Cyber-Physical Systems

Model Based Design

UNIVERSITY AT ALBANY
State University of New York

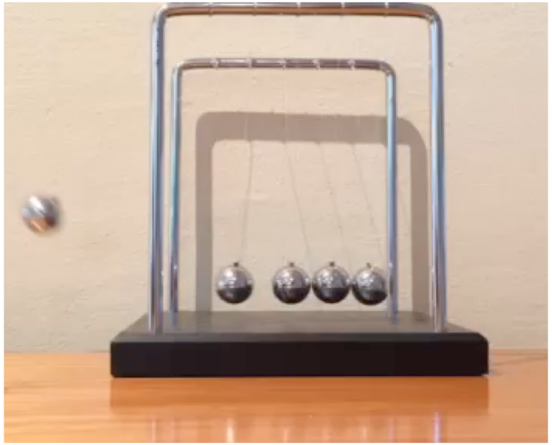
ICEN 553/453– Fall 2018

Prof. Dola Saha

Models vs. Reality

$$x(t) = x(0) + \int_0^t v(\tau)d\tau$$
$$v(t) = v(0) + \frac{1}{m}\int_0^t F(\tau)d\tau,$$

The model



The target

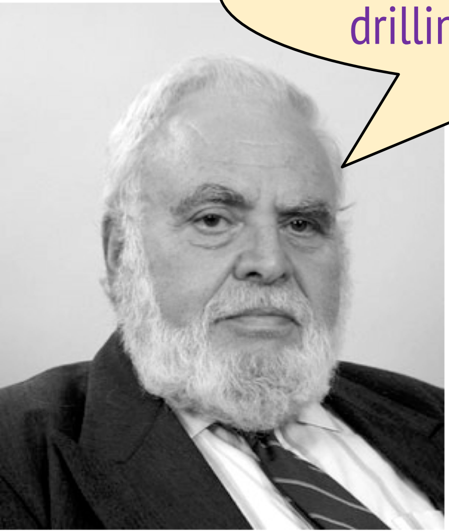
(the thing being modeled).

In this example, the modeling framework is calculus and Newton's laws.

Fidelity is how well the model and its target match.

Engineers often confuse Model with Target

You will never strike oil by drilling through the map!



Solomon Wolf Golomb

But this does not in any way diminish the value of a map!

Determinancy

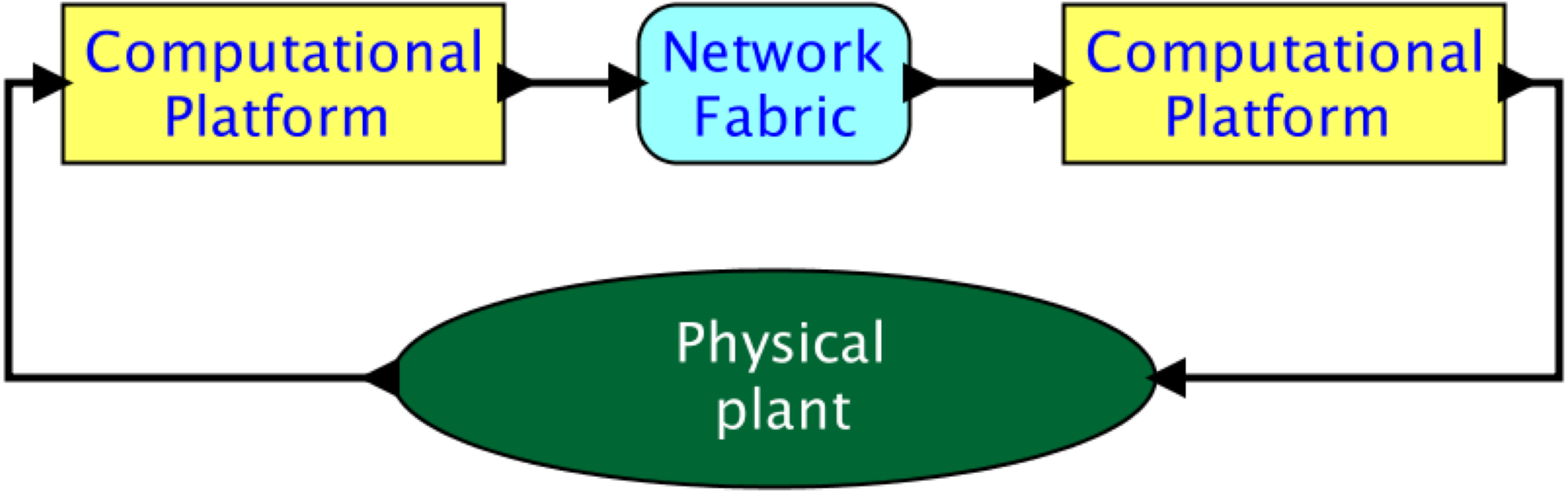
Some of the most valuable models are *deterministic*.

