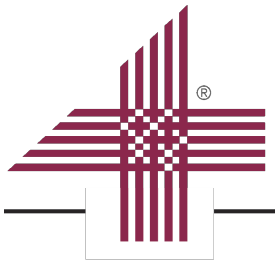


- SwiftX host may interface to a virtual UART running in an RTL simulation, implemented with the textio VHDL library.
- An RTL simulation of a SoC executes on the order of 1K instructions per second. Fast enough for interactive testing of a SoC model.
- The interface uses file mailboxes because of limitations in textio. The OS caches the files.

Ports

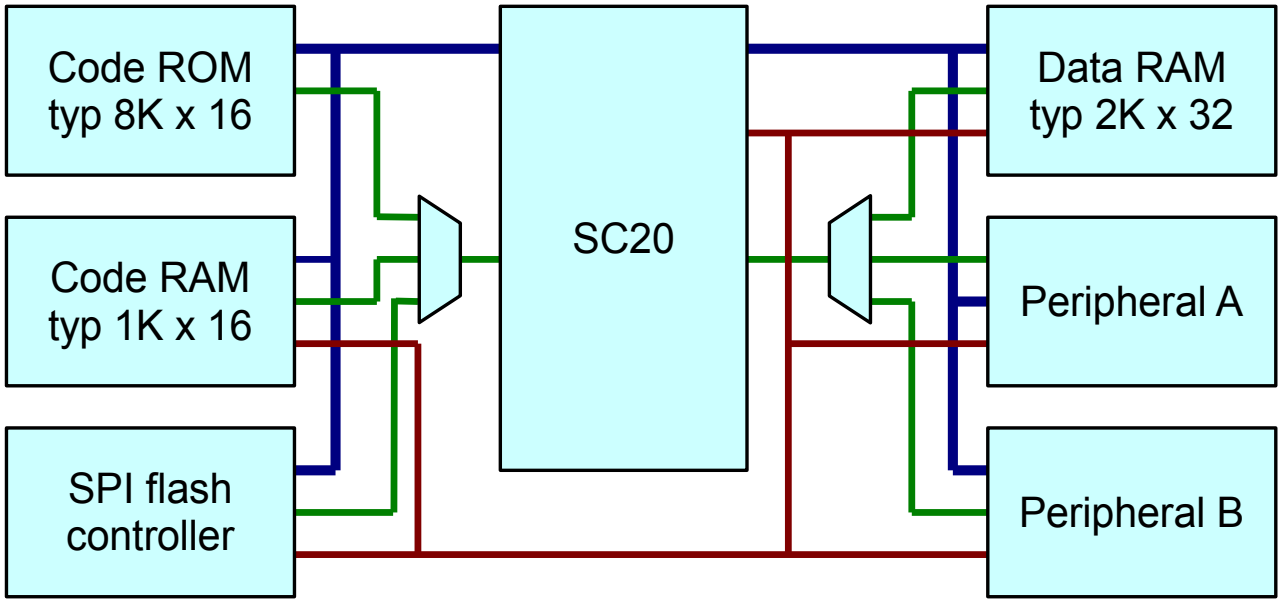
- Code memory
- Data memory

- XTL
- IRQs
- Input Bits
- Output Bits
- Wake & Sleep
- Clock & Reset



FORTH, Inc.

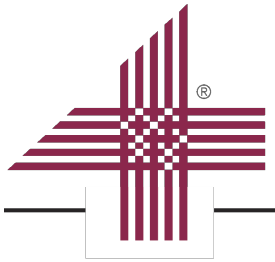
Off-core Memories



FORTH, Inc.

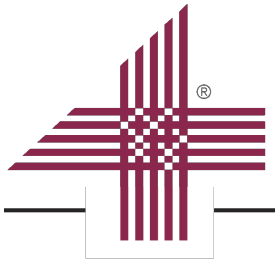
Code ROM

- Address range starts at 80000000h.
- Most synthesis tools will infer sync-read ROM from plain VHDL generated by SwiftX.
- The lower 256 words of ROM contain SWI vectors. User applications call kernel words using SWIs, insulating them from ROM changes.
- In an ASIC, ROM is cheap.



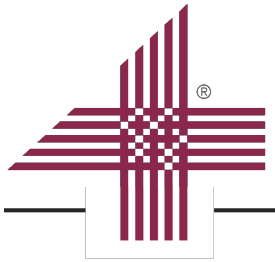
Code RAM

- Address range starts at 80010000h.
- Used to patch the kernel and/or ISRs.
- Lower 256 words is old or new SWI vectors, followed by ISR vectors.
- Application code may patch any SWI or ISR, and it may define time-critical app words in RAM.



