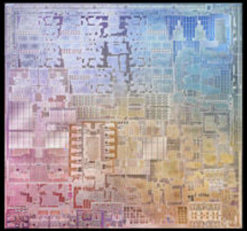
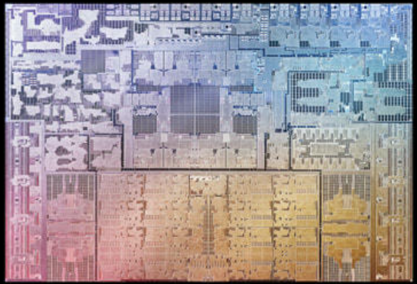
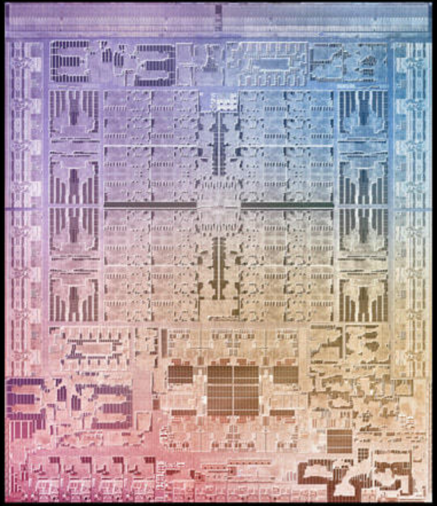
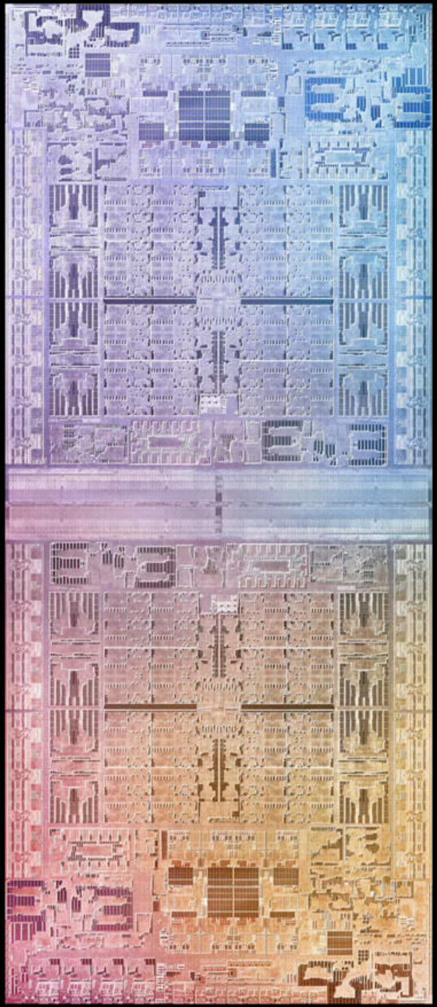
Design and Large Scale Formal Verification of Modern Embedded Microprocessors

