A Short Introduction to the SiMulPro® Core

By
Earle Jennings, CTO
QSNM, LLC, Santa Fe NM and
QSigma, Inc, Denton, Texas
+1(510)292-8328
ewj@ix.netcom.com

All information contained herein is taken from documents pending before at least the US Patent and Trademark Office

Imagine your computers

- ★ Are 100% invulnerable to cyberattack (including AI)
- ★ Use 5% (or less) energy
- ★ Can still run everyone's legacy software
- ★ Can scale from edge devices to supercomputers

This is our vision at QSigma, Inc
And we are ready to change the world!

Introducing the SiMulPro Core

The Simultaneous Multi-Processor (SiMulPro) core is a non-von Neuman computer architecture replacing general purpose micro-processors

The SimulPro core *eliminates* the cyber-security vulnerabilities of today's micro-processors

SiMulPro core implementations, comparable to a 64 bit micro-processors operate at 2% - 5% of the power, in part by removing the need for energy wasteful caches, superscalar interpreters, and out of order execution mechanisms

The SiMulPro core is a software defined entity that is compiled from loops, functions and programs into multiple simultaneous processes in hardware. Truly general purpose, it scales from wrist watches and medical devices to supercomputers

Application compatiblity insures legacy software support with minimal compile time changes

QSigma has 13 issued patents, 4 published papers, and a simulation of a SiMulPro core

How Did We Do This?

Earle Jennings and his team worked for over 20 years incorporating and extending hardware software co-design to include:

- Semiconductor design, verification, and test
- Software application development
- Operating systems and compiler technologies
- Applied mathematics, including systems analysis, numerical analysis, formal methods, graph theory, and the semantics of computer languages

These disciplines are consistently applied at every level from the standard cells of target logic, to application support, and the entire system design with it's requirements

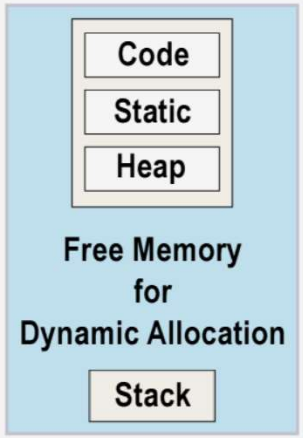
- By taking into account all of the above perspectives, a set of computer languages were selected and defined as the implementation language set.
- This allows semantically equivalent models to be built and tested supporting simultaneous, nearly independent projects for circuit development, compiler development, application tool development, and operating system implementation.
- Today these projects are often stalled by the circuit development project.

No one else has redesigned computers from scratch. We have no competition.

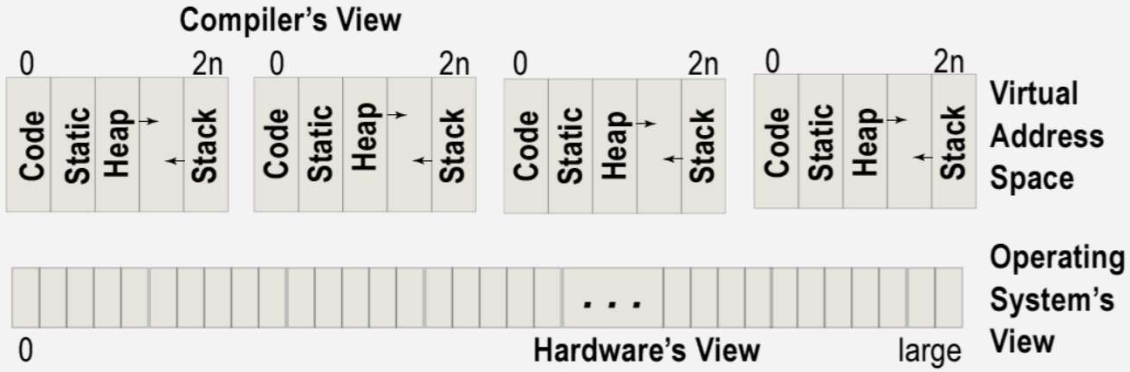
An Example of What Our Methodology Reveals

With current compilation methodologies there is no effective way to know whether a data pointer in a program is pointing to a legitimate address or not. This leads to vulnerabilities in the function return stack and the buffer overflow, among others.

