Synchronous Languages—Lecture 5a

Christian Schneider

Christian-Albrechts Universität Kiel
Department of Computer Science
Real-Time Systems and Embedded Systems Group

30 November 2011

Last compiled: December 6, 2011, 17:32 hrs

SCADE / Model Railway

Overview

Introduction
- Dataflow Language Model
- Dataflow Communiction
- Applications

SCADE
- Basics
- SCADE Language

Model railway
- Overview
- Hardware
- Network

Summary

Philosophy of Dataflow Languages

- Drastically different way of looking at computation
- Von Neumann imperative language style: program counter is king
- Dataflow language: movement of data the priority
- Scheduling responsibility of the system, not the programmer

Thanks to Stephen Edwards (`http://www1.cs.columbia.edu/~sedwards/`) for providing material for this lecture

Dataflow Language Model

Processes communicating through FIFO buffers

Dataflow Languages

- Every process runs simultaneously
- Processes can be described with imperative code
- Compute . . . compute . . . receive . . . compute . . . transmit
- Processes can only communicate through buffers

Dataflow Communication

- Communication is *only* through buffers
- Buffers usually treated as unbounded for flexibility
- Sequence of tokens read guaranteed to be the same as the sequence of tokens written
- Destructive read: reading a value from a buffer removes the value
- Much more predictable than shared memory

Dataflow Languages

- Once proposed for general-purpose programming
- Fundamentally concurrent: should map more easily to parallel hardware
- A few lunatics built general-purpose dataflow computers based on this idea
- Largely a failure: memory spaces anathema to the dataflow formalism

Applications of Dataflow

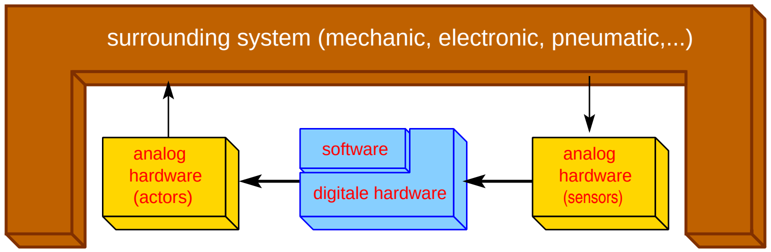
- Not a good fit for, say, a word processor
- Good for signal-processing applications
- Anything that deals with a continuous stream of data
- Becomes easy to parallelize
- Buffers typically used for signal processing applications anyway

Applications of Dataflow

- Perfect fit for block-diagram specifications
  - Circuit diagrams
  - Linear/nonlinear control systems
  - Signal processing
- Suggest dataflow semantics
- Common in Electrical Engineering
- Processes are blocks, connections are buffers

The SCADE language – Context

- Situation of software in an embedded, esp. a reactive, system:



⇒ code is executed in a cyclic manner, event loop is realized by hardware or an operating system

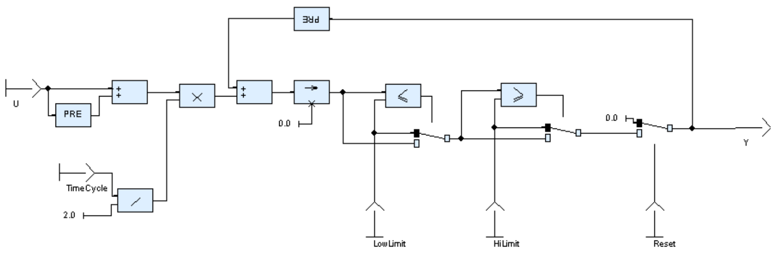
Safety Critical Applications Development Environment®

- The language ...
  - is a graphical dataflow language (with textual backend)
  - adheres to the *Hypothesis of Synchrony*
  - contains built-ins for sequential behavior (state machines)
  - provides a type system
- The tool ...
  - provides code generation + simulation
    - generated code is approved for use in safety critical systems
  - can be connected to external tools (simulation stimuli, simulation data visualization, ...)
  - equipped with further components (verification, UI modeling)

