Outline

1. Administrivia
   - Coordinates, officialdom, assessment

2. Course overview
   - Motivation, and course content
   - Real Time Systems
   - Formal methods

3. A preview...
   - Formalisms
   - System architectures for hard real time
Hugh’s coordinates

<table>
<thead>
<tr>
<th>Room</th>
<th>S15 #06-12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Telephone</td>
<td>6516-6903</td>
</tr>
<tr>
<td>E-mail</td>
<td><a href="mailto:hugh@comp.nus.edu.sg">hugh@comp.nus.edu.sg</a></td>
</tr>
</tbody>
</table>

and Spinelli’s of course!

Open-door policy (I have one!)
Please call me Hugh, and visit me in my room if you have any questions...
From the official course description...

This one semester first-year graduate course will provide an introduction to the analysis and verification of hard real time systems. These are systems typically running embedded distributed applications that must meet their temporal constraints in a range of anticipated load and fault scenarios. The course will concentrate on specification and verification aspects of distributed real time applications. The focus will be on the tools and techniques based on timed automata using, which one can verify that the scheduled behavior of a real time distributed system will meet its critical timing constraints. The overall goal is to provide the student with the current scientific and engineering insights that are relevant for the analysis and verification of distributed embedded real time systems.
## Assessment

MC 4, 3(Lect)-0(Tut)-0(Lab)-3(Proj)-4(Prep)

<table>
<thead>
<tr>
<th>Assessment</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Projects/Assignments</td>
<td>35%</td>
</tr>
<tr>
<td>Tests</td>
<td>Closed book</td>
</tr>
<tr>
<td>Final Exam</td>
<td>(Thurs 3 May) Open Book</td>
</tr>
<tr>
<td>Total marks</td>
<td></td>
</tr>
</tbody>
</table>

May vary the assessment in some small ways... Still learning
Resources

I will provide course notes and slides, and also

- selected parts of:
  - Real-Time Systems: Design Principles for Distributed Embedded Applications: Hermann Kopetz
  - Concepts, Algorithms and Tools for Model Checking: Lecture notes by Joost-Peter Katoen:

- Directed readings - all available on the Internet.
The project/assignment work is still not fixed, but is likely to ...

- Individual work - no group work.
Consider the Ubiquity of processors...

- Ariane-5 explosion
  - underlying cause was software that was re-used, and never tested, as it was considered too expensive, and needed real-time input from sensors moving in space.
  - Fault involved time delays.
  - It is likely that the problem could have been found without an explosion.
Do we need any motivation???

Software versus hardware...

- Consider these two warranties:
  - PC Manufacturer warrants that (a) the SOFTWARE will perform substantially in accordance with the accompanying written materials for a period of ninety (90) days from the date of receipt, and (b) any Microsoft hardware accompanying the SOFTWARE will be free from defects in materials and workmanship under normal use and service for a period of one (1) year from the date of receipt.
  - ACCTON warrants to the original owner that the product delivered in this package will be free from defects in material and workmanship for the lifetime of the product.

- Software is considered less reliable
Do we need any motivation???

Abstraction and software engineering...

Consider these two approaches to checking system behaviour:

1. “model it using a small C program”, “Run it a few times and see what happens”, or perhaps “Start with a file with one record of each type, then try a bigger file until a pattern emerges”.

2. Turn to mathematics for help.

When software engineers meet a problem that is too large or difficult to understand, they sometimes have a poor attitude..
Do we need any motivation???

from RISKS...

- The LA counties pension fund lost US$1,200,000,000 through programming error.
- A Mercedes 500SE with graceful no-skid brake computers left 200m skid marks. A passenger was killed.
- A computer controlled elevator door in Ottawa killed two people.
- An automated toilet seat in Paris killed a child.
- The Amsterdam air-freight computer software crashed, killing giraffes.
- General Motors recalled over 1,000,000 cars for a software change to stop erratic air bag deployment...
What is the course about?

In short...

- Real Time Computing Systems.
  - Key features and issues
  - Modeling, analysis
  - Verification

Broad outline...

- Introduction to Real Time Systems
- Timed Automata
- Verification Tools and Techniques.
- Time-triggered architectures and protocols.
Real Time systems?

What is a Real Time Computer System?

- Correct functioning depends on:
  - the values of the results produced.
  - AND the physical times at which the results are produced.

- Real Time Computer System is embedded in a larger physical system.
Real Time systems?