Louis-Emile Ploix

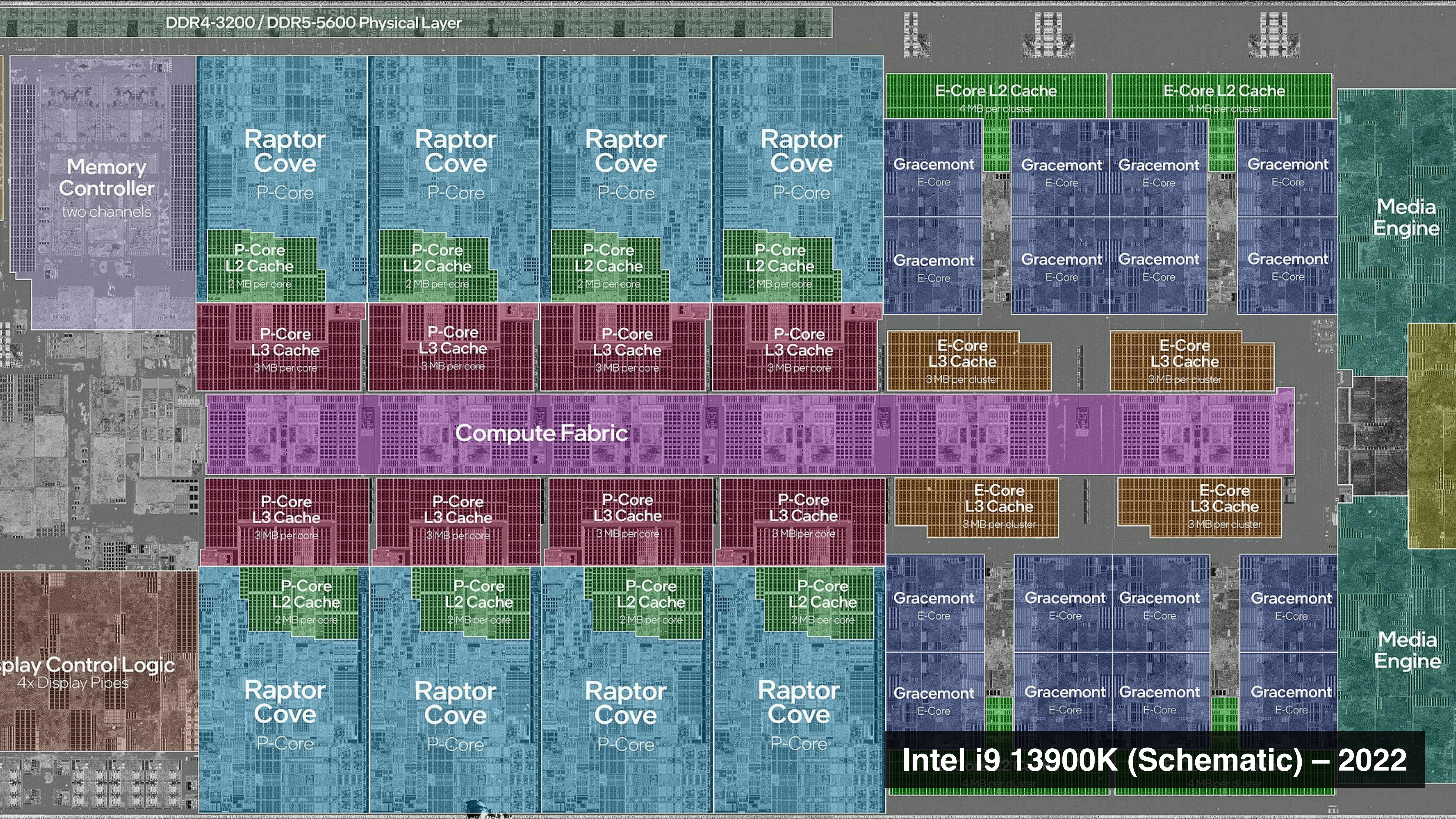




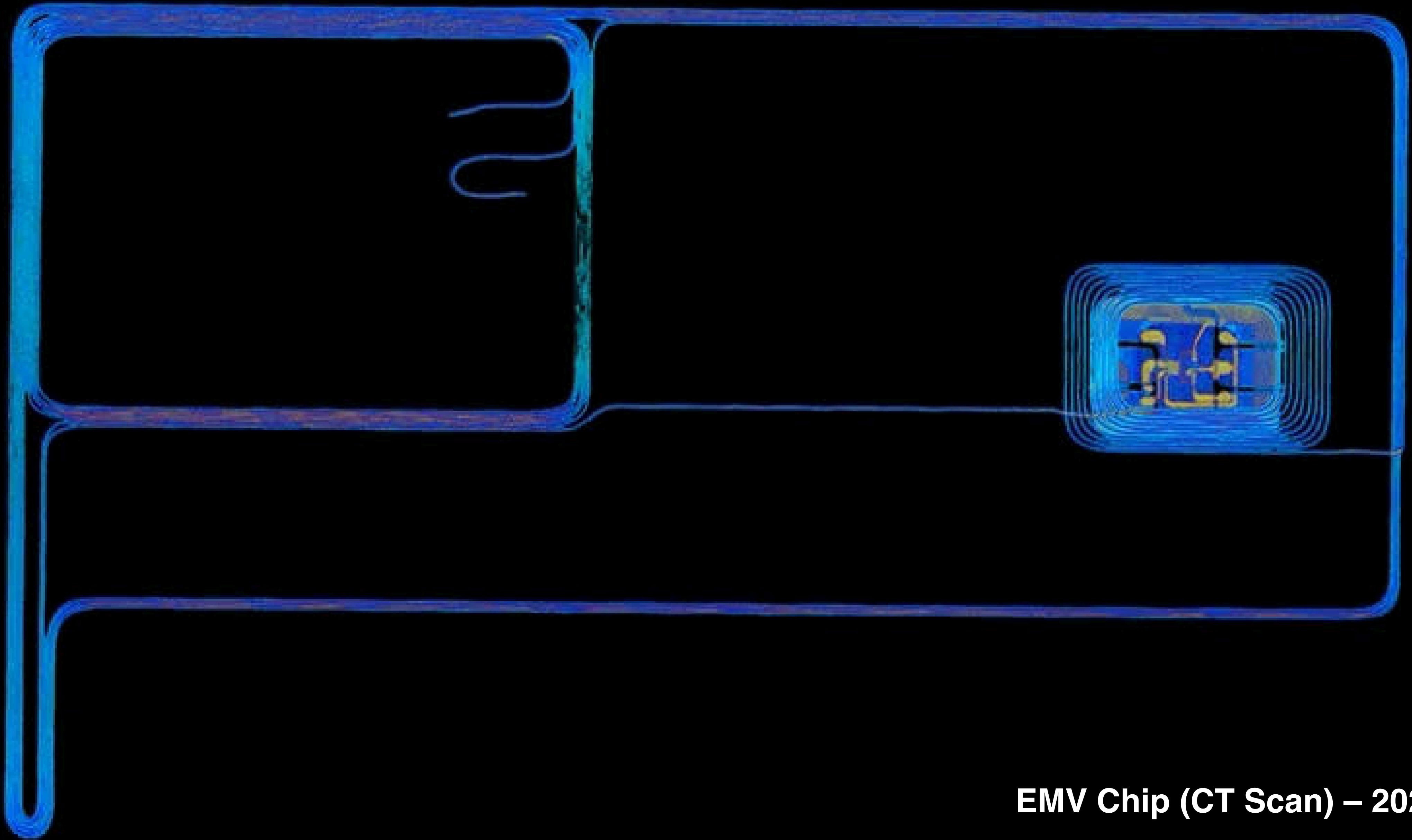




Apple M1 SoC (Die Shot) – 2020



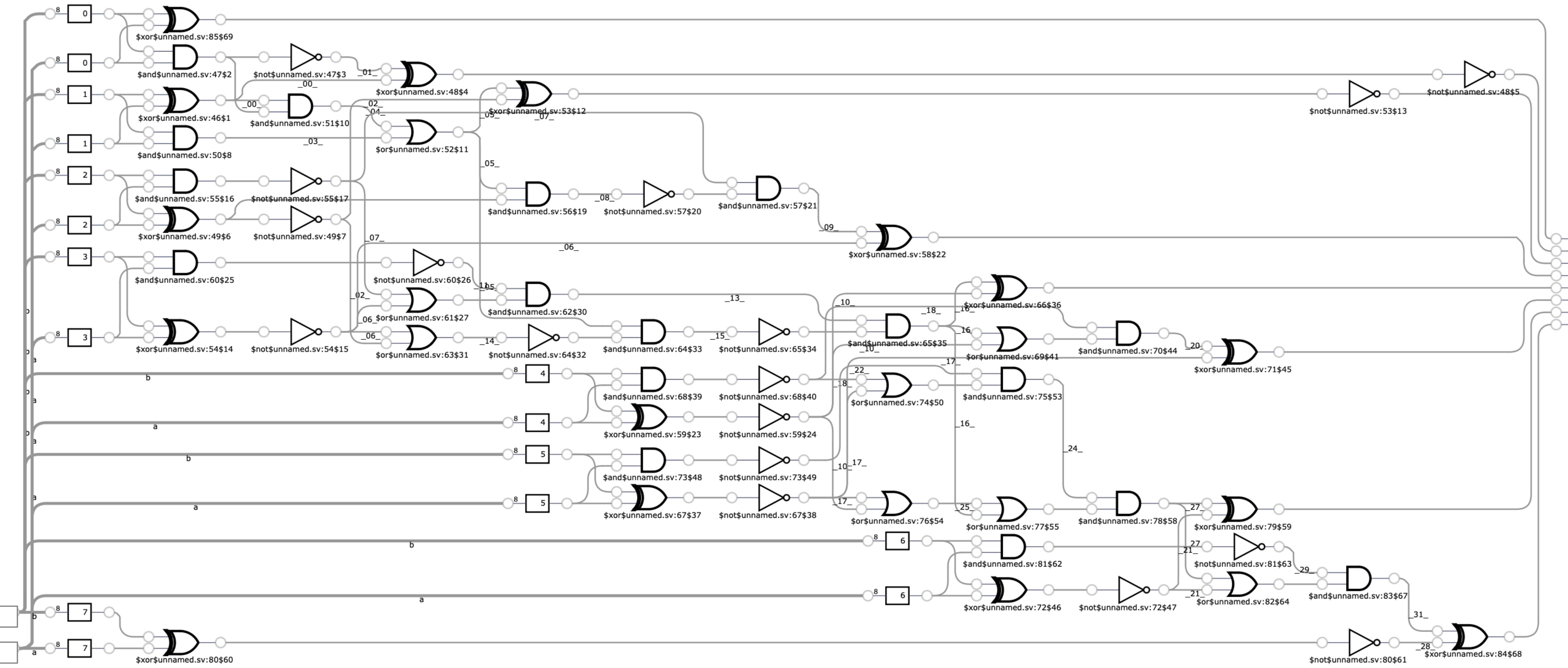
Intel i9 13900K (Schematic) – 2022



EMV Chip (CT Scan) – 2023

8-Bit SystemVerilog Adder