A model is *deterministic* if, given the initial state and the inputs, the model defines exactly one behavior.

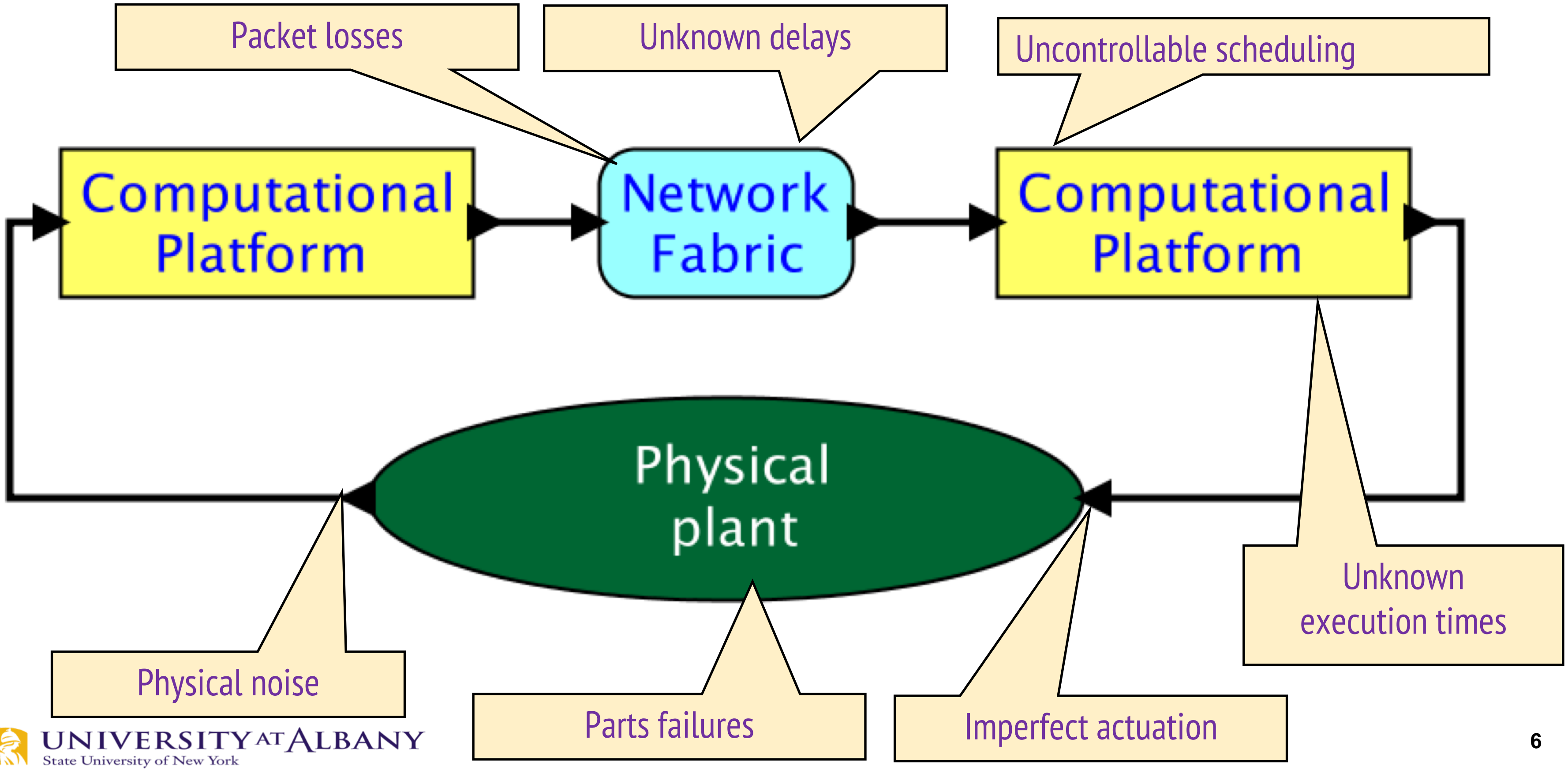
Deterministic models have proven extremely valuable in the past.