What is a Real Time Computer System?
- The state of the system evolves with time.
  - Position, velocity, acceleration
  - Pressure, temperature, level, concentration
- The state needs to be sensed and controlled by the computer system...
Real Time systems?

What is a Real Time Computer System?

- Sense
- Computer system
- Plant
- Actuate
Hard Real Time...

Time is important

- System time and external physical time are the same! (At least must have a predictable relationship)
- Computational Timing Constraints: The computing system must sense, compute and actuate in a timely fashion.
- Many sensors and many actuators.
- Many control functions!
- Must schedule computational tasks for different control functions in a timely fashion.
- Computations must finish on time. Performance
Hard Real Time...

Types of real-time systems

- Hard / Soft
  - Hard: Must meet timing constraints always.
    - Reactor control, automotive electronics.
  - Soft: Must meet timing constraints often.
    - Transaction Processing system
    - Multi-media streaming applications.

- Hard/Soft determined externally.
- Our focus will be on Hard Real Time Systems.
Characteristics of Hard Real Time systems...

- Timeliness
- Peak Load Handling  The system should not fail at peak load conditions
- Predictability
- Fault Tolerance
- Maintainability
Hard Real Time...

Impact on system architecture

- Must avoid non-determinism (why?).
- Sources of non-determinism:
  - Direct Memory Access (DMA) by peripheral devices.
  - Contention for system bus.
  - Cache Interrupts generated by I/O devices.
  - Memory Management (paging)
  - Dynamic data structures, recursion, unbounded loops (language level).
Hard Real Time...

Applications for hard real-time systems

- Chemical and Nuclear Plant Control
- Railway Switching Systems
- Automotive applications
- Flight control
- ....
Hard Real Time...

Current state of the art...

- Ad hoc techniques, heuristic approaches.
  - Code written in assembly language
  - Programmed timers
  - Low level device handling
  - Direct manipulation of task and interrupt priorities.

- Goal: Optimized predictable execution on simple architectures.
Drawbacks with this approach

- Tedious programming
  - Code quality depends on the programmer
  - Difficult to understand and maintain.
  - Verification of timing constraints is practically impossible.

- System could collapse in rare, unforeseen circumstances, leading to disasters.
Hard Real Time...

Laws of real-time systems (Buttazo)

- If something can go wrong, it will go wrong. (Murphy’s law)
- Any software bug will tend to maximize damage.
- The worst software bug will be discovered 6 months after the field test.
- A system will stop working at the worst possible time.
- Sooner or later the worst possible combinations of circumstances will occur.
Formal methods - i.e. MATH...

Formal methods in a nutshell

- Common in hardware design buildings, bridges, VHDL chip design and so on.
- It is a common misconception that FM is hard to use and slows things down. Most studies show the reverse is true.
- Design phase slower, implementation and integration faster
- Overall FASTER.
  - CICS, banking, by wire systems.
The idea is to...

- Model real time systems precisely, mathematically, formally
  - External events
  - System events

- Verify timing properties
- Propagate timing constraints down to the system level.
- Verify implementation meets the specification at each level.
Steps towards assurance...

Modelling a system

Computer system

Sense

Actuate

Plant

Model

Specification
Steps towards assurance...

Synthesizing a system

[Diagram showing the process of synthesizing a system, with arrows indicating Sense, Actuate, and Specification connections between Open system, Computer system, and Plant.]
Assurance methods on offer...

In this course at least...

- Transition systems + accepting states = automata
- Timed automata To capture high level descriptions.
- Verification tools and methods
- Time triggered architectures.
- A mix of these two paradigms.
In the analysis of finite state systems...
consider the relationship between a program and its specification as relationships between automata. For such finite state systems, we consider the transitions from each state to another, and there are two ways for describing the behaviour:

1. An (infinite) path of state transitions. A logic that is interpreted on paths is called linear-time temporal logic (LTL).

2. An (infinite) tree, where branches in the tree correspond to (nondeterministic) choices of the system. A logic that is interpreted on these trees is called branching-time temporal logic.
An example of a formalism...

A state transition system ...

A state transition system $\text{TS}$ is a 4-tuple $(S, \text{Act}, \rightarrow, S_\text{in})$, where

1. $S$ is a set of **states**
2. $\text{Act}$ is a set of **actions**
3. $\rightarrow: S \times \text{Act} \times S$ is the **transition relation**
4. $S_\text{in} \subseteq S$ is the set of **initial states**

If you add accepting conditions to a state transition system, then the resultant machine is an automaton.
Example system...