```
module example_adder(
    input logic [7:0] a,
    input logic [7:0] b,
    output logic [7:0] c
);
    assign c = a + b;
endmodule
```



Elaborated 8 Bit Adder Circuit

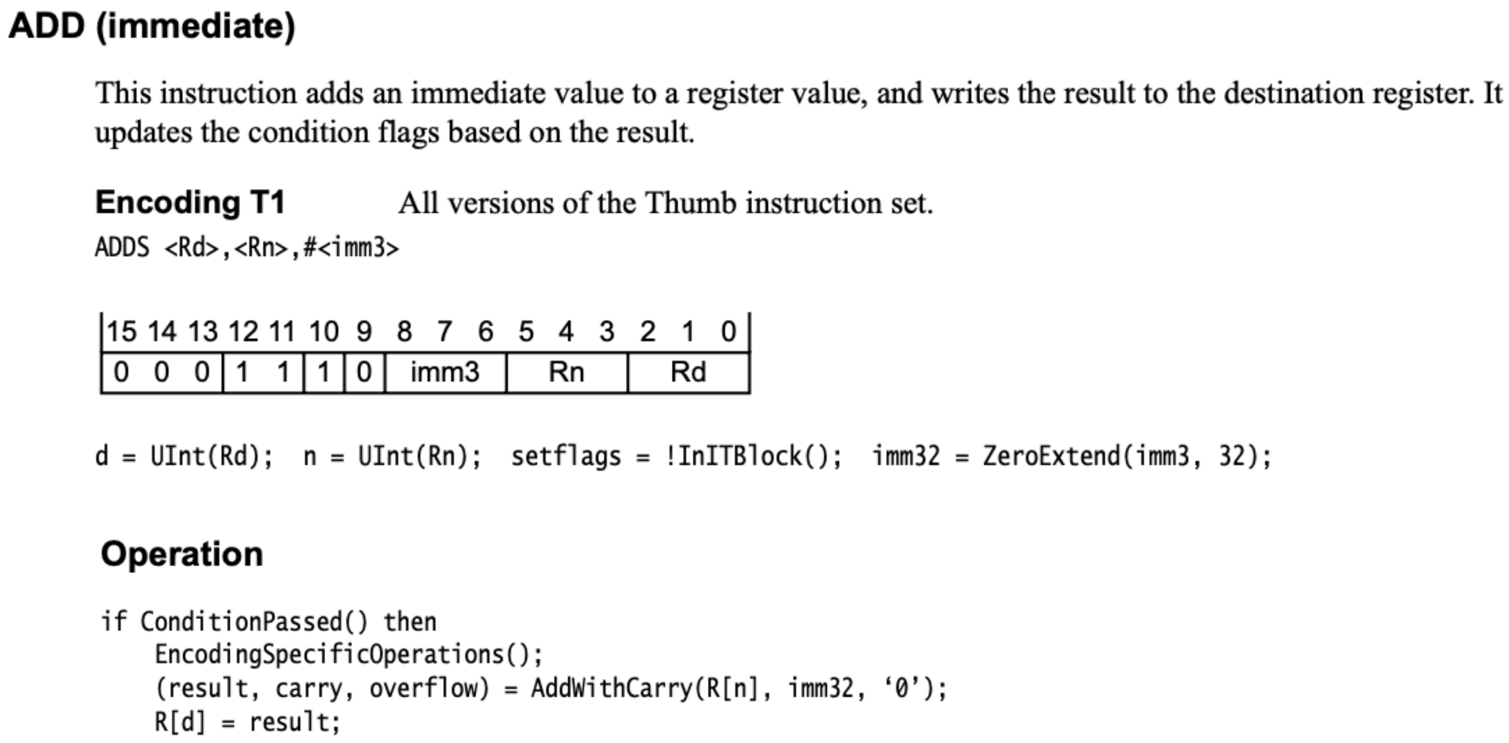
Tiny Compiled Program

```c
int f(int a, int b) {
    return a/b + b + 7;
}
```

```
0000000100003f90 <_f>:
100003f90: 1ac10c08     sdiv  w8, w0, w1 # Signed divide a by b
100003f94: 0b080028     add w8, w1, w8 # Add b to the result
100003f98: 11001d00     add w0, w8, #0x7 # Add 7 to the result
100003f9c: d65f03c0     ret # Finished
```

