International Workshop on
Applied Reconfigurable Computing
(ARC 2005)

imat

Function replacement of hard real-time systems using partial reconfiguration

Thomas Reinemann, Roland Kasper

Institute of Mechatronics and Drive Technology
Otto-von-Guericke-Universität Magdeburg

Function replacement of hard real-time systems using partial reconfiguration

Contents

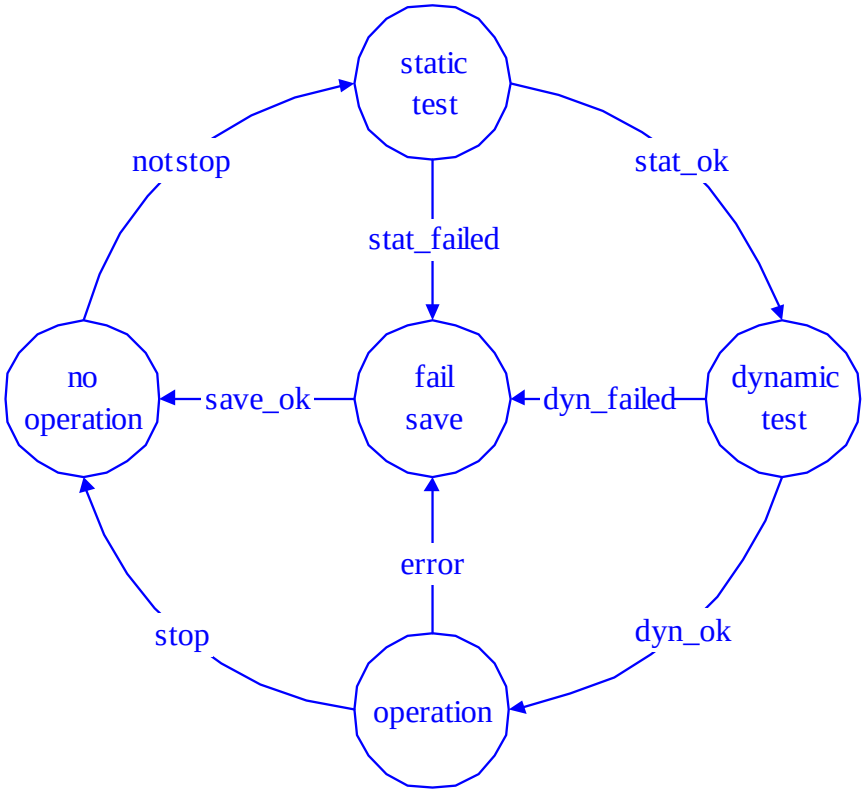
- Mechatronic Systems
- Specification and implementation using FSMs
- Communication System
- Implementation using Virtex II
- Results and prospectus