Train, gate, controller
Modelling the system...

Three simple transition systems

- **System 1**
  - States: `open`, `close`, `fin-close`
  - Transitions:
    - `open` to `close`
    - `close` to `fin-close`

- **System 2**
  - States: `approach`, `brake`, `proceed`
  - Transitions:
    - `approach` to `brake`
    - `brake` to `proceed`
    - `proceed` to `approach`

- **System 3**
  - States: `approach`, `left`, `open`, `proceed`, `close`, `fin-close`
  - Transitions:
    - `approach` to `left`
    - `left` to `open`
    - `open` to `proceed`
    - `proceed` to `close`
    - `close` to `fin-close`
Modelling the system...

Signals/Events/actions

- The signals or events (formally, the *actions*) for each of the components of this system are shown in the following table.

<table>
<thead>
<tr>
<th>Direction</th>
<th>Gate</th>
<th>Train</th>
<th>Controller</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>in</strong></td>
<td>open</td>
<td>proceed</td>
<td>approach</td>
</tr>
<tr>
<td></td>
<td>close</td>
<td></td>
<td>fin-close left</td>
</tr>
<tr>
<td><strong>out</strong></td>
<td>fin-close</td>
<td>approach</td>
<td>close</td>
</tr>
<tr>
<td></td>
<td></td>
<td>left brake</td>
<td>proceed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>open</td>
</tr>
</tbody>
</table>
Modelling the system...

Construct a parallel composition ...

Gate

Train

Controller

ParallelTS

Hugh Anderson  Verification of Real Time Systems - CS5270 1st lecture
Modelling timed automata...

Model does not fit in with the reality of the train system; we do not know when to brake!

We would prefer (for example) to be able to specify timing information, and to be able to specify temporal constraints. For example, the following assertions specify some *temporal* constraints over the behaviour of the system:

1. From the time approach is sent, proceed must be received within 5 time units.
2. From the time close is sent, fin-close must be received within 3 time units.
3. From the time close is received it takes at least 2 time units before fin-close is sent out.
Modelling timed automata...

Replace...

... with ...

where the meaning attached is that “From the time \texttt{approach} is sent, \texttt{proceed} must be received within 5 time units”.
Timed automata ...

Timed automata

We label the arcs with three annotations, described below:

1. act is an action (event/signal)
2. $Y$ is a set of clock variables that are reset to 0.
3. $\varphi$ is a clock constraint
   $\varphi ::= x \leq c \mid x \geq c \mid x < c \mid x > c \mid \varphi_1 \land \varphi_2$. 

Hugh Anderson
Verification of Real Time Systems - CS5270 1st lecture
38
Time-Triggered Architectures

- A method for organizing real-time computing systems.
- Main Application domain:
  - Automotive electronics.
  - But also used in AIRBUS 380, ...
    - See: http://www.tttech.com/technology/articles.htm
  - FlexRay is a closely related industry "standard"
    - BMW, Daimler-Benz, Philips Semiconductor, Bosch,
Consider...

an event triggered system
- Timed automata
- CAN (Controller Area Network)
- Meeting of 3 people
  - Everyone speaks whenever he/she has something to say.
  - Must wait for the currently speaker to finish before a new speaker can start.
  - Imagine a meeting of 40 people!
TTA architecture...

Time-triggered

- Every speaker is assigned a predetermined time slot.
- After one round, the speaker gets a slot again.
- Also, a topic-schedule has been worked out in advance.
  - Top1, Top2, Top4 in the first round.
  - Top1, Top3 and Top5 in the second round
  - Top2, Top4 and Top5 in the third round.
- Ensure no one breaks the rules!
TTA architecture...

Each node is a processor ...

- Each node runs some application (App), and
- communicates using Communication Node Interface (CNI),
- which in turn is using a time-triggered protocol (TTP).
Time triggered protocol...

Channels connect the nodes

- Nodes connect to each other via two independent channels.
- The communication subsystem executes a periodic Time Division Multiple Access (TDMA) schedule.
- Read a data frame + state information from CNI (Communication Node Interface) at predetermined fetch instant and deliver to the CNIs of all receiving nodes at predetermined delivery instants.
- All the TTPs in a cluster know the schedule, all nodes of a cluster have the "same" notion of global time.