In a nondeterministic framework, the model specifies a family of behaviors.

Schematic of a simple CPS



Schematic of a simple CPS - Uncertainties



A Model Need not be True to be Useful

"Essentially, all models are wrong,
but some are useful."

Box, G. E. P. and N. R. Draper, 1987: *Empirical Model-Building and Response Surfaces*. Wiley Series in Probability and Statistics, Wiley.

What kind of Models are Useful?

- The idea that complex physical, biological or sociological systems can be ***exactly*** described by a few formulae is patently absurd.
- Models provide useful approximation.
- Remember that all models are wrong; the practical question is how wrong do they have to be to not be useful.

Software is a Model

Physical System

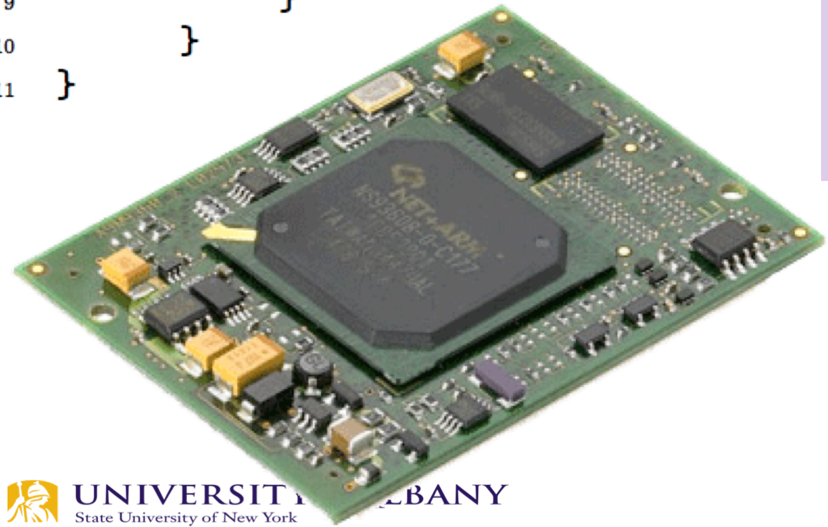
Model

```
/** Reset the output receivers, which are the inside receivers of
 *  the output ports of the container.
 *  @exception IllegalActionException If getting the receivers fails.
 */
private void _resetOutputReceivers() throws IllegalActionException {
    List<IOPort> outputs = ((Actor) getContainer()).outputPortList();
    for (IOPort output : outputs) {
        if (_debugging) {
            _debug("Resetting inside receivers of output port: "
                    + output.getName());
        }
        Receiver[][] receivers = output.getInsideReceivers();
        if (receivers != null) {
            for (int i = 0; i < receivers.length; i++) {
                if (receivers[i] != null) {
                    for (int j = 0; j < receivers[i].length; j++) {
                        if (receivers[i][j] instanceof FSMReceiver) {
                            receivers[i][j].reset();
                        }
                    }
                }
            }
        }
    }
}
```

Single-threaded imperative programs are deterministic models

Single Threaded Imperative Program

```
void foo(int32_t x) {
        if (x > 1000) {
                x = 1000;
        }
        if (x > 0) {
                x = x + 1000;
                if (x < 0) {
                        panic();
                }
        }
}
```



This program defines exactly one behavior, given the input x.

