Development of an FPGA-Based Actuator and Sensor Bus Controller for Robotics

The Dxlfpga Architecture

Tobias Alexander Klinke

University of Hamburg
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics
Technical Aspects of Multimodal Systems

10th December 2019

Contents

Typical RoboCup Humanoid Kid Size League Robot

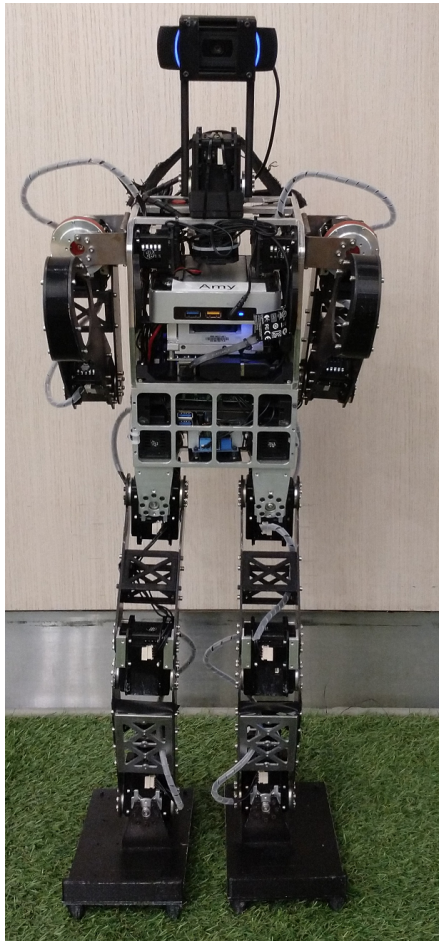


Figure: The Wolfgang robot platform [Rob19a]

- 20 servos
- 1 Inertial Measurement Unit (IMU)
- 1 or more host computers
- connector board

Dynamixel Servos

Figure: A Dynamixel Servo [Rob19c]

- containing ARM Cortex-M3 controller
- asynchronous serial half-duplex communication (8 bit + one stop bit)
- TTL or RS485
- chaining possible
- packet based protocol
- max communication speed 4MBaud

Dynamixel Protocol 2.0 Packet Layout

Field	**Size**	**Value**
Header	3 bytes	0xFF, 0xFF, 0xFD
Reserved	1 byte	0x00
ID	1 byte	{0x00...0xFC, 0xFE}
Length	2 bytes	Little endian unsigned
Instruction	1 byte	Code for instruction
Parameters	Length - 3	Depending on instruction
CRC	2 bytes	CRC16 checksum of packet

Figure: DXL2 packet layout [Rob19b]

Most important instructions:

- `PING`
- `READ` / `WRITE`
- `SYNC_READ` / `SYNC_WRITE`

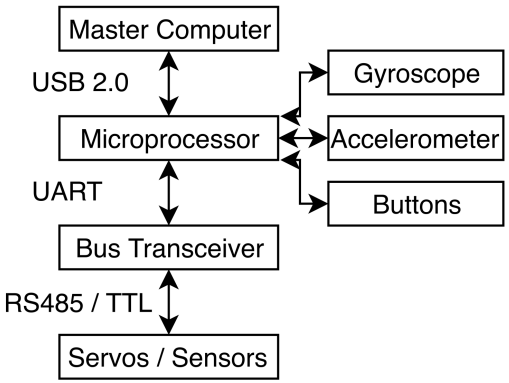
Servo responds with `STATUS` packet depending on instruction type and configuration.

