Models of Computation in Embedded System Design

Overview
- Motivation
- Models of Computation
  - Data Flow
  - Synchronous Models
  - Discrete Event Models
- SDL - Matlab Integration
- Summary
System Model and Architecture

Heterogeneous Task Graph

Heterogeneous Architecture
Dataflow Process Networks

- Networks of actors connected with streams
- Hierarchy of networks
- Communication is buffered with unbounded FIFOs
**Functional Actors**

- No side effects
- For the same input values produce the same output values
  - Functional for each firing cycle
  - Functional over the entire stream
Firing Rules

- Sequential with blocking read
- An actor with $p \geq 1$ input streams can have $N$ firing rules:

$$\mathcal{K} = \{ R_1, R_2, \ldots, R_N \}$$

$$R_i = \{ P_{i,1}, P_{i,2}, \ldots, P_{i,p} \}$$

**Adder:** $R_1 = \{[*],[*]\}$

**Selector:**

- $R_1 = \{[*], \perp,[T]\}$
- $R_2 = \{\perp,[*],[F]\}$

**Nondeterminate Merge:**

- $R_1 = \{[*], \perp\}$
- $R_2 = \{\perp,[*]\}$
Static Data Flow

- The number of tokens consumed and produced by each process is constant.
- A static schedule can always be found, if it exists, and
- The maximum buffer requirements can be computed statically.

Schedule: AABAABCD
Perfect Synchrony

- Perfect synchrony assumption:
  - Computation takes no time
  - Communication takes no time (synchronous broadcast)

<Initialize memory>
foreach period do
  <Read inputs>
  <Compute outputs>
  <Update memory>
end

- Assumption: The system reacts rapidly enough to perceive all external events in suitable order.
Features of Synchronous Languages

- Deterministic
- Amenable to formal analysis
- Efficient synthesis
- Substitution of equivalent blocks preserves behaviour
Substitution of Equivalent Blocks

\[ \begin{array}{c}
\text{P1} \quad \text{P2} \\
\quad \quad = \\
\quad \quad \text{P3}
\end{array} \]
Clocked Synchronous Models

- Computation takes 1 clock cycle
- Communication takes no time
- Substitution of blocks must consider timing behaviour
**Feedback in Synchronous Languages**

- Programs in a synchronous language represent equations.
- Recursive equations may have 0, 1 or more solutions.

- $x = \text{not } x$
- $x = x$
- $x = (x^2 + 1.0)/2.0$
- $u = \text{if } c \text{ then } v \text{ else } w$;
- $v = \text{if } c \text{ then } w \text{ else } u$;
Discrete Event Models

- Event-driven dynamics
- Events:
  - Primary input stimuli
  - Internally generated events
- Events have totally ordered time stamps
- Components have arbitrary delays
- Discrete or continuous time
- Most general timing model
- Primarily targeted to simulation
Simultaneous Events

Δ delay model
Delta Time

The model allows infinite feedback loops between $t$ and $t+1$
EventList

While (event list not empty)
begin
  t = next time in list
  process entries for time t
end
Event Driven Simulation

- Advance simulation time
- Determine Current Events
- Update Value
- Propogate Events
- Evaluate activated elements
- Schedule resulting events

Done

no more events

A → B → C

A → B

\[ t + \Delta \]
Discrete Event Models

- Global event queue is a bottleneck
- Timing model is close to physical time
  - Good to simulate timing behaviour of existing components;
  - Difficult to synthesize
  - Difficult to formally verify

- DE Models are interpreted according to a different timing model: Clocked synchronous model
Heterogeneous System Modelling

- Heterogeneous Systems
- Different Communities of Engineers
- Established Languages with different profiles
- Established design flows
SDL and Matlab

- **SDL**
  - Communicating State Machines
  - Communication is buffered with infinite FIFOs
  - Non-deterministic elements
  - Partially or totally ordered global time
  - Discrete events govern the execution

- **Matlab**
  - Data flow model
  - Demand driven execution
  - Deterministic
  - Partially ordered events; no global time
  - Vector oriented computation
Matlab - SDL Integration: Timing

- Equip Matlab with a timing model with totally ordered events.

\[ r = f(a) \text{ where } a = <a_0, a_1, \ldots, a_n> \text{ and } r = <r_0, r_1, \ldots, r_m> \]

Re-interpretation!
Matlab - SDL: Synchronisation

- Provide a synchronisation mechanism which preserves Matlab’s vector oriented computation
**Composite Signal Flow**

- **Execution Model**
  - Data flow process
  - Processes may have state

- **Signals**
  - Signals are sets of events
  - An event is a \((\text{value}, \text{tag})\) pair
Signals

- Signals
  - Signals are sets of events
  - An event is a (value, tag) pair

- Sampled Signals
  - Values are only defined for tags $t = t_0 + n \lambda$

- Vectorized Signals
  - Event values are vectors of constant length

- Vectorized, sampled signals

![Diagram of signals and events]
Vectorization

- **Head vectorization**

- **Tail vectorization**

Axel Jantsch, KTH, Stockholm
De-Vectorization

\[ \Omega^h_4 \]

Head de-vectorization

\[ \Omega^t_4 \]

Tail de-vectorization
Causality

- A process is causal if for all possible input and output streams two output streams never differ earlier than the corresponding two input streams.

\[
\begin{align*}
  i_1 &= p_1 \alpha p_2 & o_1 &= q_1 \beta q_2 \\
  i_2 &= p_1 \gamma p_3 & o_2 &= q_1 \delta q_3 \\
  \alpha &\neq \gamma & \beta &\neq \delta
\end{align*}
\]

- Tailvectorization is causal

- Head vectorization is not causal

- Tailde-vectorization is not causal

- Head de-vectorization is causal

P is causal if and only if \(\text{tag}(\alpha) \leq \text{tag}(\beta)\)
Causality and Delay Processes

- By combining a non-causal process with a delay process, the resulting compound process can be causal.
- A delay process outputs every input event delayed by a specific time.

\[ (t, v_0) \rightarrow (t+1, v_1) \rightarrow (t+2, v_2) \rightarrow (t+3, v_3) \rightarrow (t+n, v_0) \rightarrow (t+n+1, v_1) \rightarrow (t+n+2, v_2) \rightarrow (t+n+3, v_3) \rightarrow \Delta_n \]
Modelling constraints must ensure that processes have data available when they need it.
Applications

- Co-Modelling of Matlab and SDL
  - Causality constraints imply modelling constraints to safely mix Matlab and SDL processes

- Timing analysis
  - Causality constraints can be interpreted as timing constraints derived from the timing of streams

- Parallel Simulation
  - A partition must be a causal process
  - Only periodic signals may cross partition boundaries
Summary

- Different models of computation continue to coexist
- Heterogeneous system modeling is a necessity
- Short term trend: integration of different models
- Long term trend: development of unifying models