From RISC-V architecture to Layout

Introductory blog – Let’s rock and roll.

Kunal Ghosh

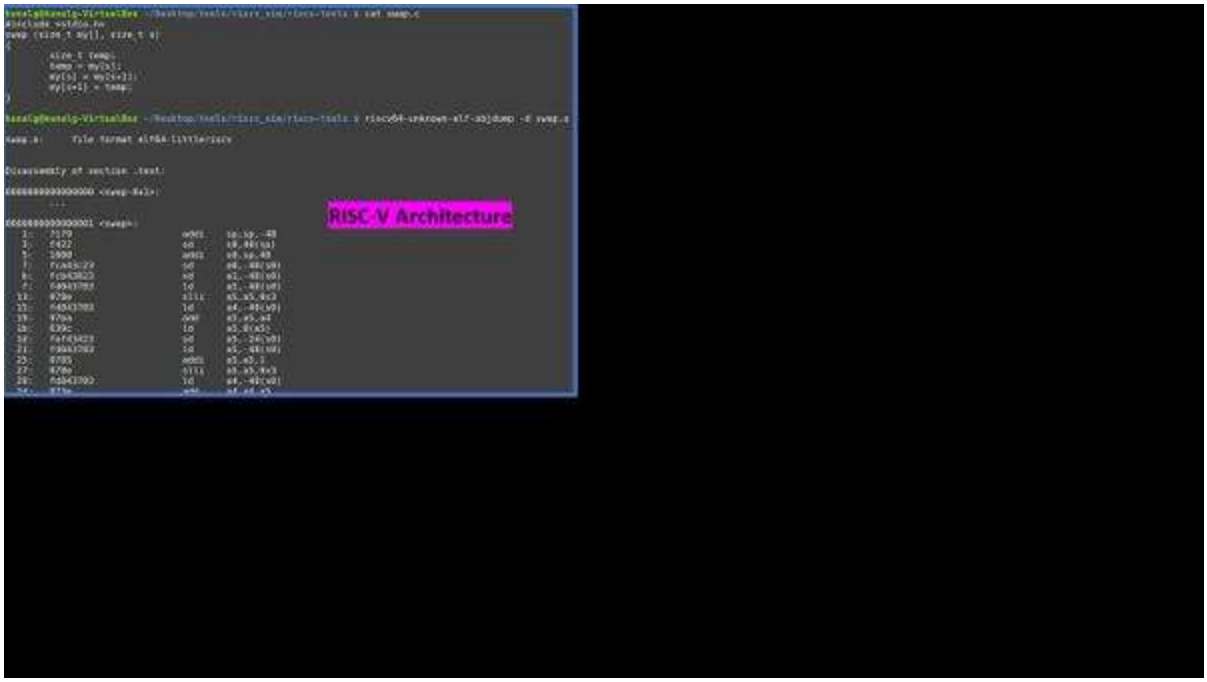
Hi

Ever thought what happens when you open an application program like Microsoft Word or Excel or a calculator on your laptop or mobile? These are application programs written in high-level programming languages, which are, eventually converted to binary machine language (in terms of '1's and '0's). What's interesting here is the software and hardware interface.

A high-level program, like swap.c as shown below is first converted to an assembly language program (RISC-V in below example) using compiler. This assembly language is converted to binary machine language program using an assembler. This level of abstraction of your application using high-level programming languages like C, C++, Java or Visual Basic, proves to be a great idea to improve design