Dynamixel Protocol 2.0 Instructions

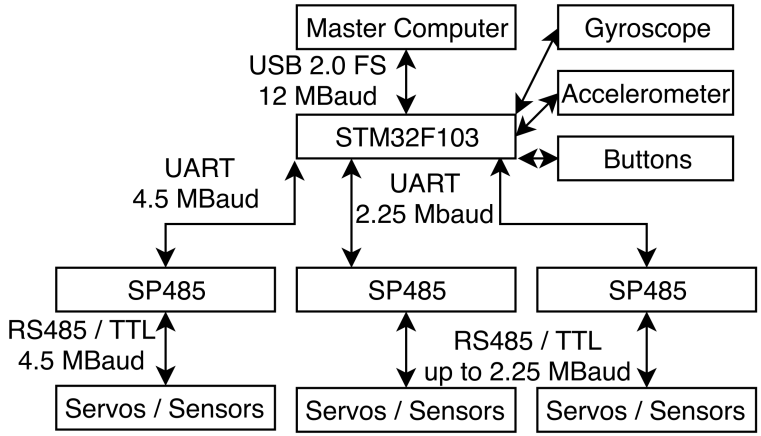
- `PING`
  - Single ping: Check existence of single ID
  - Broadcast ping: Get IDs of all connected devices
- `READ` / `WRITE`
  - Read / write contiguous chunk of control memory
  - Parameters: Start address + Data (length)
- `SYNC_READ` / `SYNC_WRITE`
  - Read / write multiple devices at the same time
  - Parameters:
    - `SYNC_READ`: Start address + Data length + IDS
    - `SYNC_WRITE`: Start address + Data length + IDs + Data

Existing Approaches

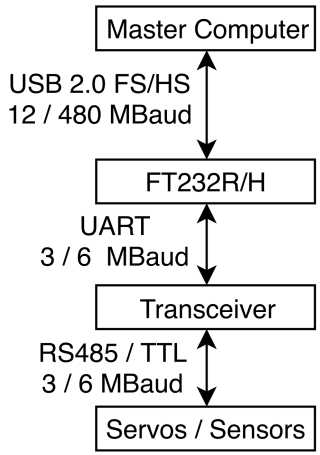
Microprocessor Approach
(CM730, Arbotix, OpenCM, OpenCR)



Multi Channel Microprocessor Approach
(Rhoban DXL)



Single Channel USB to Serial
Approach (USB2DXL / U2D2)



Multi Channel USB to Serial
Approach (QUADDXL)

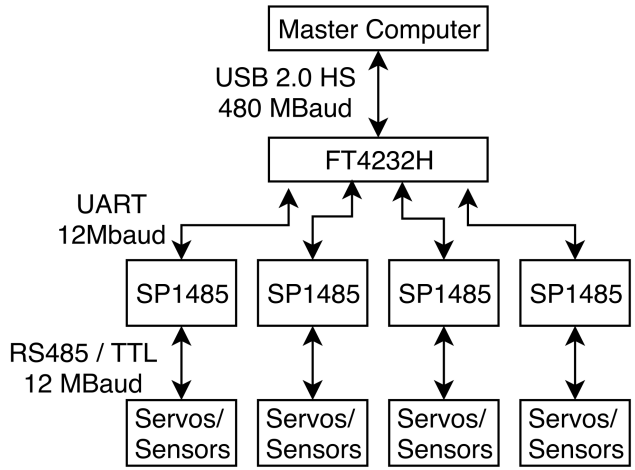


Figure: Block diagrams of different approaches to communicate with servo motors, sensors and other peripheral devices using the Dynamixel bus. [BGZ19]

Design Goals

- Performance: High update rates
- Extendability: Easy to add new sensors / actuators
- Scalability: Scale well in multi-bus setups
- Easy integration: Drop-in replacement for existing controllers
- Dynamixel support: First class Dynamixel servos support

Design Principles

- Modules should be defined that **divide** the whole system **into manageable pieces**.
- Abstractions should be used to **hide implementation details** inside modules.
- Modules should have **well-defined interfaces** for passing data between modules.
- Modules should be **loosely coupled** so that replacing one implementation by another is possible without much influence on other modules.