Compiled with Apple Clang 16.0.0 for an Apple M1 with `-O1`

ArmV6 - Add



ADD (immediate)

This instruction adds an immediate value to a register value, and writes the result to the destination register. It updates the condition flags based on the result.

Encoding T1 All versions of the Thumb instruction set.

```
ADDS <Rd>,<Rn>,#<imm3>
```

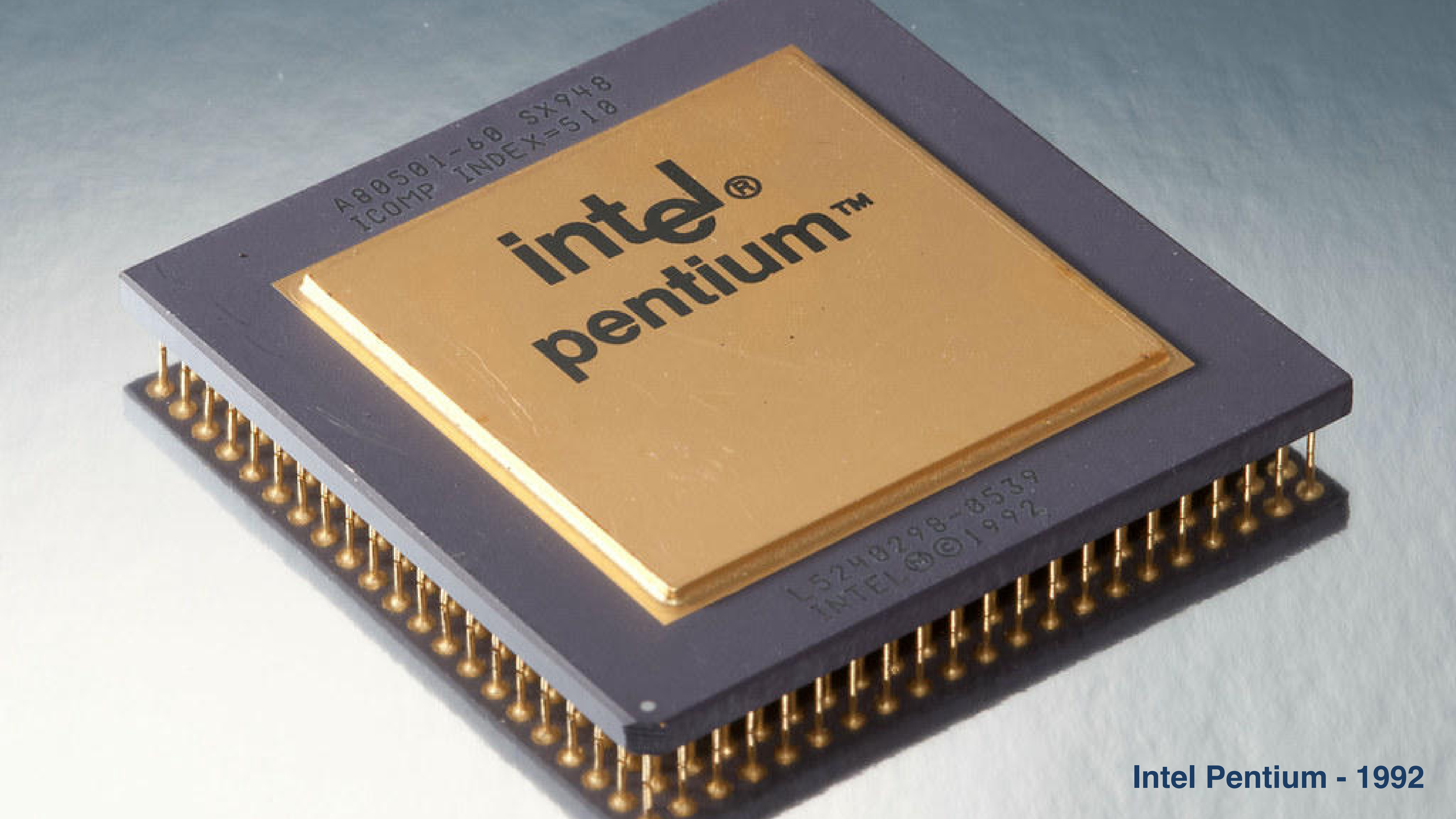
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	1	0	imm3			Rn			Rd		

```
d = UInt(Rd);  n = UInt(Rn);  setflags = !InITBlock();  imm32 = ZeroExtend(imm3, 32);
```