Structure of Mechatronic Systems

Structure not fix, depends on

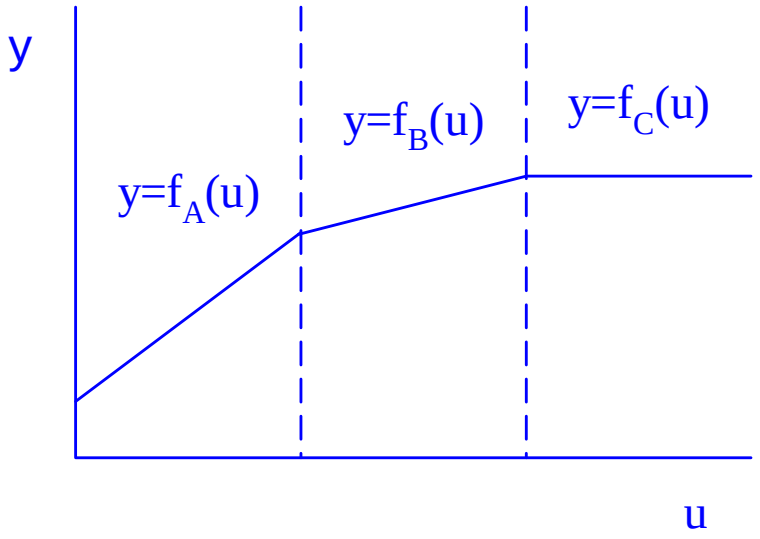
Operating states

- Self-Test, Operation, Fail-Safe
- Charge – Discharge



Operating point

- Revolutions per seconds (engines)
- Temperature
- Voltage



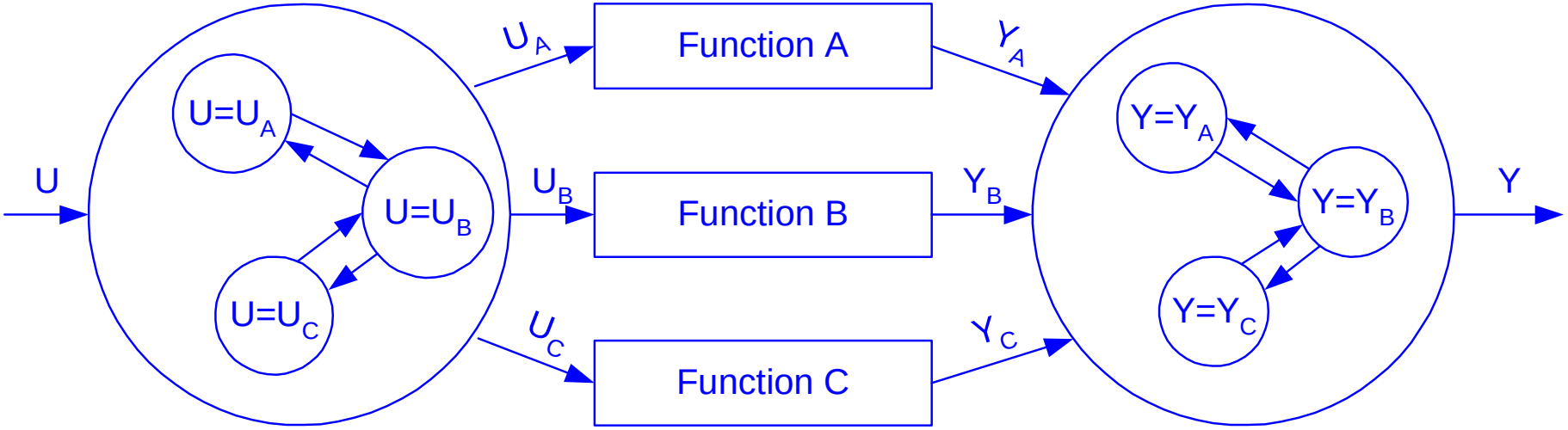
Specification of Controllers

Control Flow

- Activates algorithms
- FSM based
- Controls data flow's structure

Data Flow

- Block diagram based
- Describes signal processing algorithms



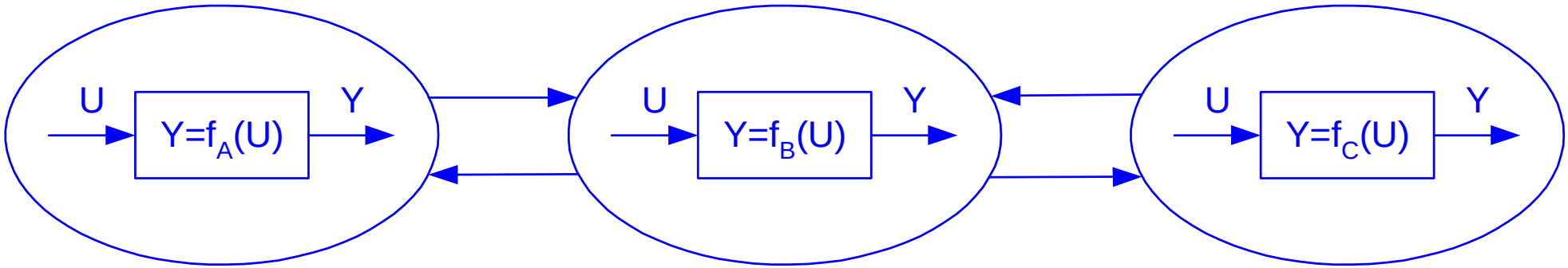
Specification of Controllers

Control Flow

- Activates algorithms
- FSM based
- Controls data flow's structure

Data Flow

- Block diagram based
- Describes signal processing algorithms



Context between Functions, Components, Modules and Types

Function - Software

- Functions express different algorithms
- Function replacement is performed by calling different functions
- During inactivity, functions require only memory

Component - Hardware

- Components express different algorithms
- Multiplexers and tri-state-buffers are used for signal flow switching
- Currently, unused components are only deactivated, but require logic