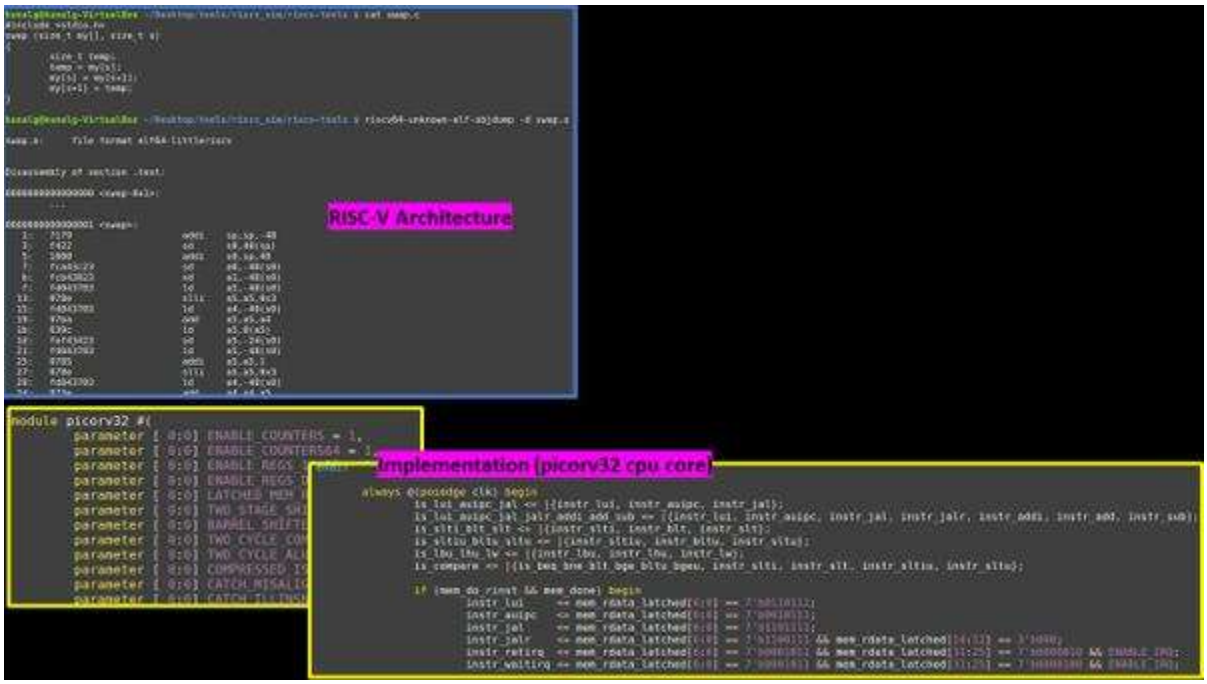
Here's the flow:

Application software- System Software-Hardware

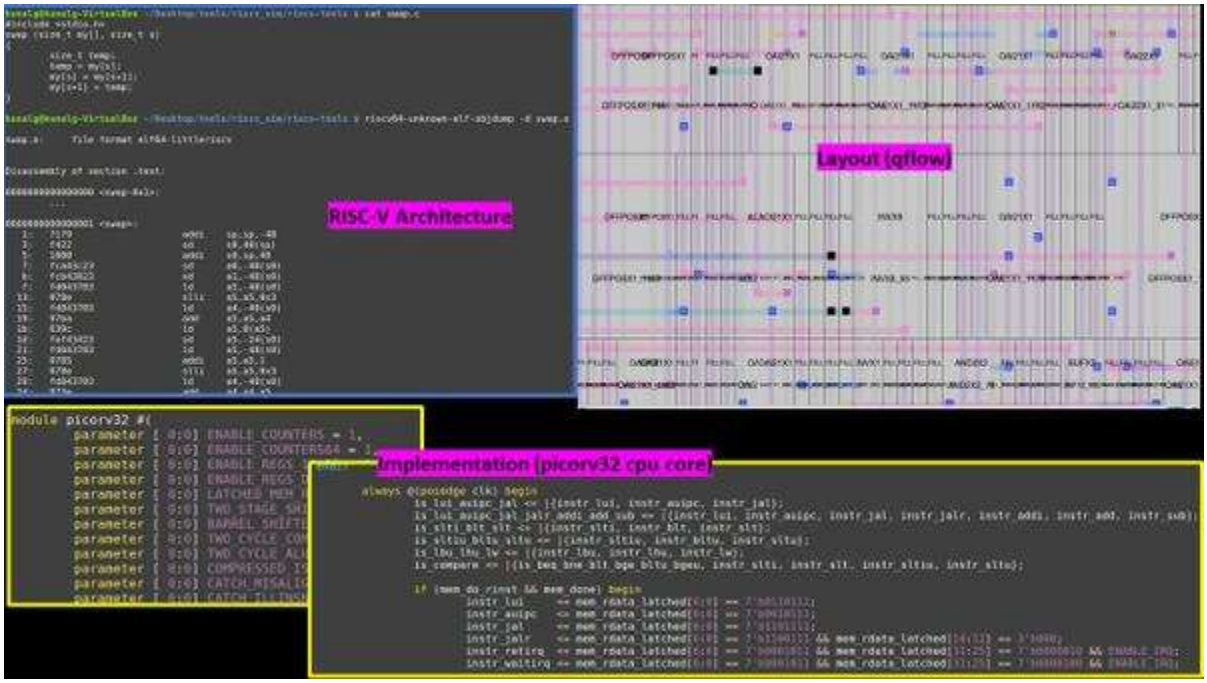


One important abstraction, as shown above, is the interface between lowest level software (see instructions like addi, sd, slli, etc. in above image) and hardware (See below). This lowest level software is therefore called as instruction set architecture (ISA) or simply architecture.

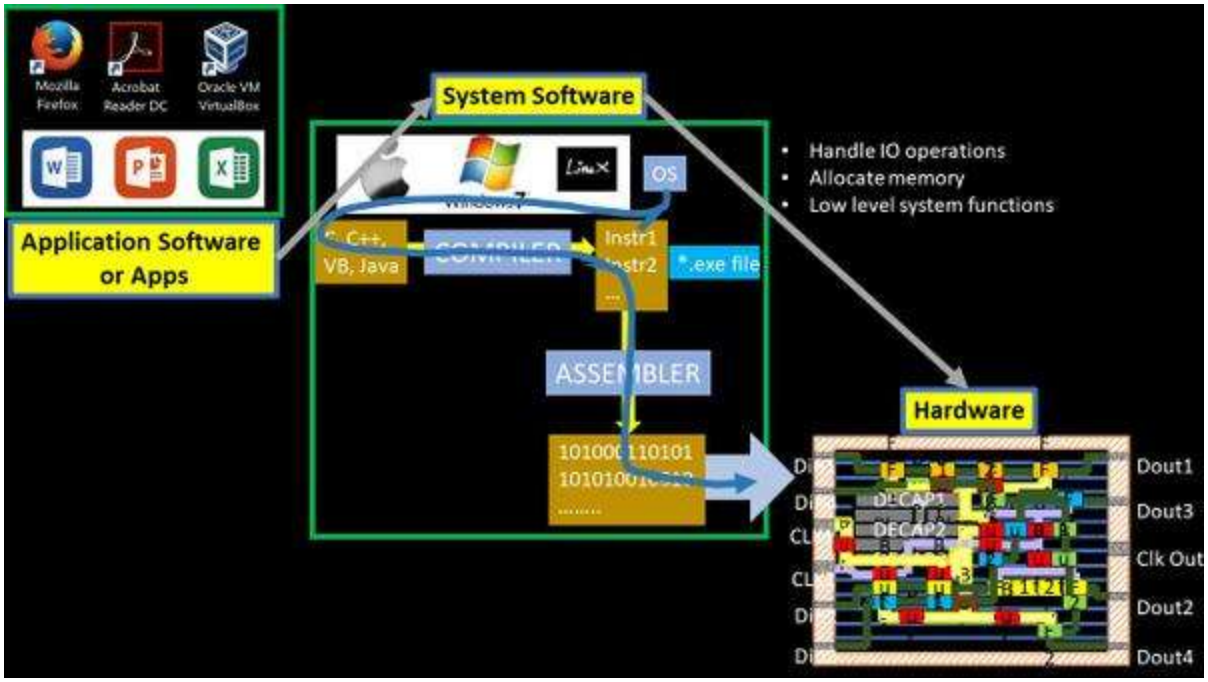
'Picorv32' is a cpu core that implements RISC-V architecture (which will be more evident as we continue with the upcoming course). Implementation is a hardware, which follows architecture abstraction. See RTL snippet in below image –



