Dynamic Reconfiguration of Mechatronic Real-Time Systems Based on Configuration State Machines

S. Toscher; R. Kasper; T. Reinemann  
Institute of Mechatronics and Drive Systems  
University of Magdeburg, Germany

A reconfigurable controller for mechatronic control systems

- A new type of adaptable real-time system on FPGAs using dynamic reconfiguration thus reducing the amount of logic resources
- Function replacement realizes switching between operating states of the controller
- Specification is based on signal flow and Finite State Machines (FSMs)
- Distributed FSMs implemented directly in hardware control the reconfiguration
- Data flow implementation makes use of bit serial algorithms
Specification of mechatronic control systems

Data Flow ↔ Control Flow
- specification of data flow: **signal diagrams**
- specification of control flow: **FSMs**
- example of structured control flow specification: run-up and operation FSM
- FSMs can be integrated into data flow and vice versa

- only one active state in a FSM at a given time → good way to determine parts of a complex hardware system that are to be loaded and ready for operation

Implementing dynamic reconfigurable systems

- implementation uses Xilinx Virtex-II FPGA and Xilinx Modular Design methodology
- a function **block** specified by the data flow can be implemented as a **module** in a slot of the FPGA
- blocks are reconfigurable if they offer the same interface → these blocks represent different **types** of the same module
- **FSMs** realize the switching of types because there can be only one active state/type at a given time
**Module-FSM**
- A **Module-FSM** realizes loading of types depending on process events or physical values.
- States of Module-FSMs represent their different types.
- Prefetching of types enables hard real-time operation.
- The structure of a Module-FSM is known at compile time.
- Distributed implementation in the types of a module.

**Type-FSM**
- Transfer of the active state of each Module-FSM from one type to the next active state → global T-Marker for each type comes into operation.
- A **Type-FSM** represents the different states of a type:
  - **Not loaded**: stored in external memory, needs no logic.
  - **Ready**: prefetched, listening on control input, but not processing data and driving outputs.
  - **Active**: loaded, listening on control input, processing data and driving outputs.
- The structure of a Type-FSM is known at compile time.
- Distributed implementation of Module-FSM and Type-FSM packed together in the types of a module.
- A distributed communication system realizes the exchange of the T-Marker between the types/states.
Implementing dynamic reconfigurable systems III

Communication System
- distributed **shift register** and bus macros (long lines)
- all modules build the SR with each storing one bit which is shifted each clock cycle
- local **multiplexers** and registers are implemented inside the bus macros → **guaranteed communication during reconfiguration**

Control System
- the **General Control Module** (GCM) accesses a static table containing all T-Markers and location and size of types in memory when receiving a T-Marker
- the GCM reads the bit stream to be loaded and sends it to the internal configuration access port **ICAP** → T-Marker is forwarded to activate the new type

Results
- implementation and test of a **run-up and feed back control system** for an electro motor using a Xilinx FPGA and development tools
- validation and test of the communication system
- network interface: **Universal Serial Bus** (USB)
- process interface: **Delta-Sigma ADCs**
- **bit serial data processing and transmission** reduces the amount of logic and lines to a minimum → based on a library containing frequently needed processing elements
- test of reconfiguration via JTAG and ICAP

**direct dynamic reconfiguration of mechatronic hard real-time and control systems**
**reduced logic amount enables more complex algorithms on smaller FPGAs**