The SCADE language

- A graphical successor of the dataflow language Lustre[2]
- Assignments on memories are understood as equations
  - each memory value must be determined in every computation cycle explicitly
  - each value computation must be realizable by a finite amount of atomic operations (*no unbound loops!*)
  - multiple assignments to a memory in a cycle are rejected at compile time.
  - *more on that later on in a dedicated lecture on dataflow*
- Provides a bunch of basic operators (arithmetic, clocking, data structuring)

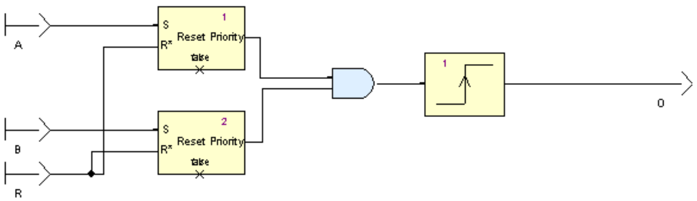
The SCADE language



Implementation of the linear trapezoidal integration in the SCADE standard library.

The SCADE language – Compound operators

- Enable structuring + reuse of particular specifications
- Can be specified...
  - graphically – the typical way
  - textually (in form of SCADE's textual backend)
  - by means of host code functions



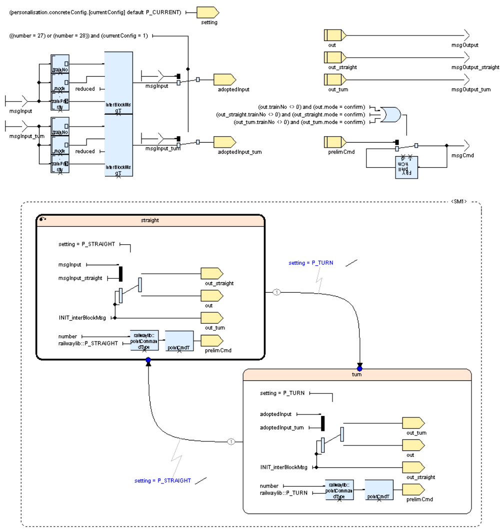
ABRO in SCADE

The SCADE language – Compound operators (cont'd)

Compound operators may ...

- define their interfaces (inputs, outputs, generic constants).
- maintain local variables & signals (act as wires – no memory).
- instantiate other compound & basic operators.
  - Recursive calls are strictly prohibited.
- accommodate State Machines
  - States can be understood as compound operators, as well.
    - may contain State Machines again ⇒ Hierarchy.
  - State Machine dialect is close to André's SyncCharts[1]

The SCADE language – State Machines

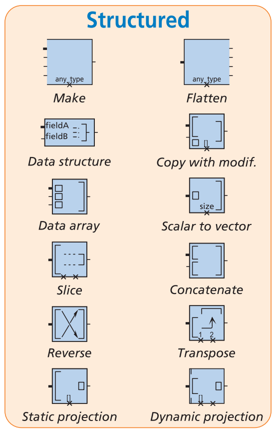


Implementation of the model railway switch controller.

The SCADE language – Further constructs 1/6

- SCADE supports (nested) structs & arrays
- Provides dedicated access/composition/manipulation operators

The SCADE language – Further constructs 2/6

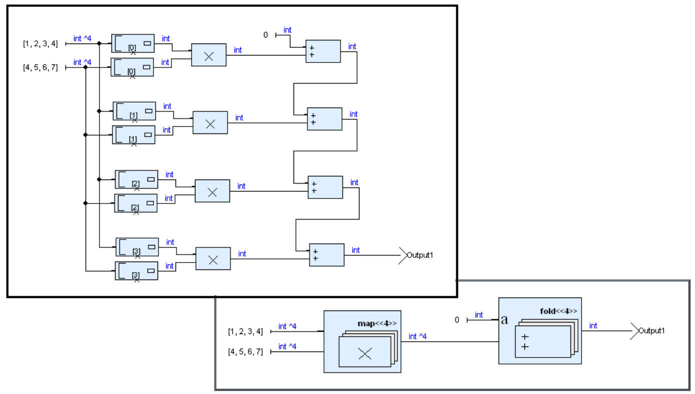


Taken from SCADE Suite reference card:
http://www.esterel-technologies.com/products/scade-suite/modeler

The SCADE language – Further constructs 3/6

- SCADE supports (nested) structs & arrays
- Provides dedicated access/composition/manipulation operators
- Iterator functions: map, mapi, fold, foldi, ...