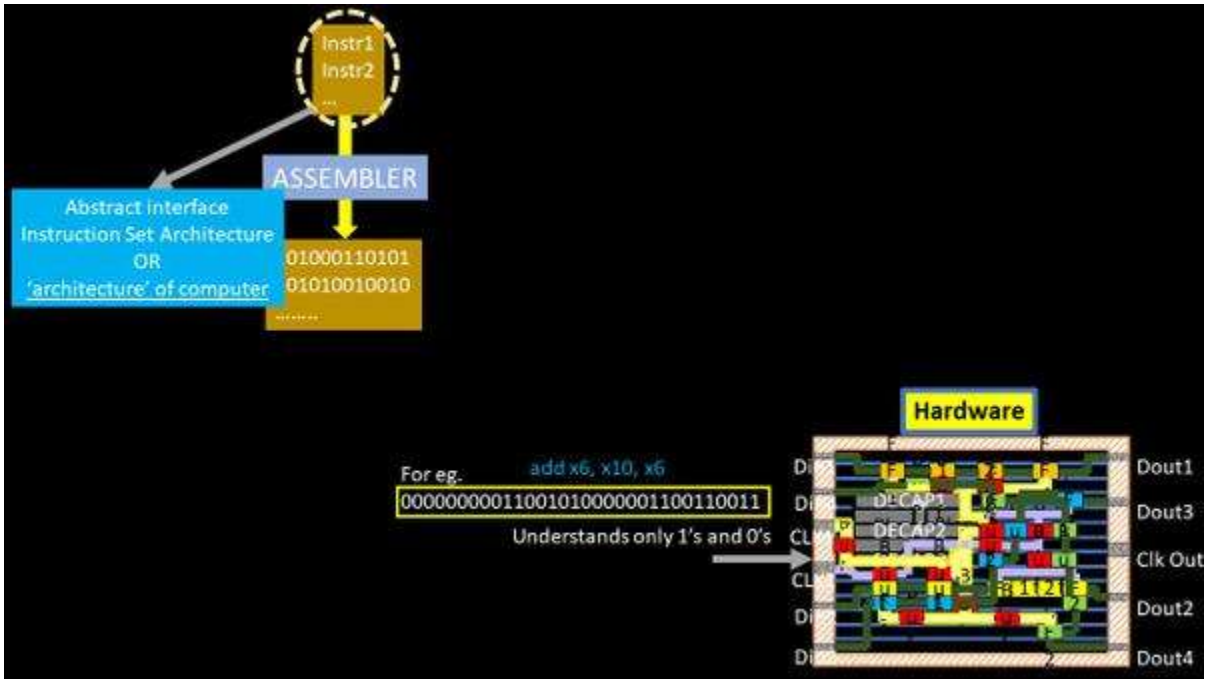
And finally, once we synthesize/pnr picorv32 using qflow (an opensource rtl2gds flow) and analyze using STA tool ‘opentimer’, below is the layout what you can see in MAGIC – Now this layout is an actual hardware implementation of a RISC-V architecture, you can call it a ‘processor’ which sits on your mobile phone and runs all applications on your phone, like facebook, linkedin and so on



Think of any application program or app (as it’s called today). It has millions of lines of codes written in some high-level language. But at the end of the day, it needs to be run on a hardware or chip which is present inside your phones or laptops. So who does that translation? – a layer system software’s – let’s focus on 2 of them which we need – OS (operating system) and compiler. Below image shows the bridge – system software – placed in between application programs and hardware:



If you observe closely, hardware understands only 1's and 0's and, believe it or not, there was an era, when first programmers did communicate with computes using binary numbers. That's tedious, right? Very quickly, some notations like add, sub, and many more were invented, and they replaced binary numbers shown as below: Here the combination of binary numbers instruct the hardware to add numbers in registers x10 and x6, and place the result in x6.