The Dxlfpga Architecture

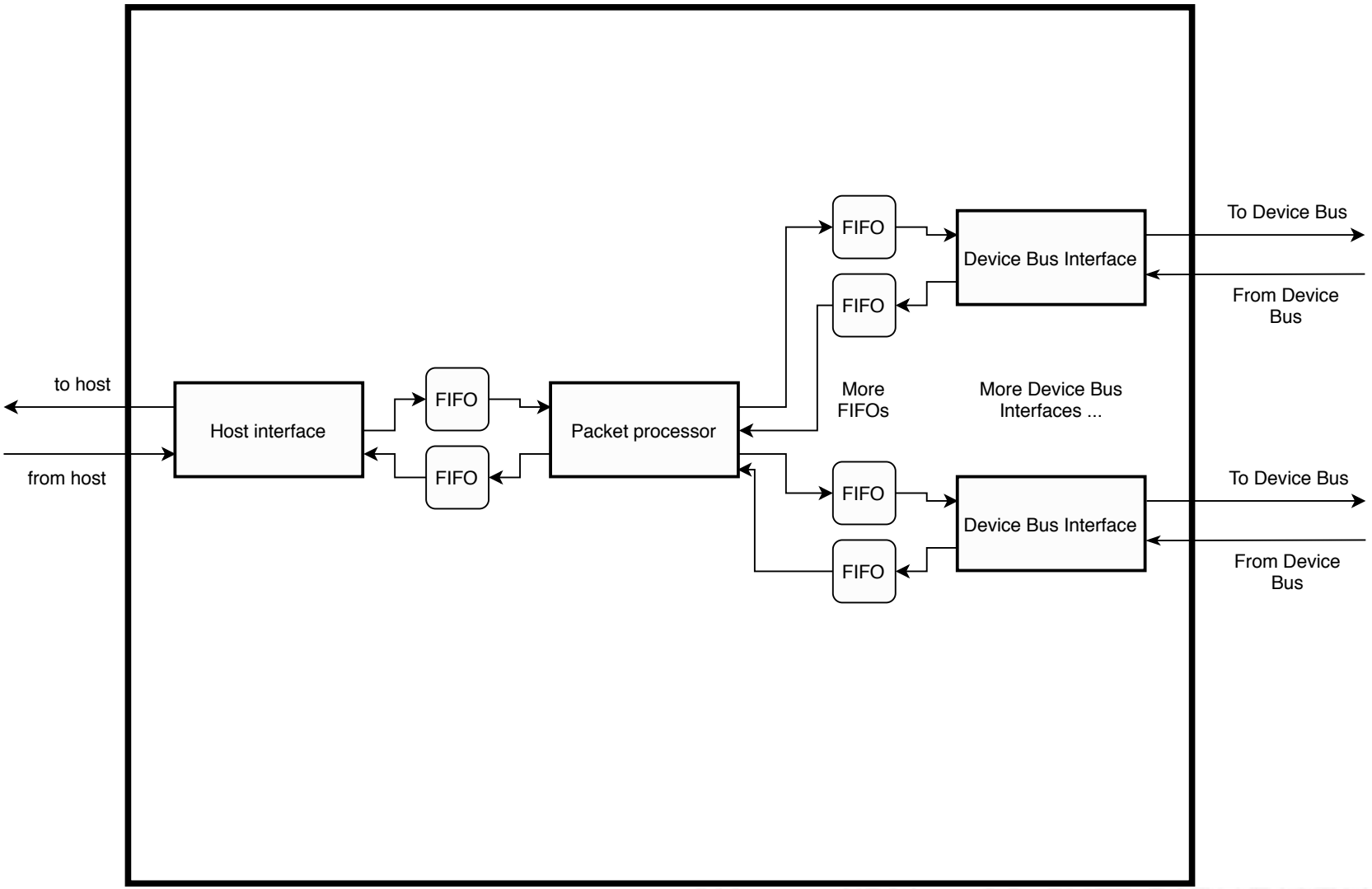


Figure: Block diagram of the Dxlfpga architecture showing the high-level components of the architecture and their connections.