The SCADE language – Further constructs 4/6

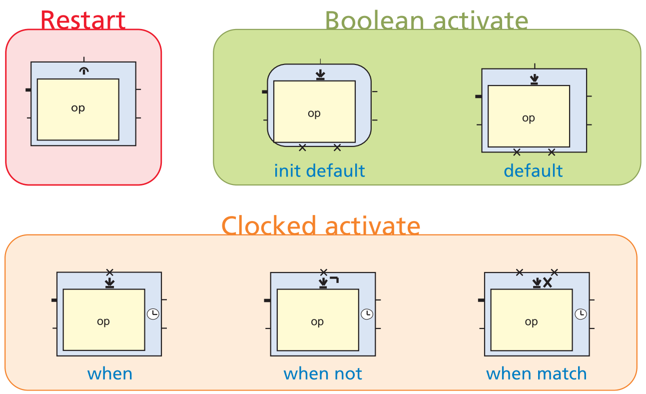


Taken from SCADE Suite website:
http://www.esterel-technologies.com/products/scade-suite/modeler

The SCADE language – Further constructs 5/6

- SCADE supports (nested) structs & arrays
- Provides dedicated access/composition/manipulation operators
- Iterator functions: map, mapi, fold, foldi, ...
- Conditional execution: computation of an operator may be restricted by a guard (initial output values are mandatory)

The SCADE language – Further constructs 6/6



Taken from SCADE Suite reference card:
http://www.esterel-technologies.com/products/scade-suite/modeler

To Go Further

Charles André.
SyncCharts: A visual representation of reactive behaviors.
Technical Report RR 95–52, rev. RR 96–56, I3S, Sophia-Antipolis, France, Rev. April 1996.

Nicolas Halbwachs, Paul Caspi, Pascal Raymond, and Daniel Pilaud.
The synchronous data-flow programming language LUSTRE.
Proceedings of the IEEE, 79(9):1305–1320, September 1991.

- Esterel Technologies
  SCADE Reference card
  `http://www.esterel-technologies.com/files/data-sheets/SCADE-Reference-card.pdf`

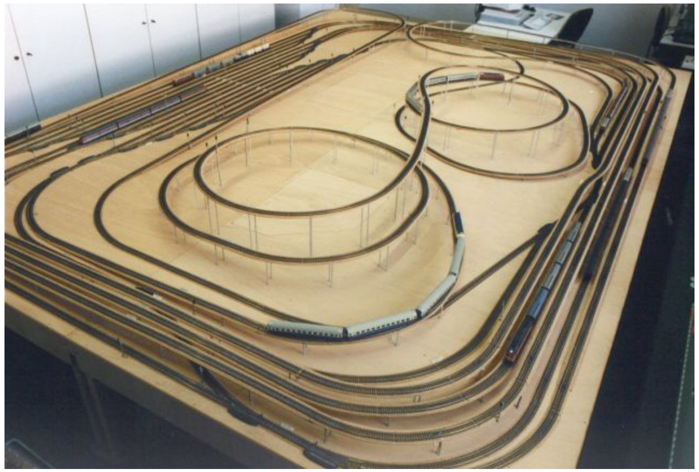
The model railway installation

The model railway installation

- idea came up in 1995 at group of Prof. Kluge
- inspired by a mountain pass in Canada
- has been re-engineered twice
- scale is H0, currently
  - 127 meter lanes (48 blocks)
  - 11 trains
  - 28 switches, 56, signals, 80 contacts
  - 24 lights, ...