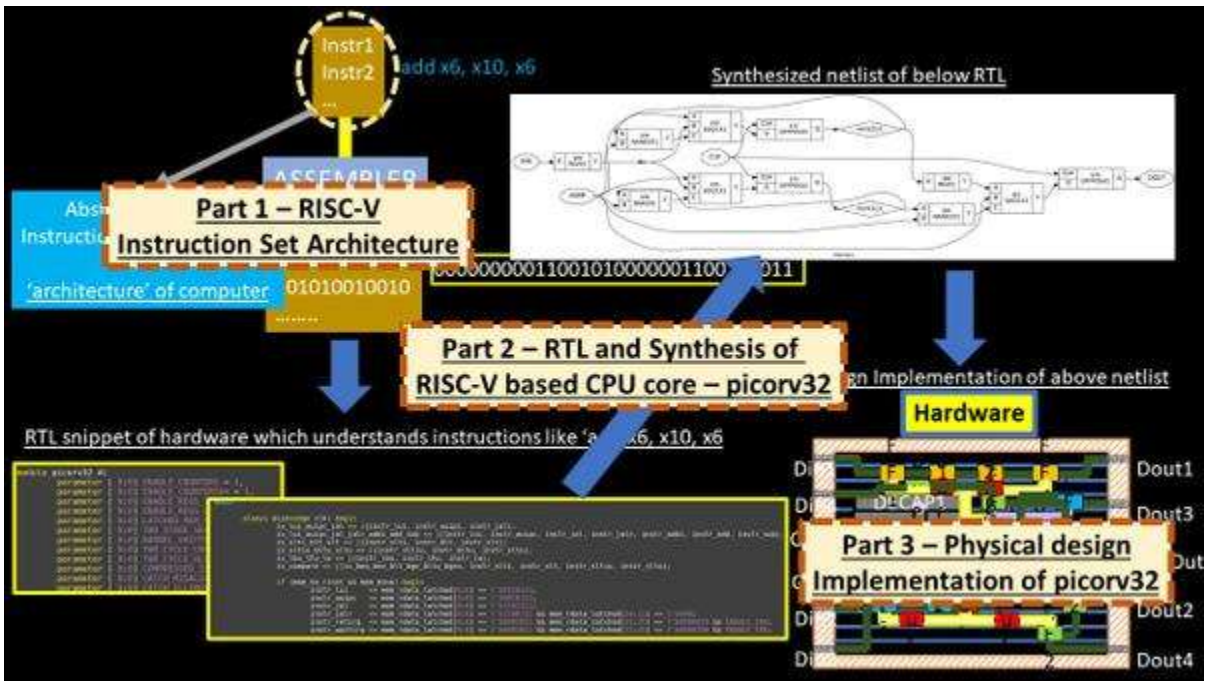
Just like this, there are hundreds of instruction sets being architected and the entire set of instructions are proprietary, like, ARM architecture for mobiles, x86 architecture for

servers, and many more. Similarly, RISC-V is a new set of instructions, that is developed in Computer Science division of EECS department at University of California, Berkeley and now, it is all set to become standard open architecture for industries

Going a level below, once you define an architecture, you would need a CPU core (to being with, an RTL) that implements and understands RISC-V instructions. In simple terms, the hardware should process binary machine language of a RISC-V command like 'add' and store results in some register which will be further processed by an IO.

The above RTL then needs to be run through a PNR tool chain to produce the layout, ready for fabrication. The below image covers it all:

Hence, we divide our set of courses in 3 parts as shown below:



Isn't this interesting? Yes, it is. "***<u>You are building a processor from scratch</u>***"
So here I announce about my 3 new courses

1. Introduction to risc-v instruction set architecture – Thanks to **SiFive** for helping me out on this
2. RTL implementation and synthesis of risc-v – Thanks to **Clifford Wolf**
3. RTL2GDS of picorv32 – Thanks to **Tim Edwards** from opencircuitdesign.com

All you need to do is 'Stay Tuned' to below site for course announcement. It will be a journey of lifetime, you would never forget
I will keep you posted with the links. Till then happy learning…