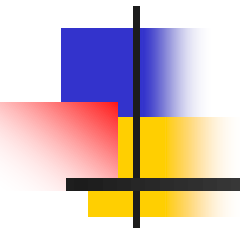
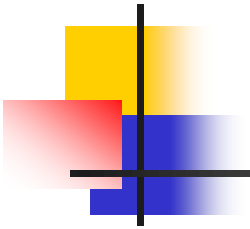


Simulate a Forth Virtual Machine

SVFIG

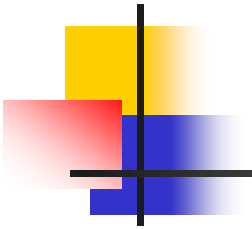
Chen-Hanson Ting

December 18, 2021



Summary

- Java Virtual Machine
- JVM Bytecode
- Forth Virtual Machine
- FVM Bytecode
- Synthesize FVM
- Simulate FVM
- Conclusion

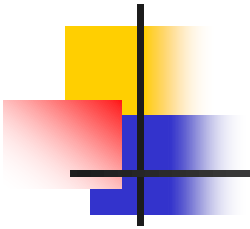


Java Virtual Machine

- JVM was specified by its bytecode.
- 202 bytecode was specified precisely.
- JVM architecture can be inferred from its bytecode.
- JVM bytecode map is as follows:

Java Bytecode

nop	aconst_null	iconst_m1	iconst_0	iconst_1	iconst_2	iconst_3	iconst_4	iconst_5	lconst_0	lconst_1	fconst_0	fconst_1	fconst_2	dconst_0	dconst_1
bipush	sipush	ldc	ldc_w	ldc2_w	iload	lload	fload	dload	aload	iload_0	iload_1	iload_2	iload_3	lload_0	lload_1
lload_2	lload_3	fload_0	fload_1	fload_2	fload_3	dload_0	dload_1	dload_2	dload_3	aload_0	aload_1	aload_2	aload_3	iaload	laload
faload	daload	aaload	baload	caload	saload	istore	lstore	fstore	dstore	astore	istore_0	istore_1	istore_2	istore_3	lstore_0
lstore_1	lstore_2	lstore_3	fstore_0	fstore_1	fstore_2	fstore_3	dstore_0	dstore_1	dstore_2	dstore_3	astore_0	astore_1	astore_2	astore_3	iastore
lastore	fastore	dastore	aastore	bastore	castore	sastore	pop	pop2	dup	dup_x1	dup_x2	dup2	dup2_x1	dup2_x1	swap
iadd	ladd	fadd	dadd	isub	lsub	fsub	dsub	imul	lmul	fmul	dmul	idiv	ldiv	fdiv	ddiv
irem	lrem	frem	drem	ineg	lneg	fneg	dneg	ishl	lshl	ishr	lshr	iushr	lushr	iand	land
ior	lor	ixor	lxor	iinc	i2l	i2f	i2d	l2i	l2f	l2d	f2i	f2l	f2d	d2i	d2l
d2f	i2b	i2c	i2s	lcmp	fcmpl	fcmpg	dcmpl	dcmpg	ifeq	ifne	iflt	ifge	ifgt	ifle	if_icmpeq
if_icmpne	if_icmplt	if_icmpge	if_icmpgt	if_icmple	if_acmpeq	if_acmpne	goto	jsr	ret	table	lookup	ireturn	lreturn	freturn	dreturn
areturn	return	getstatic	putstatic	getfield	putfield	invokevir	involespe	invokesta	invokeint	invokedyn	new	newarray	anewarray	arraylen	athrow
checkcast	instanceof	monitorenter	monitorexit	wide	multianew	ifnull	ifnonnull	goto_w	jsr_	breakpoint					
														impdep1	impdep2

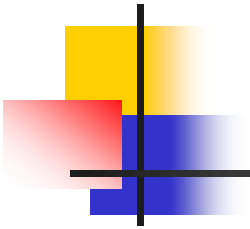


Forth Virtual Machine

- From JVM bytecode map, you can see a Forth Virtual Machine.
- If we restricted FVM to 32-bit integers, we would need only about 40 bytecodes.
- We need 4 more bytecodes to deal with an explicit return stack.

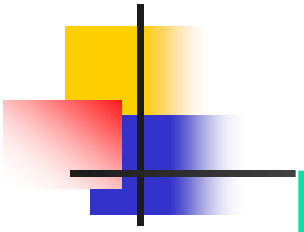
FVM Bytecode

nop	aconst_null	iconst_m1	iconst_0	iconst_1	iconst_2	iconst_3	iconst_4	iconst_5							
bipush	sipush														
														iaload	
			baload		saload										
															iastore
				bastore		sastore	pop	pop2	dup	dup_x1	dup_x2	dup2	dup2_x1	dup2_x1	swap
iadd				isub				imul				idiv			
irem				ineg				ishl		ishr	iushr			iand	
ior		ixor		iinc											
									ifeq						if_icmpeq
	if_icmplt		if_icmpgt				goto								
	return					invoke									
										donxt	ldi	popr	pushr	dupr	ext
get	put														



