ROB Complex Organization in the ATM Test-Bed

On behalf of the Saclay Group I. Mandjavidze

CEA Saclay, 91191 Gif-sur-Yvette CEDEX, France

ROB Complex Hardware Organisation and Mapping on Various
Platforms

- ROB Complex Operation
- ROB Complex Software Organisation
- ROBIN API
- Discussion
• ROB Controller is based on either PowerPC VME/CompactPCI SBC, or on PC

• Same network connection serves for L2 / DAQ dataflow, run control and monitoring

• Host memory and ROBINs event memories are directly accessible to NIC
ROB Complex: VME Platform

- CES RIO2 with 300 MHz 604 PowerPC and 1 MByte L2 Cache
- ROBIN: PMC with 33 MHz I960Jx, 512 kByte on-board RAM and PLX9080 PCI bridge
- 155 Mbit/s ATM NIC based on IDT NicStar chip
• CES RIOC with 200 MHz 604 PowerPC and 512 MByte L2 Cache
• Up to 6 ROBIN-s could be grouped in this configuration
• 155 Mbit/s ATM NIC based on IDT NicStar chip
ROB Complex Organisation Pilot Project and DAQ Prototype-1 Common ROB Meeting

Irakli.Mandjavidze@cea.fr 6 / 13 CERN, 4 June 1999

ROB Complex: PC/Linux Platform

- DELL PC with 400 MHz Pentium II
- ROBIN: PMC with 33 MHz I960Jx, 512 kByte on-board RAM and PLX9080 PCI bridge
- 155 Mbit/s ATM NIC based on IDT NicStar chip
- ROB Complex based on a Sharc PMC ROBIN and WinNT has been implemented as well

Irakli.Mandjavidze@cea.fr
**ROB Complex Operation**

- **Integration of ROB Complex into ATM test-bed capable to saturate it**

  **Rob Controller**

  - **ROB\textsubscript{IN}-s**
  - **Sup 0**
  - **Sup 1**
  - **Dst 0**
  - **Dst 1**
  - **Dst 13**
  - **Mon**
  - **ATM Switch**

  **ROB Complex tasks**
  - Receive Request
  - Find concerned ROB\textsubscript{in}-s
  - Pass Request to ROB\textsubscript{in}-s
  - Receive ROB\textsubscript{in} responses
  - Build event fragment
  - Update statistics
  - Send to requester
  - Receive decisions
  - Broadcast to ROB\textsubscript{in}-s
  - Receive Monitor requests
  - Respond to Monitor

- **Start-up from central console via Ethernet UDP/IP application**

  -> Each node reads simple ASCII file that replaces database

- **Subsequent run control, monitoring uses common data and protocol network**

  -> resume, suspend, stop

  -> periodic Echo and Statistics request/responses
Operation of ROB Complex for Data Requests

- Single thread application: No Interrupts, polling
- Buffers can be allocated in ROB Controller: ROBIN DMAs event data - preprocessing
- Buffers can be mapped on ROBINs event memory: NIC DMAs event data
- Capability for sending of chained buffers
ROB Complex Software Organisation

- Host Platform and ROBIN hardware are hidden
- Robin hardware abstraction performs basic operation: Load Code, Start, Reset, Halt...
- Low Level Communication performs message passing and allows for debugging
- Simplified ROBIN Application: no data input
- C code for:
  - Saclay ROBIN, LynxOS / PowerPC VME SBCs, Linux / PC
  - Transtech Sharc PMC, WinNT / PC
Low Level Communications

- **Client**
  - Command Descriptor
  - Command Table
  - Command
  - Free Command Queue
  - Status Queue
  - Get Free Command
  - Fill Command
  - Post Command
  - Check Status

- **Server**
  - Posted Command Queue
  - Get Command
  - Pass it to App or process it
  - Return Status

- Bi-directional communications
- Synchronous and asynchronous commands
- Minimal use of PCI bus for implementation on PLX9080
ROBIN API: Control

• Create resources, set common parameters
  RobIn_Open( void* *robin, RobInParams param )
  -> robin - a handle for future use
  -> param - common parameters such as Id, detector type, detector coverage, code

• Free resources
  RobIn_Close( void* *robin )

• Set ROBIN type specific parameters
  RobIn_Init( void *robin, void spec_param )
  -> spec_param - specific parameters, e.g. selection and EB data size for generic ROBIN

• Hides ROBIN platform
• ROBIN handle allows for multiple ROBINs in ROB Complex
• Monitoring of individual ROBINs to be done
**ROBIN API: Dataflow**

- **Request event data from ROBIN**
  
  ```c
  RobIn_Request( void *robin, RobInReq *request )
  ```
  
  `request` - defines type, event, response location

  ```c
typedef struct __RobInReq {
    RobInReqType req_type;  // RoI, Selection or EB data, clear...
    unsigned int event_id;  // Event ID
    unsigned int local_id;  // Id to assist fragment building in ROB Controller
    BufDesc* *resp_buf;    // Response buffer
  } RobInReq;
  ```

- **Get response from ROBIN**
  
  ```c
  RobIn_Response( void* *robin, unsigned int *local_id )
  ```
  
  `local_id` - local id of corresponding request

- **Clear a list of events from event memory**
  
  ```c
  RobIn_ClearEvent( void* *robin, int *num, unsigned int *events )
  ```

- Allows for ROBIN (one copy) and NIC (zero copy) DMA operation: both implemented
- Local ID mechanism for ROB Complex multi-ROBIN fragment building
- Buffer descriptor allows for chaining; it does not know what information it carries
  
  `Event management is separated from buffer management`
**Discussion**

- **ROB Complexes based on several platforms and ROBIN-s has been tested**
  - VME platform with Lynx PowerPC SBC and Saclay ROBIN
  - CompactPCI platform with Lynx PowerPC SBC and Saclay ROBIN
  - PC platform with Linux and Saclay ROBIN
  - PC platform with WinNT and Transtech Sharc PMC

- **ROB Complexes has been integrated in the ATM test-bed**
  - First measurements show performance limits

- **Modular software has been developed**
  - ROBIN API allows for different modes of operation
  - ROBIN hardware platform is hidden
  - Event and Buffer management is separated
  - ROB Complex software is platform and network technology independent

- **ROBIN Application is still under development**
  - Waiting for the Version 2 of the Saclay ROBIN
  - More realistic event handling will be added