Module and Type – Partial Reconfiguration

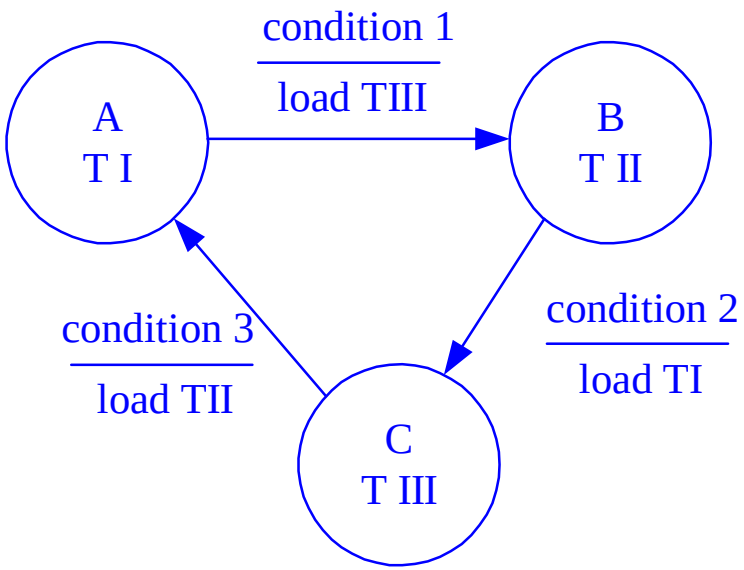
- Module defines the interface
- Types express the algorithm
- Replacing types results in function replacement
- Not loaded types require only memory

Mechatronics and Partial Reconfiguration

- Real-Time conditions
  Sampling time in the range of µs to ms
- Reconfiguration speed
  A sufficient reconfigurable area needs some Milliseconds for reconfiguration. (Xilinx Virtex II)
  Synchronized to sampling period
- Prefetching is needed for high performance applications.

Module FSM

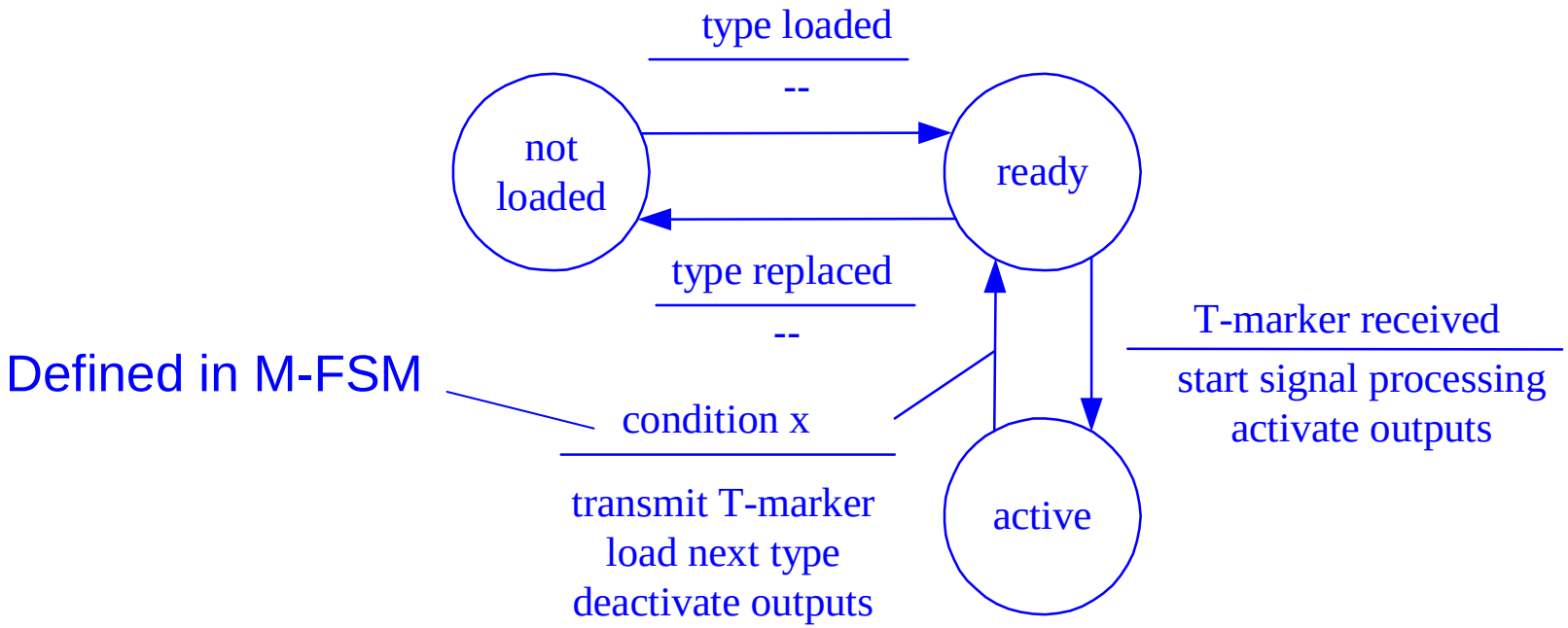
- Defines sequence of type loading and activation
- Results from specification