This drawing shows a typical data memory layout as show in *Engineering a Compiler* 3rd ed. by Cooper and Torczon, p 254, 2023



This is called a virtual address space layout, which is divided into four categories of storage shown as code, static, heap and stack



"The heap and the stack grow toward each other...From the compiler's perspective, this virtual address space is the whole picture. However, modern computer systems typically execute multiple programs in an interleaved fashion. The operating system maps multiple virtual address spaces into a single physical address space supported by the processor." pp 254-255

Today, there is no efficient way to know at the hardware level, whether tweaking the operating system environment, a minor typo in the source code of a program, or function, may trash this intricate interleaving.

Cybersecurity adversaries can, and do, exploit this.

Energy Savings Estimate for an Example General Purpose SimulPro Core

A module instance of the core block diagram (shown on the Block Diagram slide), with the following parameters (shown on the Overview of Stripes slide):

- Data Access Unit Length = 8 bytes
- Up to 32 data access requests for both write and read issue from the 4 stripes of Data Processor 1 as 32 read and 32 write access requests from Coordinator Out to Auto Pipe 2
- Pipe clock frequency of 100 Million Hertz (MHz)

The maximum performance for this core module instance:

- FP80 operations (6.4 Gflops)
- FP64 operations (12.8 Gflops)
- FP32 operations (25.6 Gflops)
- FP16 operations (51.2 Gflops)
- FP8 operations (102.4 Gflops)

Int64 operations (12.8 Gops)
Int32 operations (25.6 Gops)
Int16 operations (51.2 Gops)
Int8 operations (102.4 Gops)

Point of Comparison:

Today, a typical 64 bit microprocessor has a data access unit length of 8 bytes, and typically runs between 2 to 4 GHz clock frequency

The SimulPro core, operating at 100 MHz, is estimated to operate at about 2% to 5% of the energy of the 64 bit microprocessor, assuming comparable performance, die size (area) and sheet capacitance.
This 2% to 5% is roughly the ratio of 100MHz / 2 to 4 GHz

Cybersecurity Adversary Defined

An Adversary is *anything* that has the ability to:

-Execute and control the input of user programs in an operating system

-Gain operating system privileges over time without immediate detection
  -security features alterable by system privileges are vulnerable

-Alter compiled code at the assembly language level

-Alter manufacturing processes and inject back doors into hardware

-Configure data memory device that installs malware altering task and/or program information when connected

-Send data messages that install malware altering task and/or program information

Cybersecurity Summary of SiMulPro Core

Cybersecurity attacks that cannot happen by design

- Attacks altering instructions in a program
- Attacks altering function return stack
- Attacks using data leakage from leftover state after a program runs

Detected and stopped using new circuits

- Read before write attacks

Detected and stopped using new circuits and compiler stage tools

- Attacks using buffer overflow
- Attacks using illegal pointer references
- Dynamic allocation related attacks
- Attacks writing to Read Only Memory
  - Object oriented specific attacks

Prevented in design process using the SiMulPro core's operating features

- Generation of back doors in chips

Examples of How the SiMulPro Core Defeats Attacks

Attacks that Alter Instructions in a Program

No single address space in the core holds all instructions

- Each instruction memory of each instructed resource is in a separate address space
- Each instruction memory is contained in a separate module, which can only be accessed within that module

No program, whether compiled or assembled, running on this core, can read or write any instruction memories

- Executing a program does not require the program being able to read or write it's instruction memory
- Only hardware running the program needs to read these instruction memories, not the program being executed

Buffer Overflow Attacks

The SiMulPro core requires the following

- Restructuring the compiler code generation to create contiguous memory domains from the compiler, operating system, and hardware perspectives
- These contiguous memory domains support fast, cheap runtime testing of buffer boundaries and program data memory domains

The runtime testing detects buffer overflow attacks, and illegal program data memory access

- Once detected. a hardware fault is injected in the Task Wave Front (TWF)
- Once the Task Wave Front with the injected hardware fault is received by the Operating System Execution Module (OSEM), it can terminate running the program by altering the TWF that is driving the SiMulPro core

Security Features of the SiMulPro core are inherent and cannot be turned off

AI Can Be 100% Invulnerable to Cybersecurity Attacks

Replace all CPUs with SiMulPro cores, including all CPUs in GPUs and in TPUs, making them all invulnerable to attack.