Operation

```
if ConditionPassed() then
    EncodingSpecificOperations();
    (result, carry, overflow) = AddWithCarry(R[n], imm32, ‘0’);
    R[d] = result;
```

From the ARM®v6-M Architecture Reference Manual



Intel Pentium - 1992

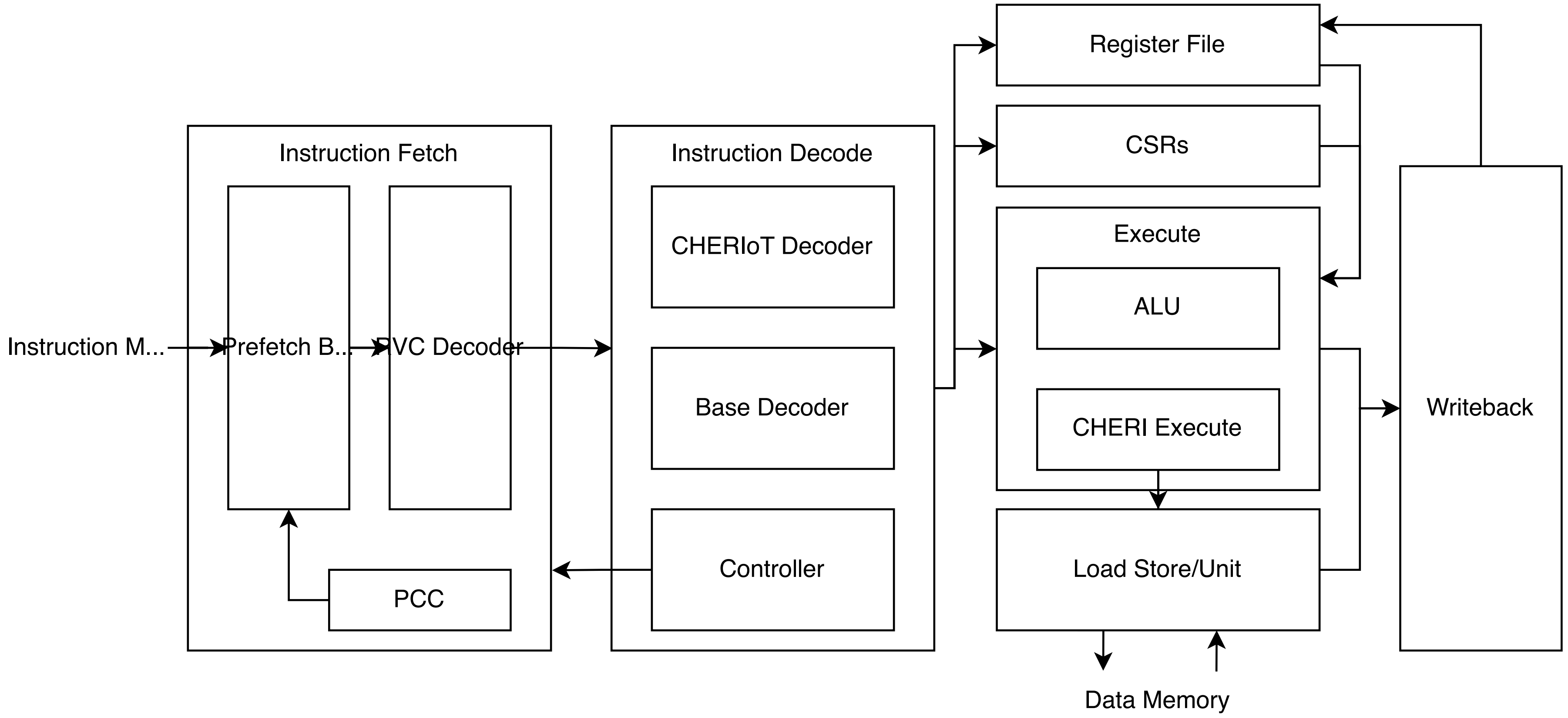
Ad Hoc Block Level Formal Verification - ALU

```
logic [31:0] op_a, op_b, alu_result;
logic [3:0] alu_op;
logic alu_new, alu_ready;

alu alu_i(
  .clk_i, .rst_ni,
  .op_a_i(op_a), .op_b_i(op_b), .op_i(alu_op),
  .result_o(alu_result),
  .new_i(alu_new), .ready_o(alu_ready)
);

// If the ALU receives a multiply req, then three cycles later the
// result is ready and correct.
AluMult: assert property (
  alu_new && alu_op == ALU_MUL |-> ##3 alu_ready && alu_result == op_a * op_b
);
```