- To satisfy real time needs, all accessible state (types) have to be prefetched.
- This results in a distributed M-FSM implementation, having multiple types loaded.
- Since only one type is allowed to be active, an activation information has to be exchanged between types.

Type FSM

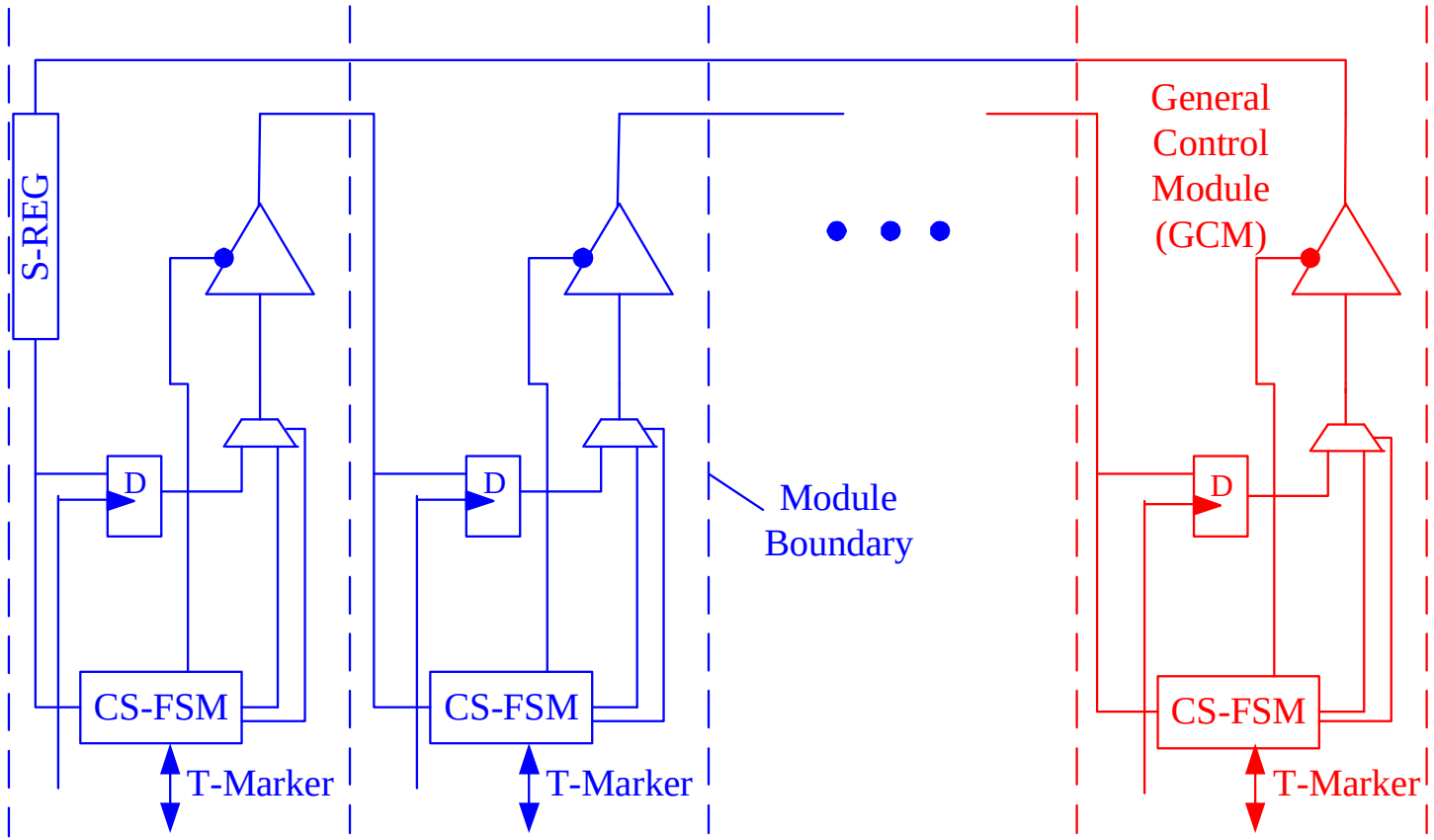
- Controls type activation and deactivation
- Structure identical for all FSMs