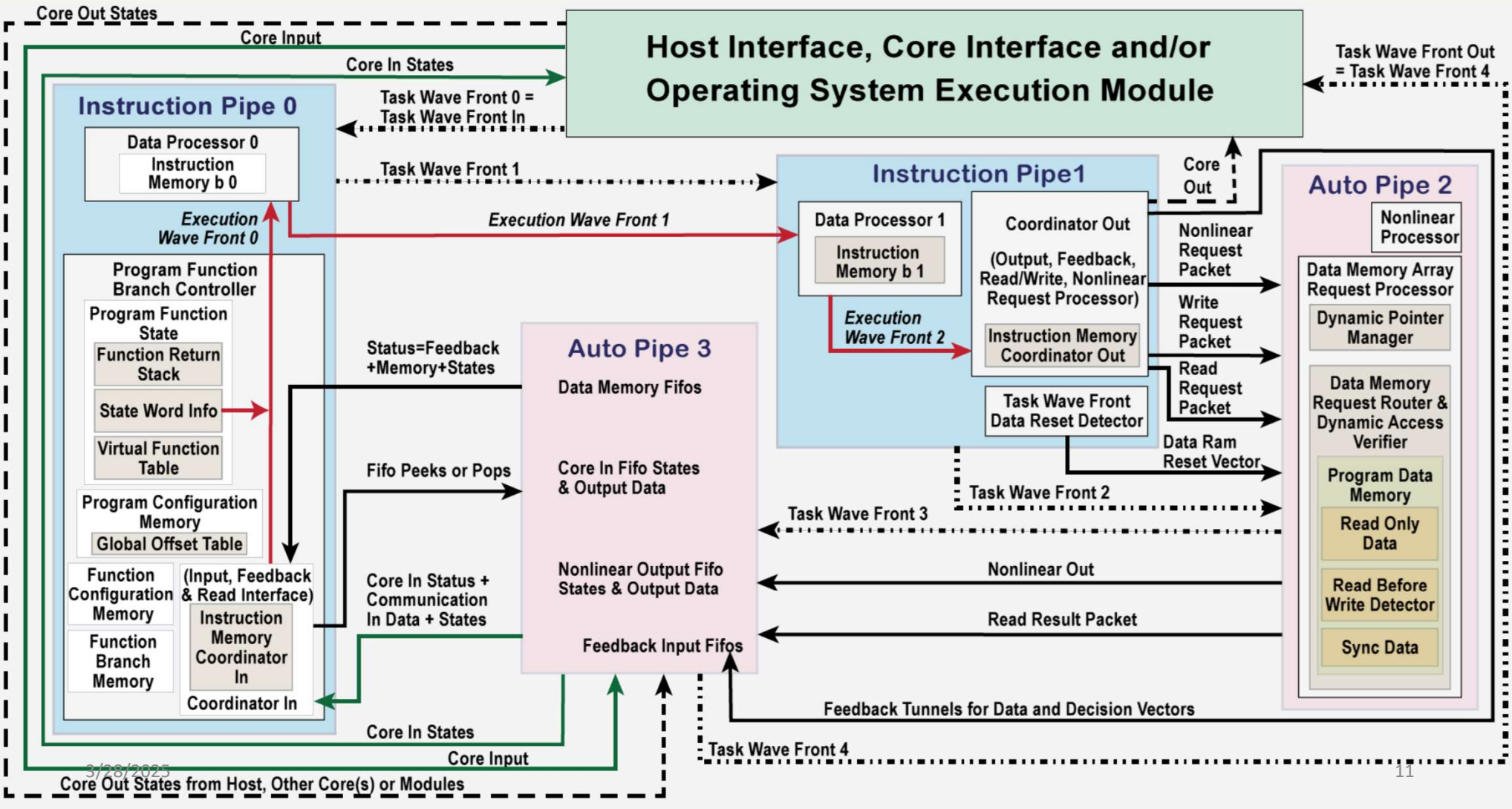
The operating system becomes hardware, the Operating System Execution Module, embodied in a SiMulPro core, operating in real Realtime and invulnerable to attack

All controllers, microprocessors, or state machines can be replaced by realtime SiMulPro cores