Note that the modeling framework (the C language, in this case) defines “behavior” and “input.”

The target of the model is nondeterministic (electrons sloshing around in silicon).

Underlying Hardware

Physical System

Image: Wikimedia Commons

Model

Integer Register-Register Operations

RISC-V defines several arithmetic R-type operations. All operations read the *rs1* and *rs2* registers as source operands and write the result into register *rd*. The *funct* field selects the type of operation.

31 — 27	26 — 22	21 — 17	16 — 7	6 — 0
rd	rs1	rs2	funct10	opcode
5	5	5	10	7
dest	src1	src2	ADD/SUB/SLT/SLTU	OP
dest	src1	src2	AND/OR/XOR	OP
dest	src1	src2	SLL/SRL/SRA	OP
dest	src1	src2	ADDW/SUBW	OP-32
dest	src1	src2	SLLW/SRLW/SRAW	OP-32

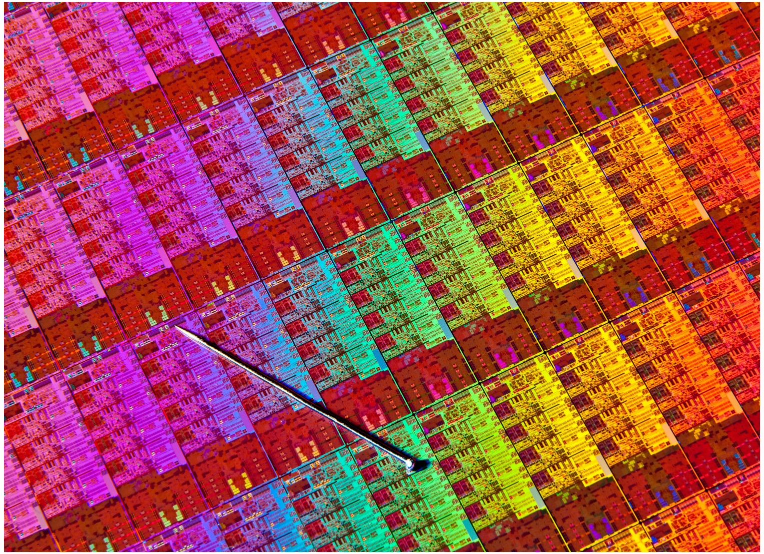
Waterman, et al., The RISC-V Instruction Set Manual, UCB/EECS-2011-62, 2011

Software relies on deterministic model that abstracts the hardware

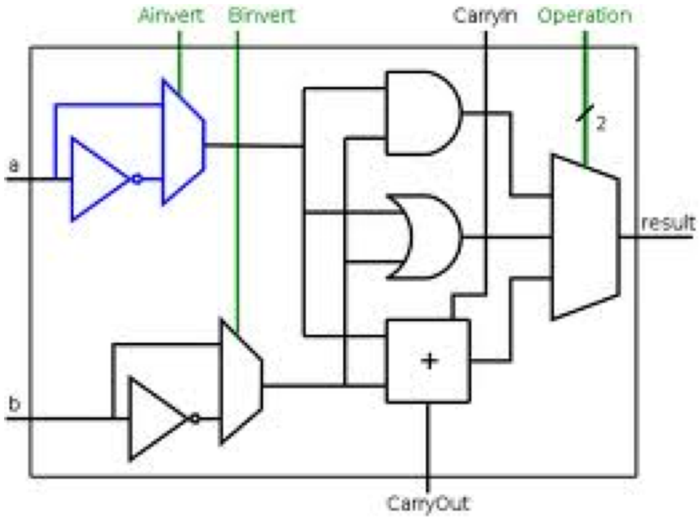
Instruction Set Architectures (ISAs) are deterministic models

