Precision-Timed (PRET) Machines

Stephen A. Edwards

Columbia University

Edward A. Lee

University of California, Berkeley

A Major Historical Event

In 1980, Patterson and Ditzel did not invent reduced instruction set computers (RISC machines).

D. A. Patterson and D. R. Ditzel, “The case for the reduced instruction set computer,” ACM SIGARCH Computer Architecture News, 8(6):25-33, Oct. 1980.

Another Major Historical Event

In 2006, Lee and Edwards did not invent reduced precision-timed computers (PRET machines).

S. A. Edwards and E. A. Lee, "The Case for the Precision Timed (PRET) Machine," EECS Department, University of California, Berkeley, Technical Report No. UCB/EECS-2006-149, November 17, 2006.

http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-149.html

The World as We Know It

We do not consider how fast a processor runs when we evaluate whether it is "correct."

Salvador Dali, *The Persistence of Memory*, 1931. (detail)

This Is Sometimes Useful For

- Programming languages
- Virtual memory
- Caches
- Dynamic dispatch
- Speculative execution
- Power management (voltage scaling)
- Memory management (garbage collection)
- Just-in-time (JIT) compilation
- Multitasking (threads and processes)
- Component technologies (OO design)
- Networking (TCP)

But Time Sometimes Matters

Kevin Harvick winning the Daytona 500 by 20 ms, February 2007. (Source: Reuters)

Certification in Avionics

- Rather expensive
- Software is *not* certified
- Entire system is certified
- Slight change, e.g., in the microprocessor, requires recertification
- Solution: stockpile parts; trust nobody

(Source: NASA)

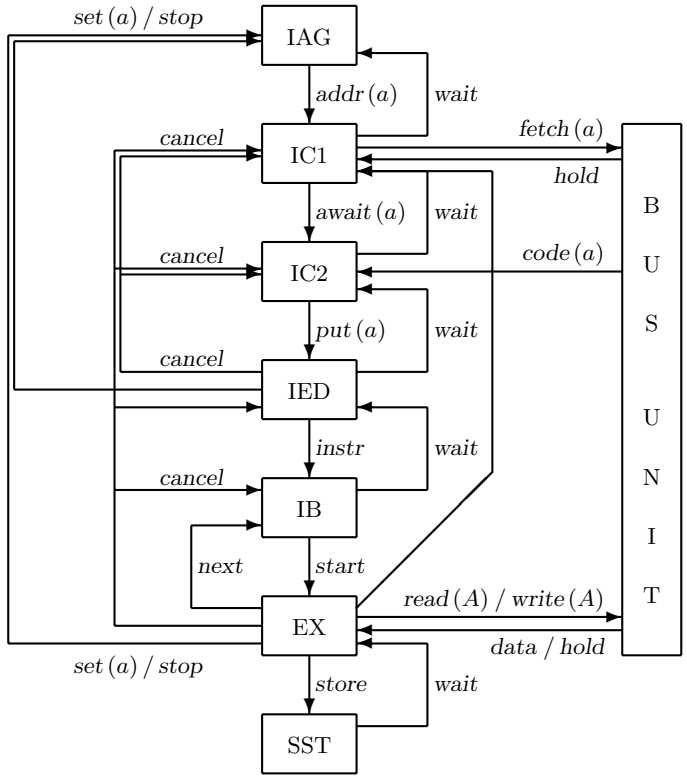
Worst-Case Execution Time

Virtually impossible to compute on modern processors.

Feature	**Nearby instructions**	**Distant instructions**	**Memory layout**
Pipelines	√		
Branch Prediction	√	√	
Caches	√	√	√

State-of-the-art WCET

- Motorola ColdFire
- Two coupled pipelines (7-stage)
- Shared instruction & data cache
- Artificial example from Airbus
- Twelve independent tasks
- Simple control structures
- Cache/Pipeline interaction leads to large integer linear programming problem



C. Ferdinand et al., "Reliable and precise WCET determination for a real-life processor," EMSOFT 2001

The Problem

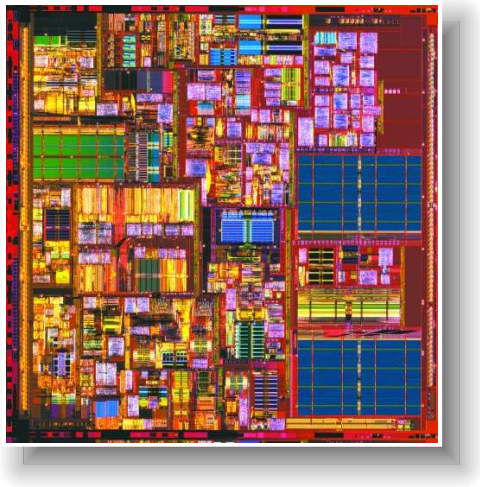
Digital hardware provides extremely precise timing

20.000 MHz ($\pm$ 100 ppm)

and architectural complexity discards it.