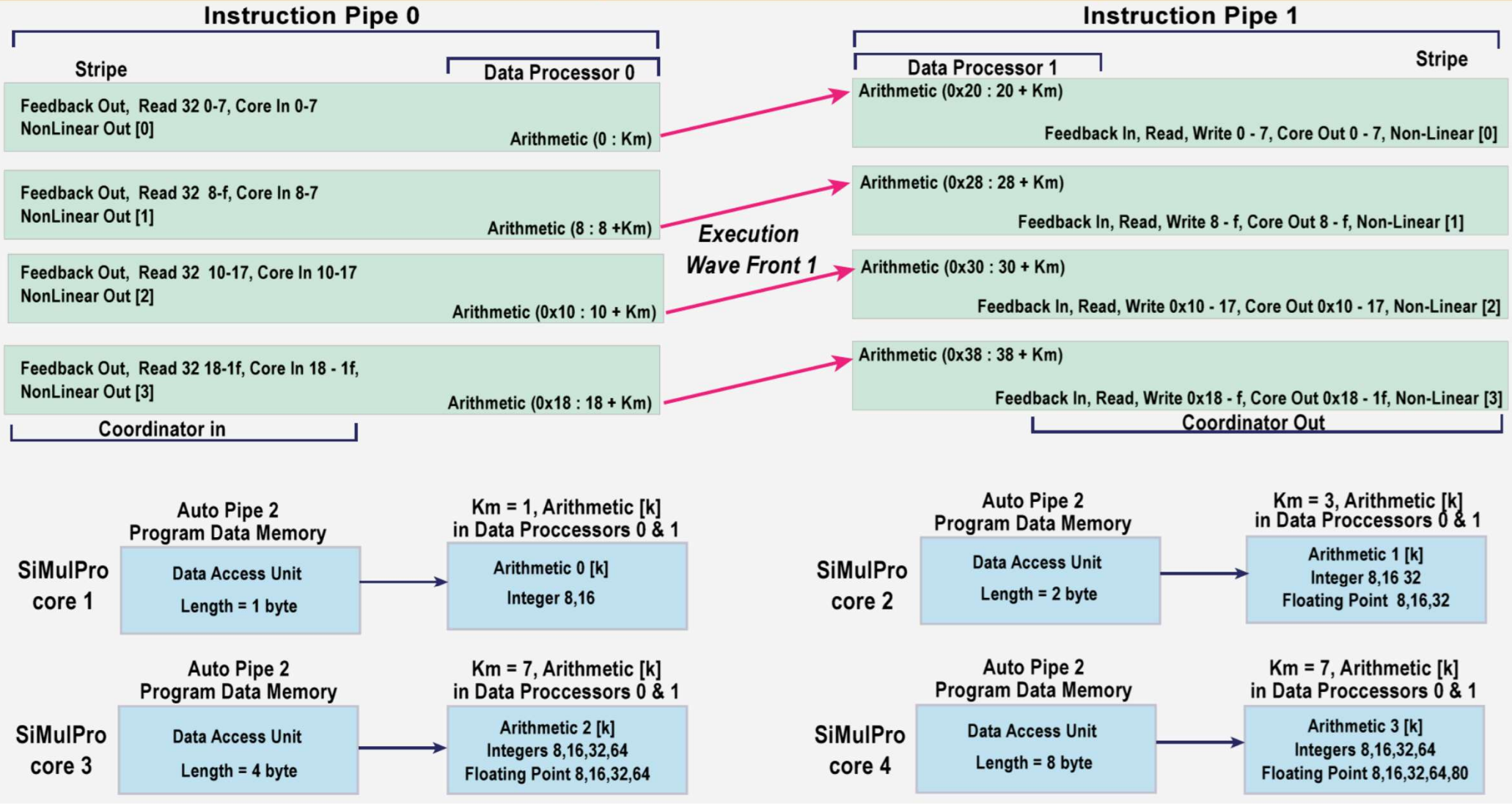
Flexible crossbar switches give performance with speed on each clock cycle
All active source requests are always delivered to requested targets in one clock cycle, eliminating the possibility of asymmetric/unpredictable message delays
A bonus: Energy inefficient caches and buses are no longer required

SiMulPro cores are simpler to design, simpler to build, simpler to program, and debug, than any other approach