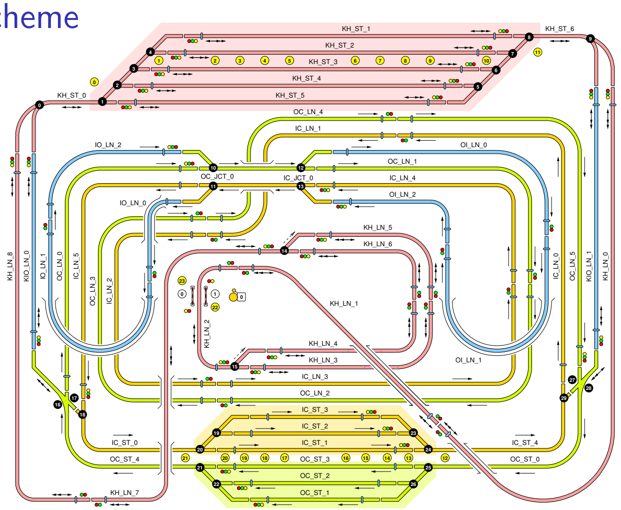
This part of the lecture is based on Stephan Höhrmann's colloquium talk in context of his diploma thesis.

First generation

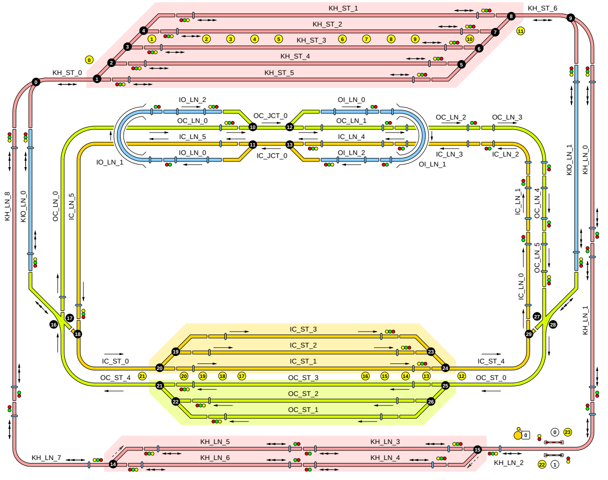


Second generation

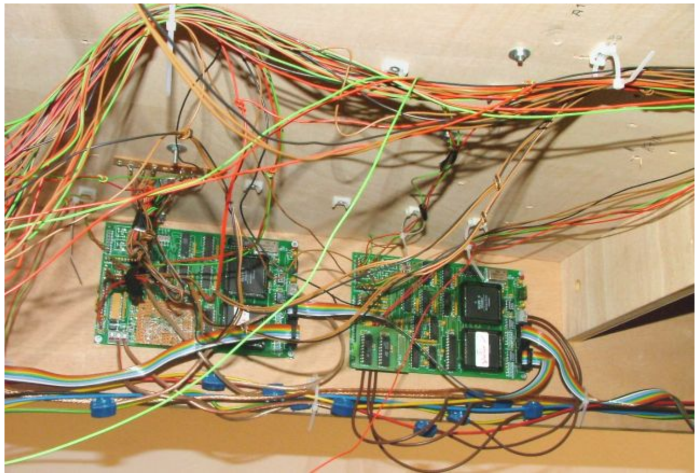
Track scheme



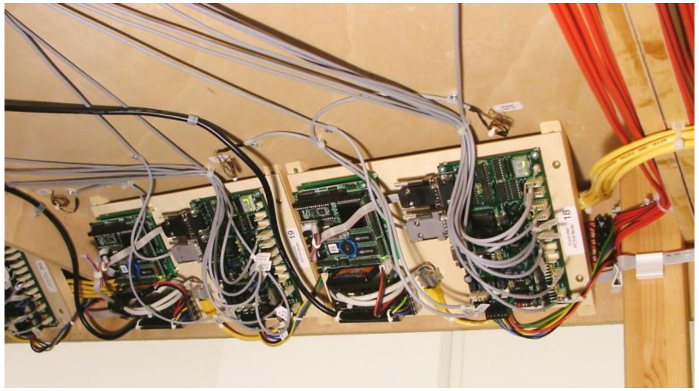
Simplified scheme



Second generation

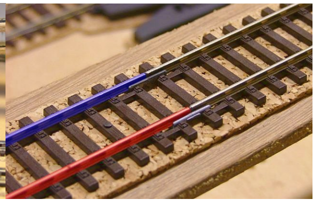


Third generation



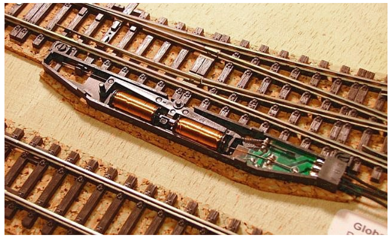
Hardware: Powering

- engines are driven with 12 volt direct current
- direction is according to polarity, speed is determined by PWM
- $\Rightarrow$ track must be separated in blocks

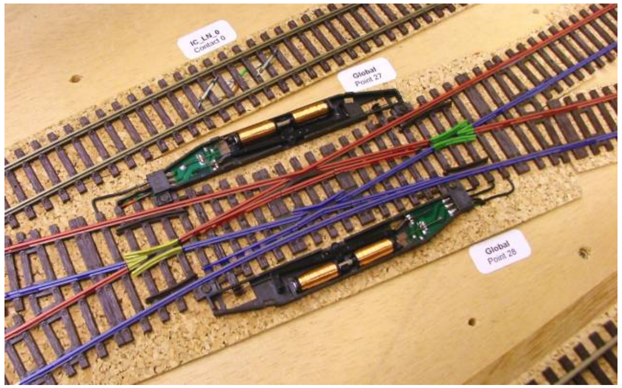
Hardware: Switches

- enable change from main to branch and vice versa
- driven by electromechanical device ⇒ heavy noise



Introduction
SCADE
Model railway
Summary
Overview