Our Vision: PRET Machines

PREcision-Timed processors: Performance & Predicability



+

= PRET

(Image: John Harrison's H4, first clock to solve longitude problem)

Our Vision: PRET Machines

Predictable performance, not just good average case

Current	**Alternative**
Caches	Scratchpads
Pipelines	Thread-interleaved pipelines
Function-only ISAs	ISAs with timing
Function-only languages	Languages with timing
Data Races	Deterministic concurrency
Best-effort communication	Fixed-latency communication

Related Work: Giotto

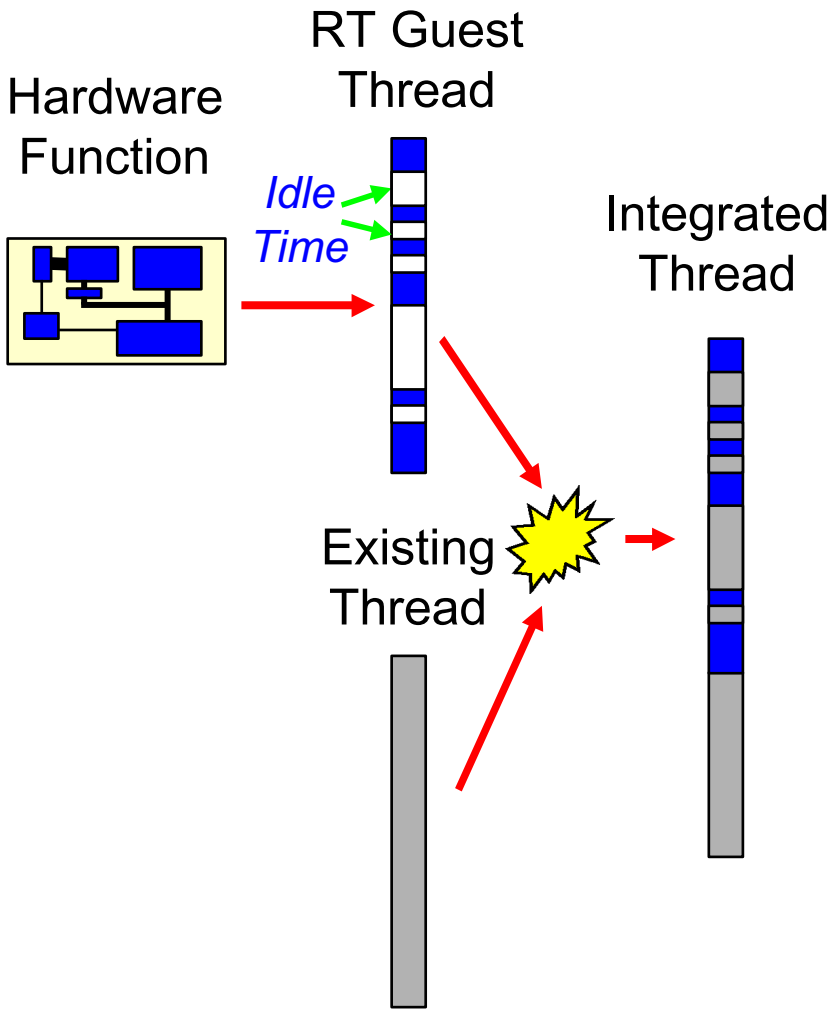
Giotto [Henzinger, Horowitz, Kirsch, Proc. IEEE 2003]

The RTOS style: specify a collection of tasks and modes. Compiler produces schedule (task priorities).

Precision limited by periodic timer interrupt.

```
mode forward() period 200 {
  actfreq 1 do leftJet(leftMotor);
  actfreq 1 do rightJet(rightMotor);
  exitfreq 1 do point(goPoint);
  exitfreq 1 do idle(goIdle);
  exitfreq 1 do rotate(goRotate);
  taskfreq 2 do errorTask(getPos);
  taskfreq 1 do forwardTask(getErr);
}
```

Related Work: STI



Software Thread Integration [Dean, RTSS 1998]

Insert code for a non-real-time thread into a real-time thread.

Pad the rest with NOPs

Often creates code explosion

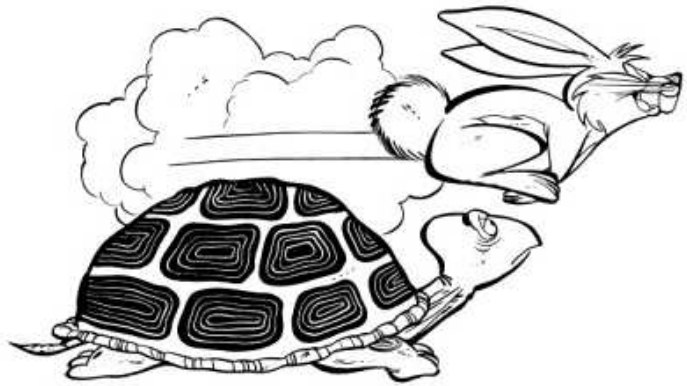
Requires predictable processor; he uses AVRs

Related Work: VISA

VISA [Meuller et al., ISCA 2003]

Run two processors:

- Slow and predictable
- Fast and unpredictable



Start tasks on both.