Underlying Digital Logic

Physical System



Model



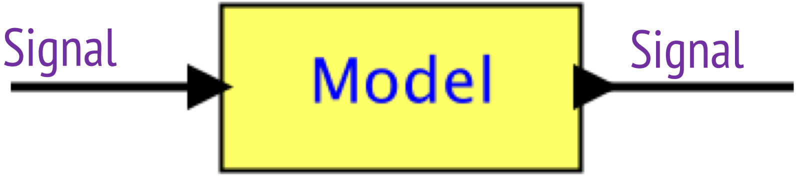
Synchronous digital logic is a deterministic model

Deterministic Models for the Physical Side of CPS

Physical System

Image: Wikimedia Commons

Model



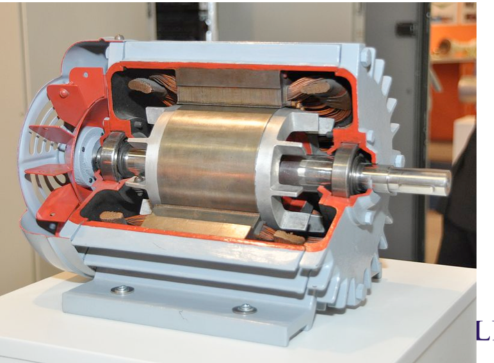
$$\dot{\mathbf{x}}(t) = \dot{\mathbf{x}}(0) + \frac{1}{M}\int_0^t \mathbf{F}(\tau)d\tau$$

Differential Equations are deterministic models

Major Problem of CPS

Combinations of Deterministic Models are nondeterministic


```
void initTimer(void) {
    SysTickPeriodSet(SysCtlClockGet() / 1000);
    SysTickEnable();
    SysTickIntEnable();
}
volatile uint timer_count = 0;
void ISR(void) {
    if(timer_count != 0) {
        timer_count--;
    }
}
int main(void) {
    SysTickIntRegister(&ISR);
    .. // other init
    timer_count = 2000;
    initTimer();
    while(timer_count != 0) {
        ... code to run for 2 seconds
    }
    ... // other code
}
```