T-Marker

- Each type has its own T-Marker for activation
- T-Marker is assigned to its type during compile time

Communication System



- Static, not reconfigurable, bypasses data during slot reconfiguration
- General Control Module (GCM) takes part on communication
- Each subscriber stores one bit at a time
- Deterministic time behaviour

Message Structure

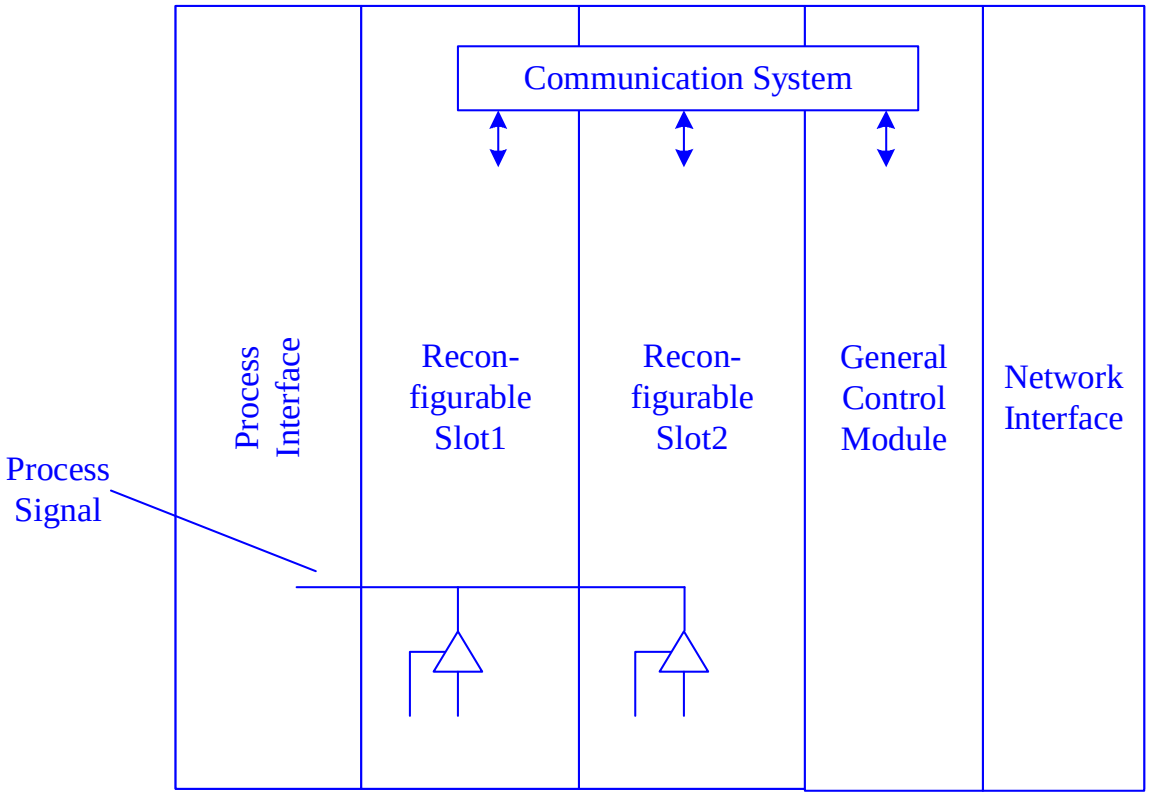
- Start delimiter (0xA5)
- Data Field Used Flag
- Data Field (fixed length), carries T-Marker

Control System

- A bit stream representing a type is assigned to each T-Marker.
- GCM holds a table containing the bit stream’s location and length within memory.
- GCM receives the T-Marker too and dispatches loading a type.

Implementation Example

- Xilinx XC2V1000
- Two reconfigurable slots
- Tri-state buffer is only driven by the active slot, holding a T-Marker



Results

- Controller’s function has been implemented
- Reconfiguration via JTAG
- Reconfiguration via ICAP in progress
- Bit streams are transmitted via USB
- Bit streams are stored in internal memory and transmitted accordingly to ICAP by GCM
- Communication system implemented and operational

Future Work

- Burst mode
- Virtex 4
- Using external memory to store more configurations
- Investigation of alternative communication systems
- More implementation examples:
  - Piezo Controller
  - Rapid prototyping system for ECUs