MicroCore's top priority is simplicity and understandability. MicroCore is rooted in the Forth language but it is not confined to execute Forth programs – it is a pretty good general purpose processor and the addition of indexed addressing into the return stack allows easy compilation of C programs that execute efficiently.

Yet its design approach has been different from most other processor cores: Its assembler was first and it realises about 25 Forth primitives. Whereas most other processors attempt to give you the utmost in instruction diversity from a minimum of hardware, you will find some very specialised instructions of high semantic content in MicroCore, because 30 years of Forth experience have proven that these are useful primitives. Nevertheless, each instruction executes in one clock cycle.

MicroCore is not the only architecture that uses Forth as its assembler. I took Chuck Moores NC4000 as a model, enhanced it to become the FRP1600, which never made it beyond second silicon and its vectored interrupt bug. A fresh approach was the realisation of the "Fieldbus Processor" IX1 that still sells in industrial automation. It introduced the Harvard Architecture to Forth machines. Eventually, the MicroCore architecture resulted from discussions with Christophe Lavarenne about the Transputer architecture and it utilises two inventions of the Transputer, namely: Concatenation of "nibbels" to form higher precision literals followed by an instruction that consumes it, which is the enabling technology making MicroCore's object code independent from the data path width. Secondly the TRAP signal and instruction as a hardware mechanism to deal with resource scheduling, which is the enabling technology for efficient multi-tasking.

MicroCore attempts to be an optimum in terms of hardware complexity versus instruction semantics. There are other architectures that are simpler at the expense of instruction semantics. But then you have to execute more of those simpler instructions to get a certain job done, which leads to higher power consumption because, after all, fetching instructions from program memory usually is the major source of power consumption in microprocessor systems. Therefore, MicroCore is an attempt to do those things in hardware, which are best done in hardware, leaving those operations to software, which are done in software more efficiently and with lower complexity. To give you a hint of what I mean: Compare peripheral microprocessor chips that have been designed by Intel and Motorola engineers: Intel repeatedly has tried to solve software problems in hardware, whereas Motorola usually got it right.

Due to Forth's extensibility - i.e. the openness of its unconventional compiler for user modifications and extensions - the Forth standardisation process is a slow consensus building process in a large community compared to the compiler writer specialists' circles of most other languages. Forth words could be thought of as the micro cells of a software system that has been realised in order to solve a specific problem. In much the same way as MicroCore's MicroCells should be used to realise a more complex, specific piece of hardware. Simplicity is an indication of proper factoring and understandability is the condition for wide acceptance. Proper factoring is a matter of experience and a crave for aesthetic solutions. E.g. the "proper" building blocks for a UART may be uart_clk, uart_tx, and uart_rx. Only time and experience can tell.

In releasing MicroCore to the public, I hope that it will become a catalyst to spawn additional peripheral functions, or MicroCells, placed under the same license conditions. As the Original Developer of MicroCore I reserve the right to certify conformance of Derived Work with the Original Code. This is my only business interest in MicroCore besides using it myself in embedded systems development. In that respect, the MicroCore licensing terms are differnt from GPL or other "Open Source" licenses, because MicroCore deals with hardware that runs application code rather than with application software or tools to develop application software.

As long as I can not support MicroCore as a product I will not be offering a "Public" MicroCore license. Instead, the "MicroCore Exploratory License" will allow individuals, universities, and research institutes to experiment with it. I will give a "MicroCore Unilateral License" to every company that is backed by an entity using MicroCore under the exploratory license free of charge as long as they don't bother me.

WWW.MICROCORE.ORG has been set up for community building.