Hardware
Network

Centerpieces



Introduction
SCADE
Model railway
Summary
Overview
Hardware
Network

Hardware: Contacts

- recognition of train passages by means or reed contacts
- task: maintaining train positions, stopping in time/place
- installed with redundancy ⇒ direction observation possible

Introduction
SCADE
Model railway
Summary
Overview
Hardware
Network

Hardware: Signals

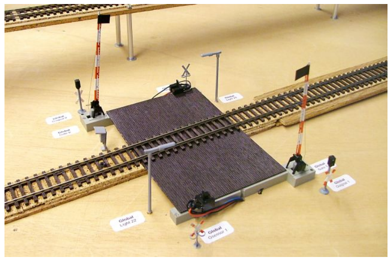
- main signals (red/green) und block signals (+yellow)
- reality, visualizing of the system state
- independent ⇒ are to be controlled by the software

Hardware: Lights

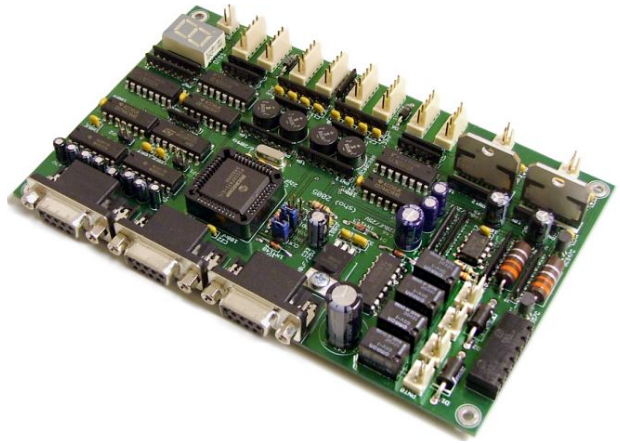
- decoration, highlighting of prominent parts
- can be used for debugging purposes

Hardware: Railroad crossing

- barriers, lights, bell, and sensors
- independent $\Rightarrow$ are to be controlled by the software, as well