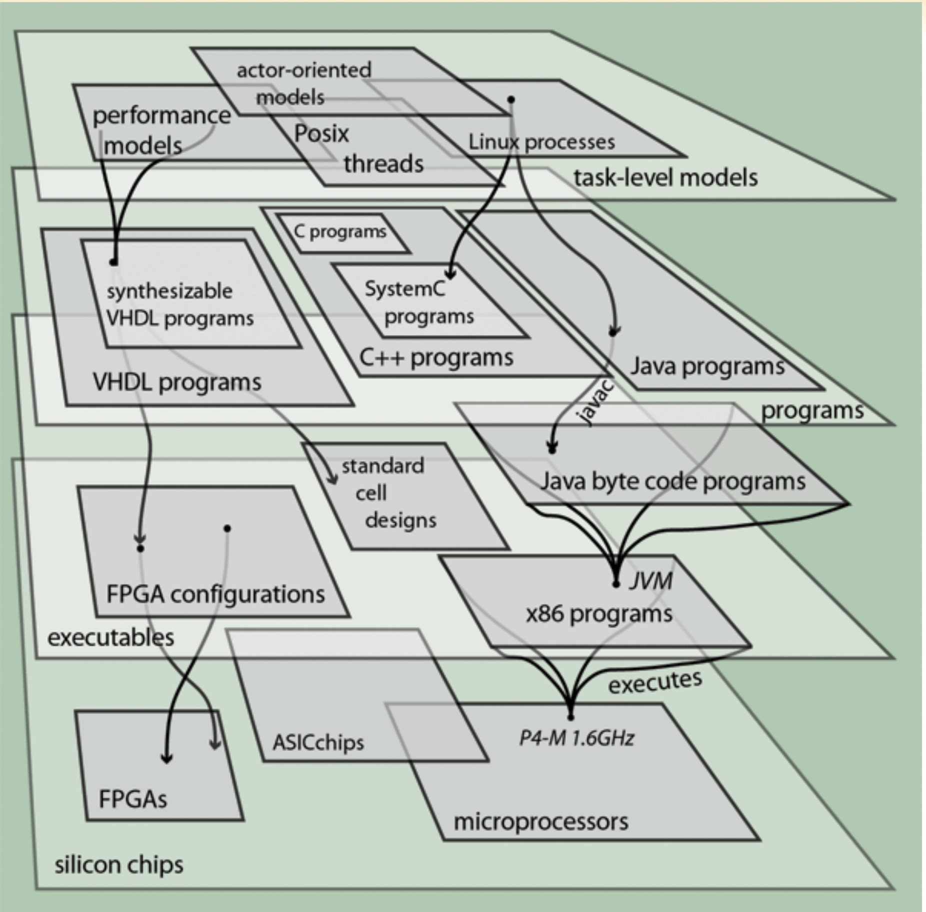


$$\dot{\mathbf{x}}(t) = \dot{\mathbf{x}}(0) + \frac{1}{M}\int_{0}^{t} \mathbf{F}(\tau)d\tau$$

CPS in Flight

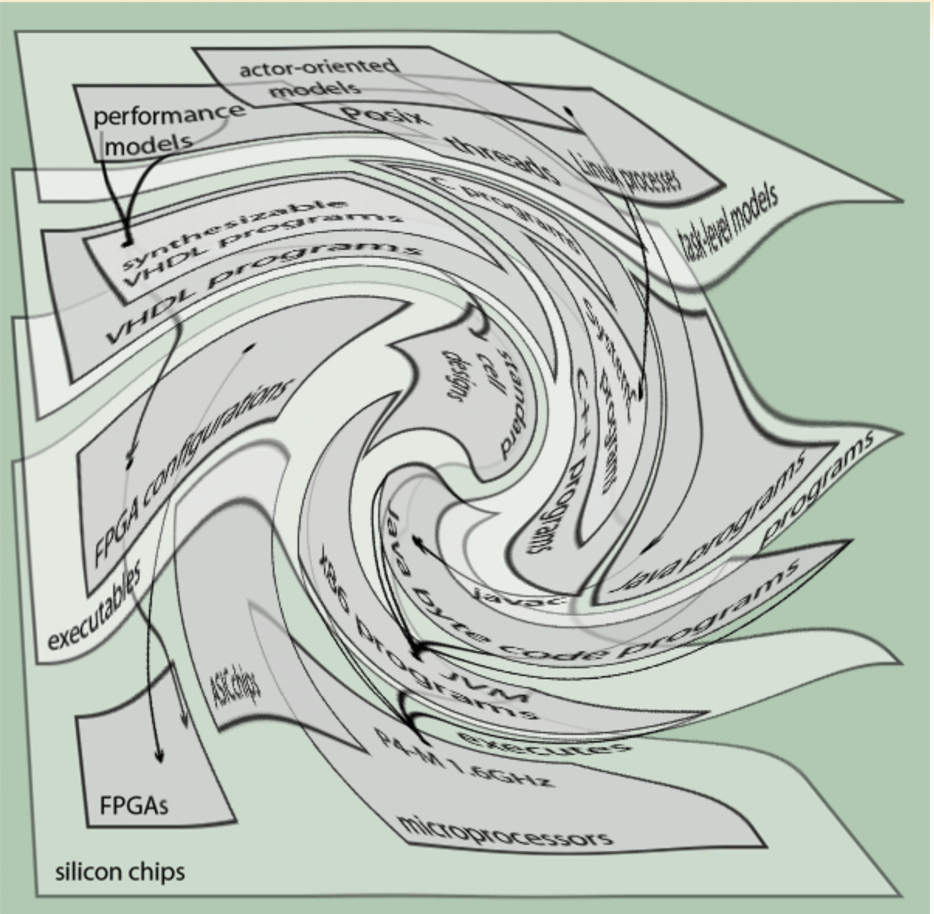
In “fly by wire” aircraft, computers control the plane, mediating pilot commands.

Abstraction Layers



The purpose of an abstraction is to hide details of the implementation below and provide a platform for design from above.