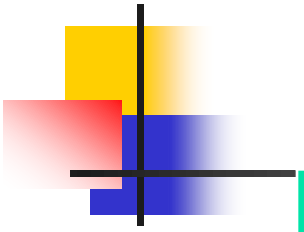
Map FVM to JVM

- ALU and logic
- Data stack
- Memory
- Branching
- Return stack



ALU

+	iadd
-	isub
*	imul
/	idiv
mod	irem
negate	ineg



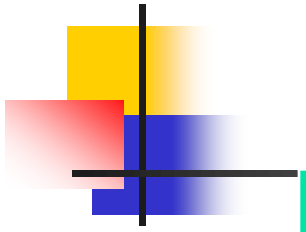
Logic

and	iand
or	ior
xor	ixor
lshift	ishl
rshift	ishr

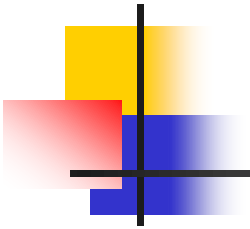
Data Stack

dup	dup
drop	pop
swap	swap
over	dup2,pop
2dup	dup2
2drop	pop2

Memory



@	iaload
!	iastore
c@	baload
c!	bastore
w@	saload
c!	sastore

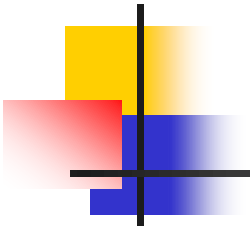


Branching

branch	Goto
0branch	ifeq
call	invokevirtual
ret	return
exit	return

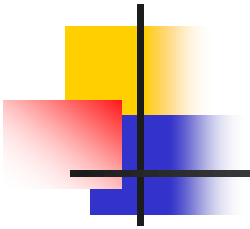
Return Stack

donxt	
>r	(pushr)
r>	(popr)
r@	iload_0



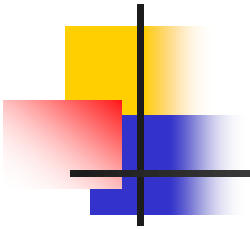
Synthesize FVM

- I have a free Quartus Lite license from Altera (acquired by Intel).
- It is not compatible with the NIOS II Kit I have from an earlier NASA project.
- I used its synthesizer and simulator to build and prove my FVM.



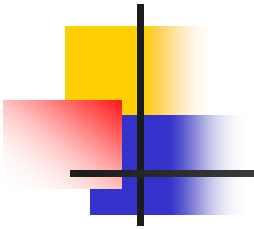
Synthesize FVM

- I have the advantage of doing an eP32 FVM in VHDL.
- eP32 had 5-bit instructions to be converted to bytecode.
- eP32 also had a Forth outer interpreter based on eForth, which cried out to be modernized.



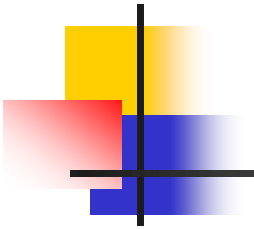
Features of FVM

- A dictionary, an input buffer, and an output buffer are instantiated.
- Input buffer stores Forth test code interpreted by FVM.
- Output buffer stores characters generated by FVM for verification.



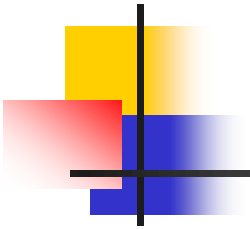
Synthesize FVM

- With insights gained from my OOP Forth, I rewrote the eForth outer interpreter for a bytecode FVM.
- The outer interpreter was fully debugged with a F# metacompiler.
- It serves well as a testbench in simulating FVM.



Synthesize FVM

- Synthesized on Quartus II IDE from Altera/Intel.
- FVM Core runs at 100 MHz
- With Multiply/Divide/Shifter library modules, it runs at 20 MHz.

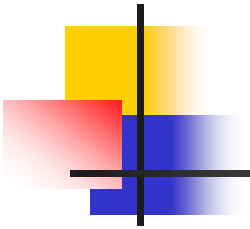


Synthesize FVM

- Forth outer interpreter was assembled, compiled, and simulated on F#.
- The dictionary is 3,424 bytes in size.
- Code fields have executable bytecode.

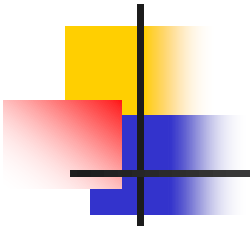
Synthesize FVM

Functions	ALUT
Core	3,936
Multiply/divide	5,478
Barrel shifter	5,786



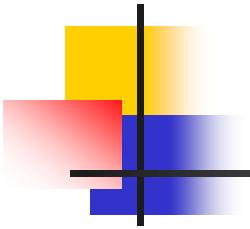
Simulate FVM

- A dictionary, an input buffer, and an output buffer are instantiated in main memory.
- Input buffer stores Forth test code interpreted by FVM.
- Output buffer stores characters generated by FVM for verification.



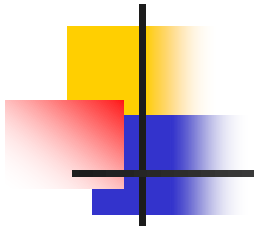
Simulate FVM

- The outer interpreter runs at master clock rate and greatly eases simulation and verification.
- You cannot design a better testbench, which is the most difficult task in implementing a CPU in FPGA.

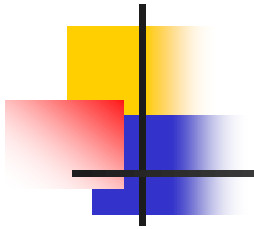


Conclusion

- JVM is a very good hardware specification of a bytecode CPU.
- FVM maps well to an integer JVM.
- FVM was synthesized in Quartus II IDE for a Stratix II FPGA device.
- Simulation continues.



Questions?



Thank you.