Target Hardware

22,320 logic elements
594Kbit on-chip memory

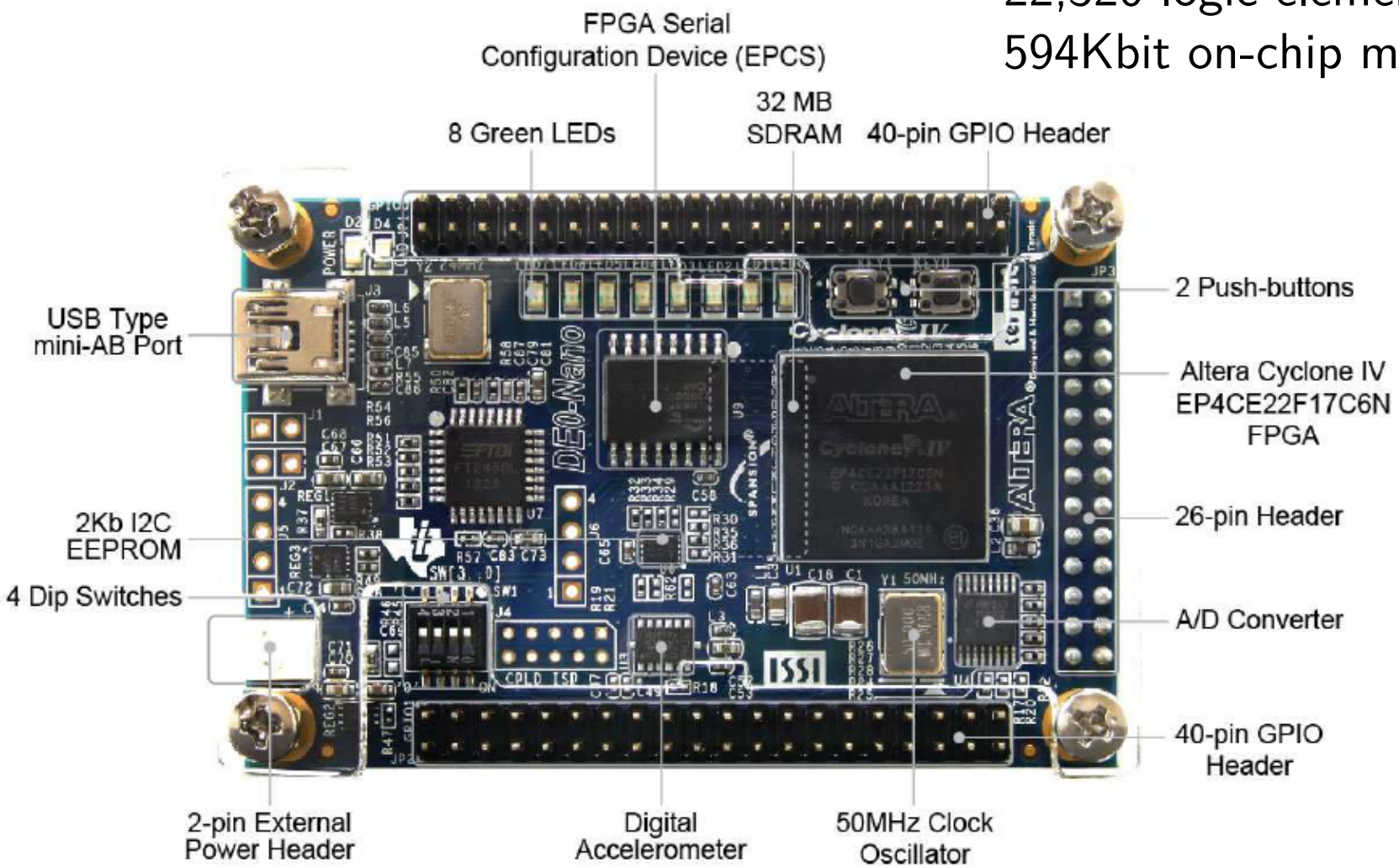
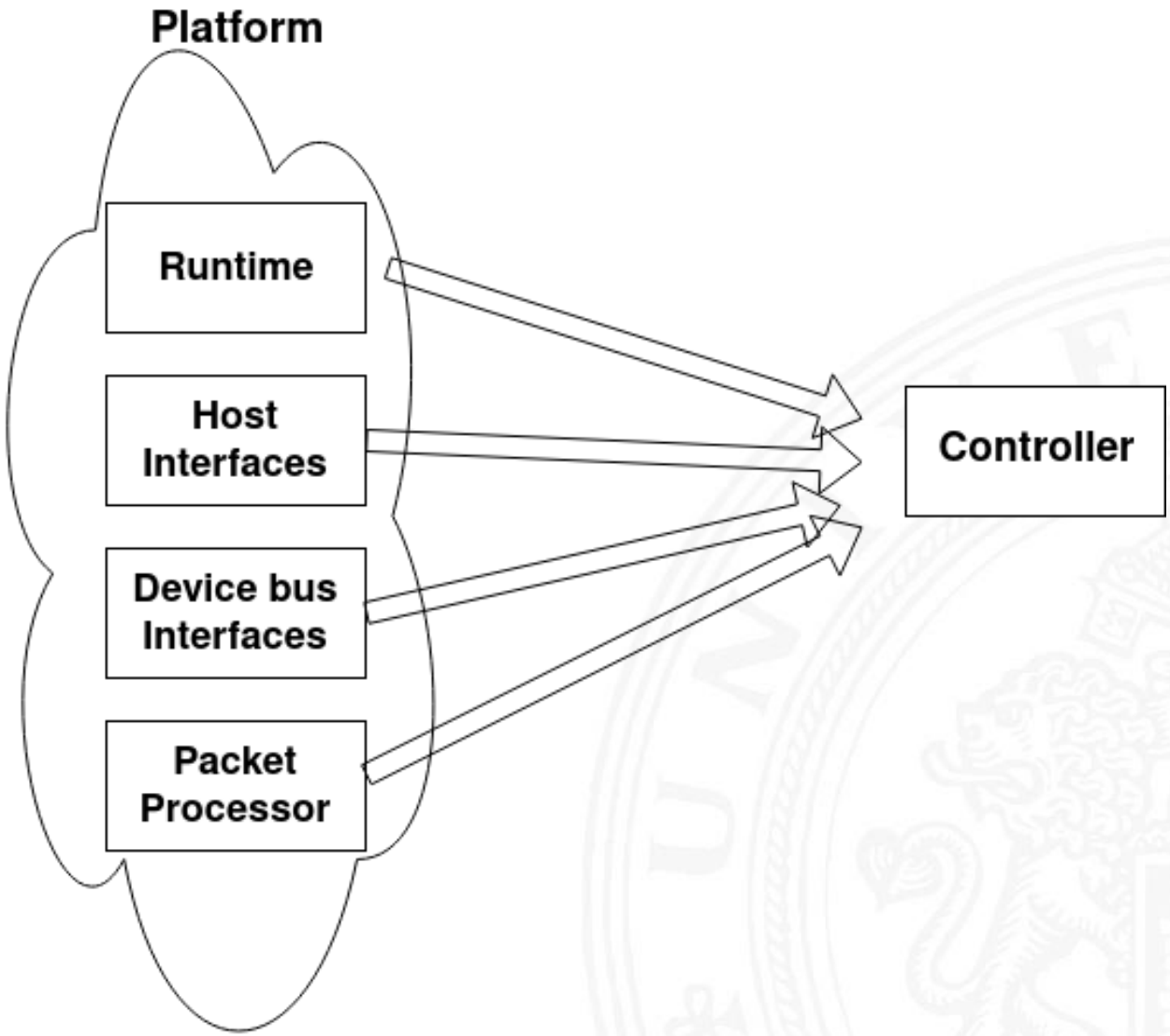