Hardware: Power device



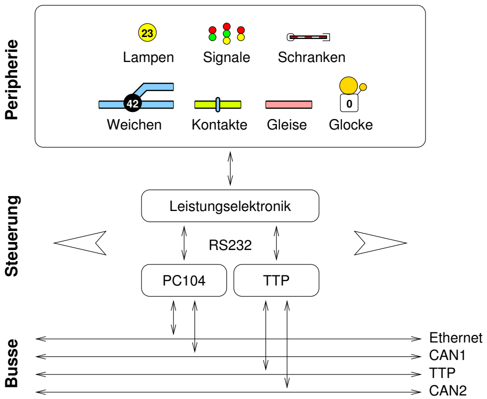
Hardware: Power device

- drives parts of the periphery, boards can be connected to arbitrary bus systems
- signals
  - 4 outputs each driving 3 LEDs
- contacts
  - 4 inputs each observing a pair of reed contacts
  - complex filtering + examination, redundancy management
- track driver
  - 2 short circuit protected outputs supporting forward, backward, brake, and speed regulation by means of PWM
  - integrated occupancy detection, continuous speed control

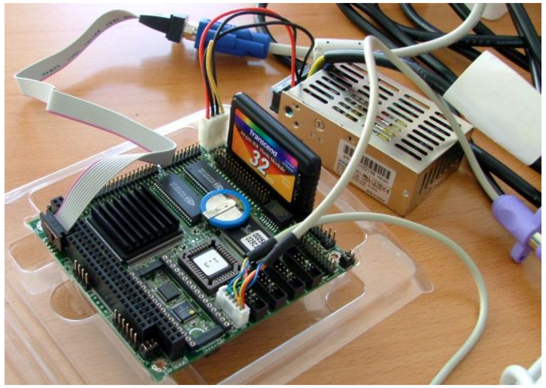
Hardware: Power device

- switches
  - 4 outputs driven by by a separate power source
  - may drive switches, lights, bell, . . .
  - robust wtr. to heavy disturbance
- Serial ports
  - 4 port for connecting with computers
  - UART-based protocol
  - firmware chooses active port, at most 1 active at once
  - 19200 Baud, 8N1, full duplex, cycle of at most 10 ms
- failures
  - EEPROM backups failures of reed contacts, short curcuits, . . .
- display
  - 7 segment display exhibits internal state

Connectivity: Modular concept

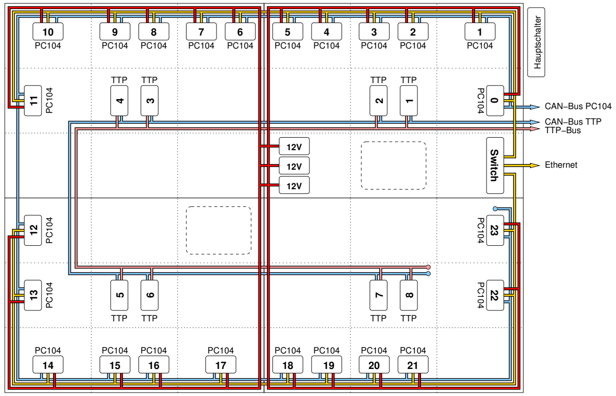


Connectivity: PC104 computers

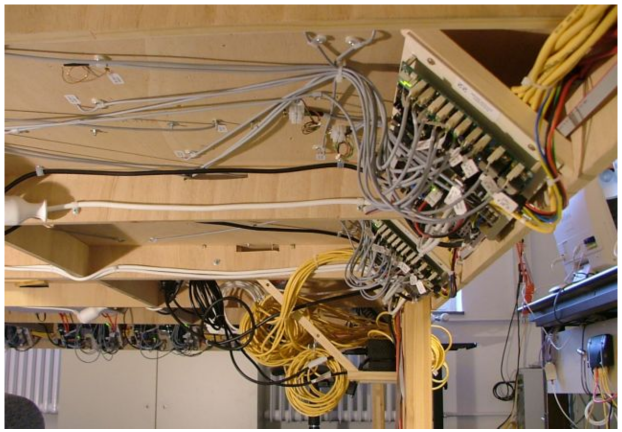


Connectivity: TTP Powernodes

Connectivity: Networking



Connectivity: Wiring/Ethernet



Connectivity: TTP

Summary

- introduction into the dataflow programming paradigm
- "crash course" on the SCADE language
- presentation of our model railway demonstrator