If fast completes first, use extra time.

If fast misses a checkpoint, switch over to slow.

A First Attempt

16-bit MIPS-like processor augmented with timers

One additional "deadline" instruction:

dead *timer*, *timeout*

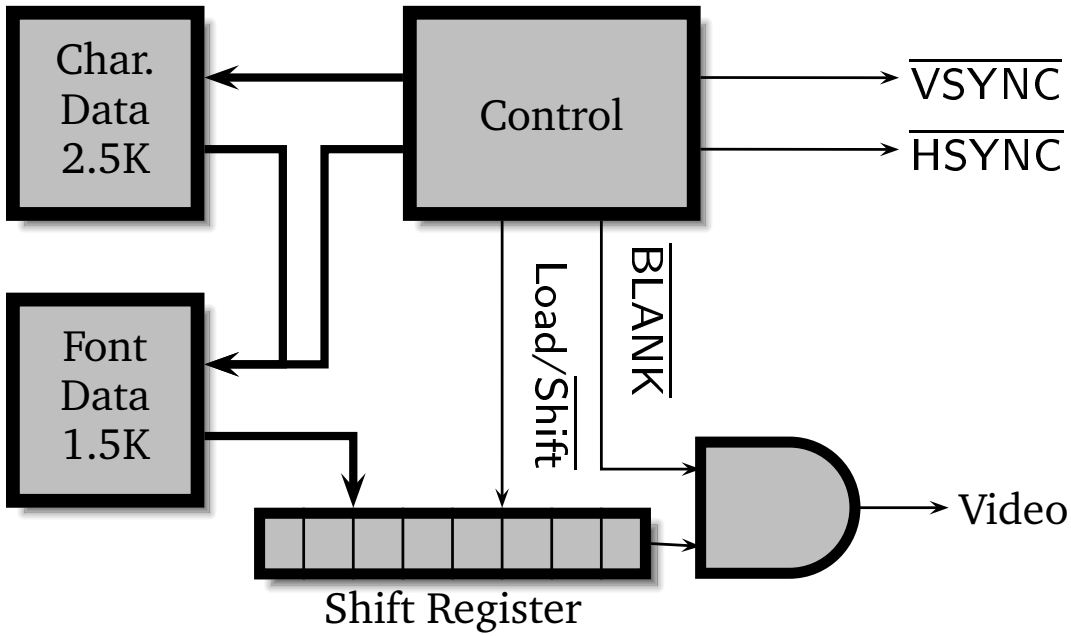
Wait until *timer* expires, then reload it with *timeout*.

Nicholas Ip and Stephen A. Edwards, "A Processor Extension for Cycle-Accurate Real-Time Software," Proceedings of EUC, Seoul, Korea, August 2006.

Case Study: Video

80 $\times$ 30 text-mode display, 25 MHz pixel clock

Shift register in hardware; everything else in software



Case Study: Video

```
   movi   $2, 0              ; reset line address
row:
   movi   $7, 0              ; reset line in char
line:
   deadi  $t1, 96            ; h. sync period
   movi   $14, HS+HB
   ori    $3, $7, FONT       ; font base address
   deadi  $t1, 48            ; back porch period
   movi   $14, HB
   deadi  $t1, 640           ; active video period
   mov    $1, 0              ; column number
char:
   lb     $5, ($2+$1)        ; load character
   shli   $5, $5, 4          ; *16 = lines/char
   deadi  $t0, 8             ; wait for next character
   lb     $14, ($5+$3)       ; fetch and emit pixels
   addi   $1, $1, 1          ; next column
   bne    $1, $11, char
   deadi  $t1, 16            ; front porch period
   movi   $14, HB
   addi   $7, $7, 1          ; next row in char
   bne    $7, $13, line      ; repeat until bottom
   addi   $2, $2, 80         ; next line
   bne    $2, $12, row       ; until at end
```

Two nested loops:

- Active line
- Character

Two timers:

- $t1 for line timing
- $t0 for character output

78 lines of assembly replaces 450 lines of VHDL (1/5th)

Conclusions

- Embedded applications need timing control
- We need hardware support
- High-performance processors with predictable timing
- Predictable performance our mantra
- A first cut: MIPS-like processor with timers
- *Dead* instruction waits for timeout, then reloads
- Video controller 1/5 the size of VHDL