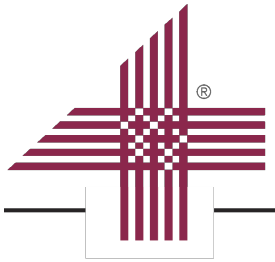
Off-chip SPI flash

- Address range starts at 80100000h.
- Low-end FPGAs boot from SPI flash. The same (usually 8-pin) part can hold application code.
- Modern SPI flash parts offer several Mbytes of storage for $1 – $2.
- The kernel and time-critical code is on-chip, so slow SPI speed is okay. But SPI isn't slow. Most parts allow >50 MHz clock and some have “quad rate” (4 bits/clk).

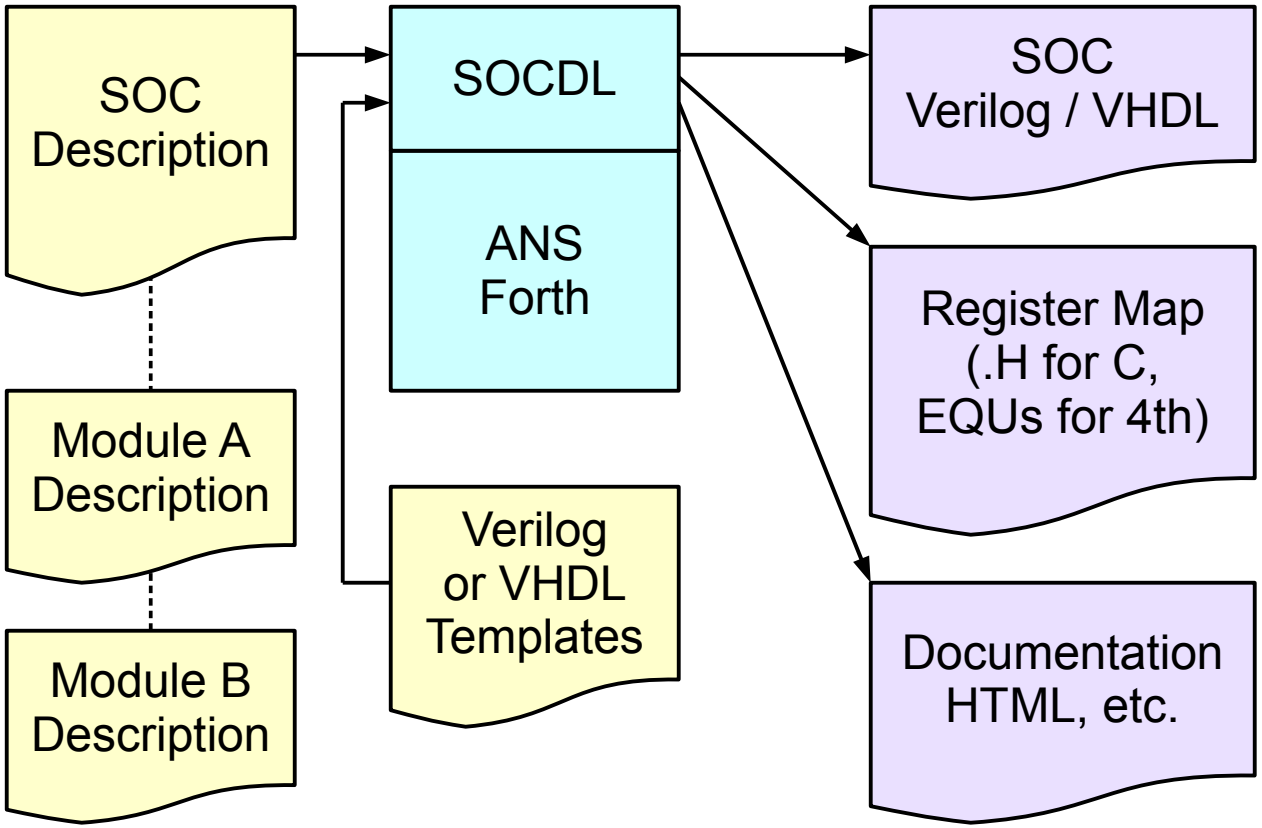


Peripherals

- Address range: F0000000 to FFFFFFFF allows the use of small negative numbers for addressing.
- RTL for a memory bus such as Wishbone or AMBA AXI should be machine-generated from a SoC specification.

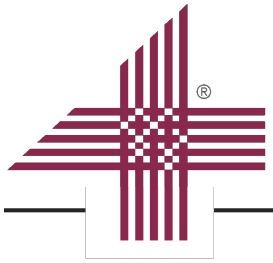


SoC Description Language



Questions

- Technical questions: ask Brad Eckert: brad@forth.com



FORTH, Inc.