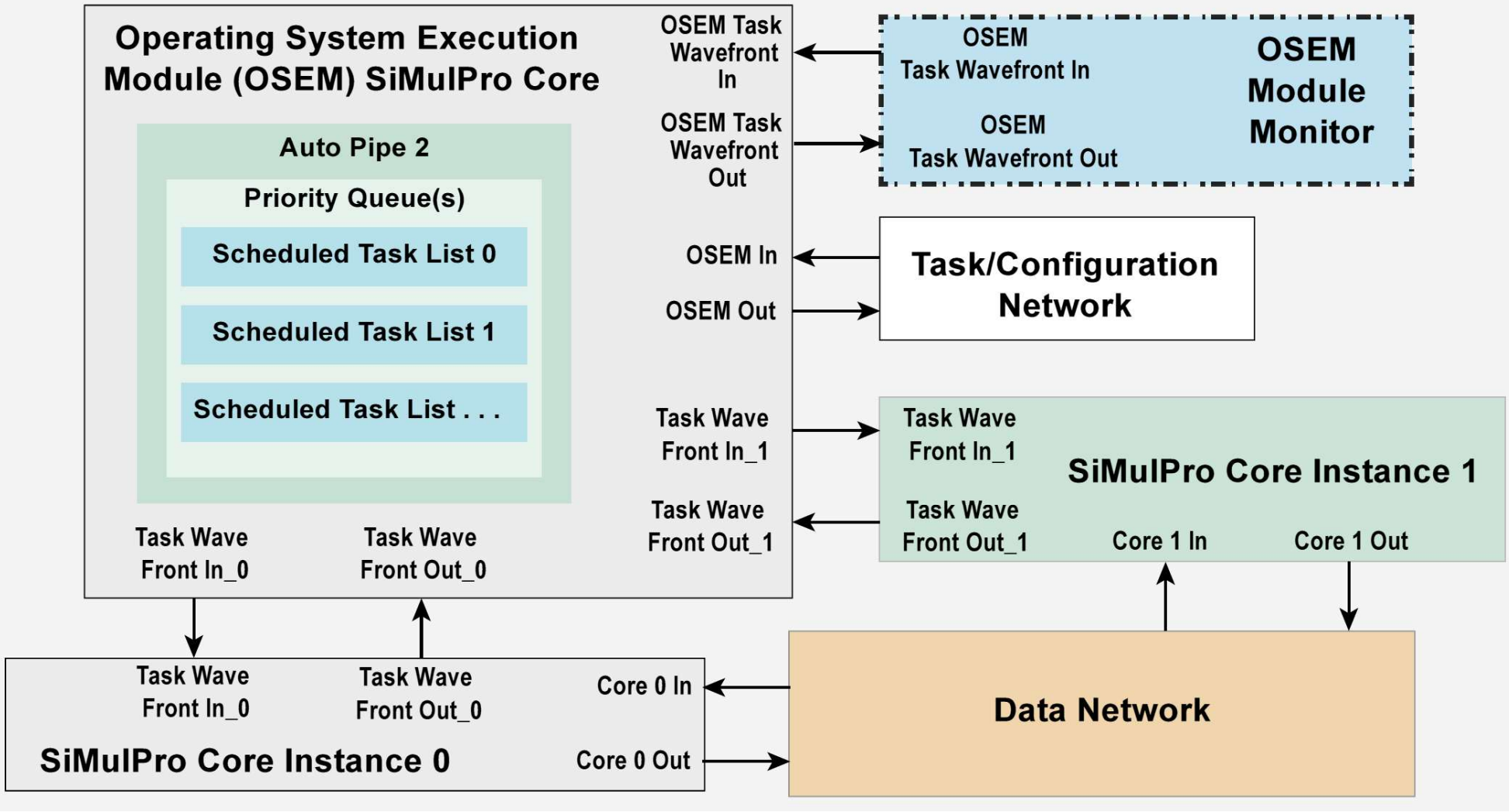
Simultaneous Multi-Processor Core Block Diagram



Overview of Stripes in Instruction Pipes 0 & 1



Operating System Embodied as SiMulPro Core Hardware



SiMulPro Cores Secure Data Centers

SiMulPro cores protect data centers by replacing one general purpose network with two non-overlapping networks, one for data traffic and one for task, configuration, control, and monitoring traffic, completely separating data traffic from task, configuration, control and monitoring traffic

Neither network can sense the other network's traffic. Viruses and rootkits may get into a system via the data network, but they cannot affect (infect) the task/configuration network, nor the instructions of any receiving SiMulPro core

OSEM SiMulPro cores implement any operating system as hardware running each application on data processing SiMulPro core(s)
Each of these cores implement all of the SiMulPro core security features

SiMulPro core Energy usage is estimated to be 2-5% of standard microprocessors

NEXT STEPS

QSigma is seeking $500,000 to build an emulation of a SiMulPro core designed for edge devices.

This core will make the following insecure devices, among others, 100% cybersecure, low power, ease of programming, and scalable:

Augmented Reality (AR)

Autonomous Vehicles

Medical Devices

IoT in Homes and Smart Factories

Surveillance and Security

Traffic Management and Navigation Systems

Video Streaming

This money covers the purchase of test/development tools, FPGA expert support, as well as, compiler support to fix standard code generation problems including pointer/buffer related issues. The compiler will automate translation of C functions and programs by generating the contiguous memory domains required for runtime hardware fault detection and the efficient porting of legacy software.

Once the prototype can be demonstrated, we plan to license this technology to, or be acquired by, an existing company.