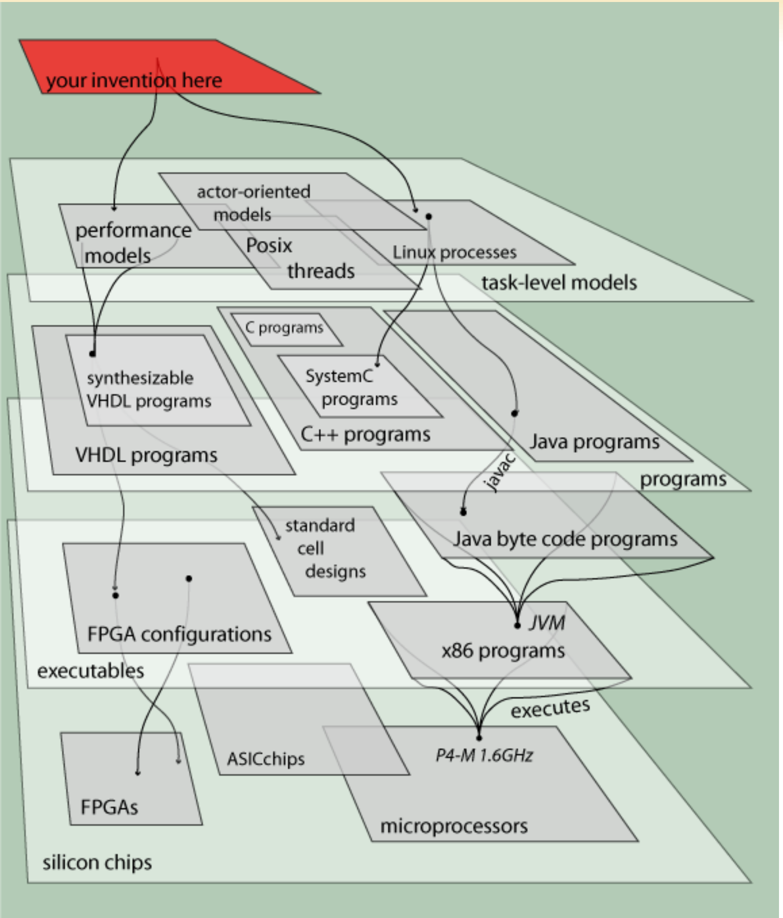
Abstraction Layers



Every abstraction layer has failed for the aircraft designer.

The design *is* the implementation.

Abstraction Layers



How about raising the level of abstraction to solve these problems?

Higher abstractions -> increasingly problematic

A story:

Ferdinand et al. [2001] determine the WCET of astonishingly simple avionics code from Airbus running on a Motorola ColdFire 5307, a pipelined CPU with a unified code and data cache. Despite the software consisting of a fixed set of non-interacting tasks containing only simple control structures, their solution required detailed modeling of the seven-stage pipeline and its precise interaction with the cache, generating a large integer linear programming problem.

Fundamentally, the ISA of the processor has failed to provide an adequate abstraction. And the problem has gotten worse since 2001!

Timing is not Part of Software Semantics

- Correct execution of a program in all widely used programming languages, and correct delivery of a network message in all general-purpose networks has nothing to do with how long it takes to do anything.

- Programmers have to step outside the programming abstractions to specify timing behavior.

- Embedded software designers have no map!

Determinism? Really?

CPS applications operate in an intrinsically nondeterministic world.

Does it really make sense to insist on deterministic models?

The Value of Models

In *science*, the value of a *model* lies in how well its behavior matches that of the physical system.

In *engineering*, the value of the *physical system* lies in how well its behavior matches that of the model.

In engineering, model fidelity is a two-way street!

For a model to be useful, it is necessary
(but not sufficient) to be able to
construct a faithful physical realization.

Model Fidelity

To a *scientist*, the model is flawed.

To an *engineer*, the realization is flawed.

For CPS

The question is *not* whether deterministic models can describe the behavior of cyber-physical systems (with high fidelity).

The question is whether we can build cyber-physical systems whose behavior matches that of a deterministic model (with high probability).

What about Resilience? Adaptability?

Deterministic models do not eliminate the need for robust, fault-tolerant designs.

In fact, they *enable* such designs, because they make it much clearer what it means to have a fault!