CHERIoT-Ibex



Sail Specification of Add Instruction

```
union clause ast = ITYPE : (bits(12), regidx, regidx, iop)

mapping clause encdec = ITYPE(imm, rs1, rd, op) <->
                        imm @ rs1 @ encdec_iop(op) @ rd @ 0b0010011

function clause execute(ITYPE(imm, rs1, rd, op)) = {
  let immext : xlenbits = sign_extend(imm);
  X(rd) = match op {
    RISCV_ADDI  => X(rs1) + immext,
    /* ... */
  };
  RETIRE_SUCCESS
}
```

Bugs

Illegal CLC load ● CLC tag bit leak ● CSeal otypes ● CJALR alignment checks ● CSEQX memory vs. decoded ● MTVEC/MEPC legalisation (in several instances) ● CSC alignment checks ● CSC decoding ● Store local violation ● Memory capability layout ● PCC.address vs. PC ● CJAL vs. CJALR ● **Memory bounds check overflow** ● `tvec_addr` alignment ● MSHWM/MSHWMB updates ● **CLC tag/perms clearing** ● Illegal instruction MTVAL values ● Memory and branch exception priorities ● CSpecialRW exception priorities and SLC issues ● EBreak MTVAL values ● Sealed PCC ● **CSetBounds lower bound check** ● MRet MStatus.MPRV ● 16 vs. 32 register spec issues ● IF granules and overflow ● MEPCC `set_address` ● User mode WFI ● **PMP pipeline flushing on CSR clear** ● CSR instruction problems ● TRVK RF write collision ● Stack_EPC for CHERI NMIs ● Undocumented CJALR

Red *indicates security issues.*

Chip/Spec Trace Equivalence – *Informal Proof*

- Inputs/outputs to/from chip are i^μ/o^μ, inputs/outputs to specification are i/o.
- Sequence of chip/specification states under i^μ/i are s/a respectively.
- For chip state s, define $\mathrm{abs}(s)$ to be the abstract (specification) state representing s.
- Let $\mathrm{Spec}(a, i_k)$ be the next specification state after a on the input i, and let $\mathrm{SpecOut}(a, i_k)$ be the outputs for that period.

Theorem 1 – State Matching: $\mathrm{Spec}(\mathrm{abs}(s_n), i_k) = \mathrm{abs}(s_{n+1})$
Proof: Verified by model checker.

Corollary – Continuity: $a_n = \mathrm{abs}(s_n)$ for all $n \geq 0$.
Proof: By numerical induction on n.

Theorem 2 – Memory: $\mathrm{SpecOut}(\mathrm{abs}(s_n), i_k) = o^\mu_n$
Proof: Verified by model checker.

Corollary – Trace Equivalence: $o_n = o^\mu_n$ for all $n \geq 0$.
Proof: Direct consequence of (1) and (2).