Figure: The DE0-Nano board with Cyclone IV FPGA [Inc13]

Target Hardware

Figure: The FTDI FT2232H Mini Module [Lim19]

Platform Approach



VHDL Implementation

- Written in subset of VHDL-2008
- Simulated using GHDL
- Synthesized using Quartus
- One entity per module (host interface, packet processor, device bus interface)
- Protocol handler as package
- Modules decoupled using FIFOs

The Dxlfpga Architecture

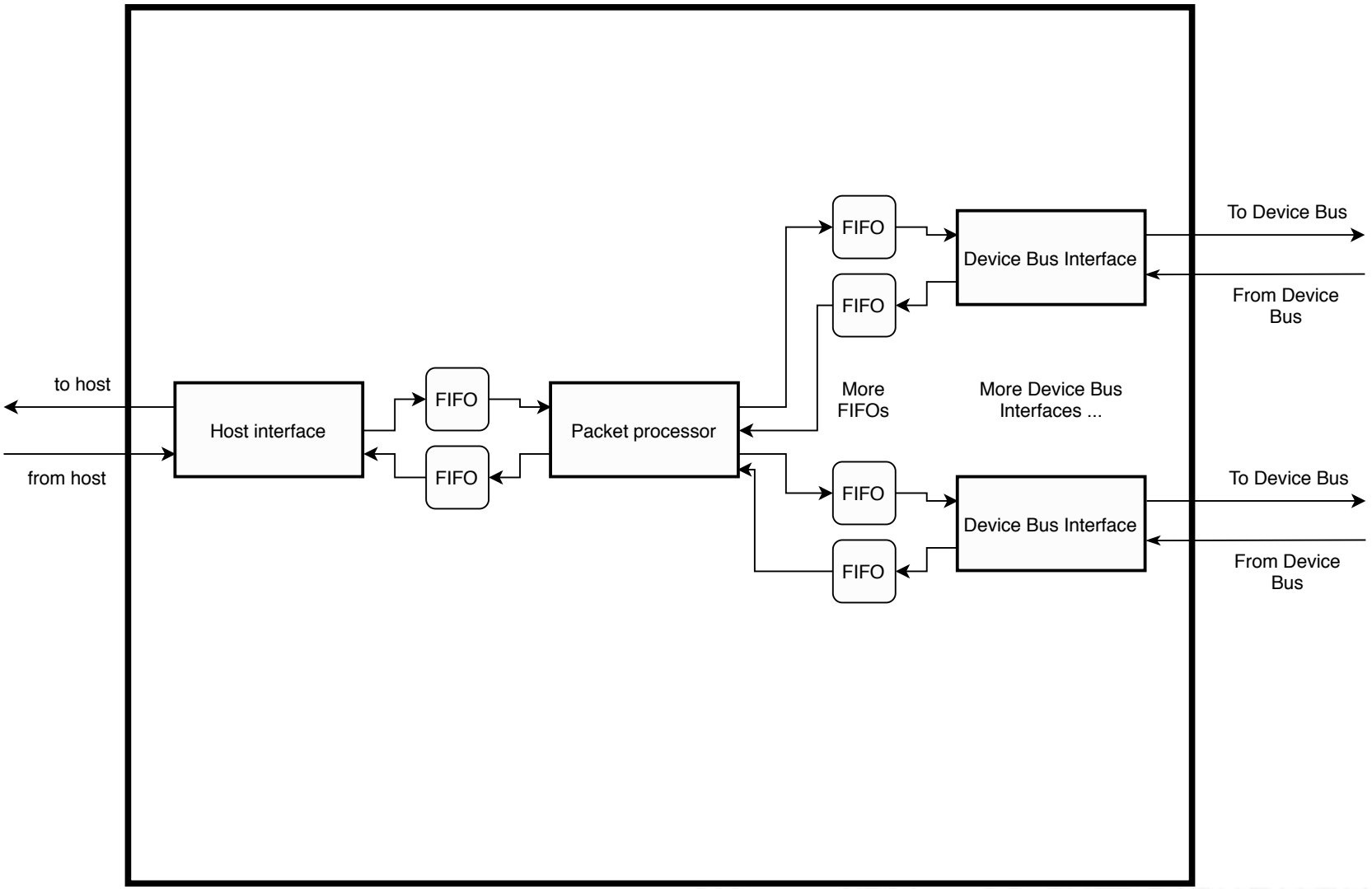


Figure: Block diagram of the Dxlfpga architecture showing the high-level components of the architecture and their connections.

Synchronous FIFO Implementation


```
-- a synchronous FIFO.
entity sync_fifo is
  generic (
      -- The maximum number of elements in the FIFO.
      constant DEPTH: positive;
      -- The size of one FIFO element
      constant ELEMENT_SIZE: positive
  );
  port (
      clk: in std_logic;
      rst: in std_logic;
      write_enable: in std_logic;
      data_write:
          in std_logic_vector(ELEMENT_SIZE-1 downto 0);
      read_enable: in std_logic;
      data_read:
          out std_logic_vector(ELEMENT_SIZE-1 downto 0);

      -- status flags
      full: out std_logic;
      empty: out std_logic
  );
end entity;
```

Host Interface Implementation

FTDI Asynchronous FIFO interface:

- `dbus`: 8 bit parallel half duplex data
- `txe`: FTDI accepts data to be sent
- `wr`: Send data from `dbus` to host
- `rxf`: Indicate whether data has been received
- `rd`: Put next received byte onto `dbus`

Outgoing packets have higher priority!

Packet Processor Implementation

Processing steps for a packet:

1. Analyze packet to decide type of packet and destination
2. Distribute packet to its target device bus interfaces
   ⇒ Distributor
3. Collect response packets from the device bus interfaces
   ⇒ Collector
4. Pass response packets to host interface

Packet Processor Implementation

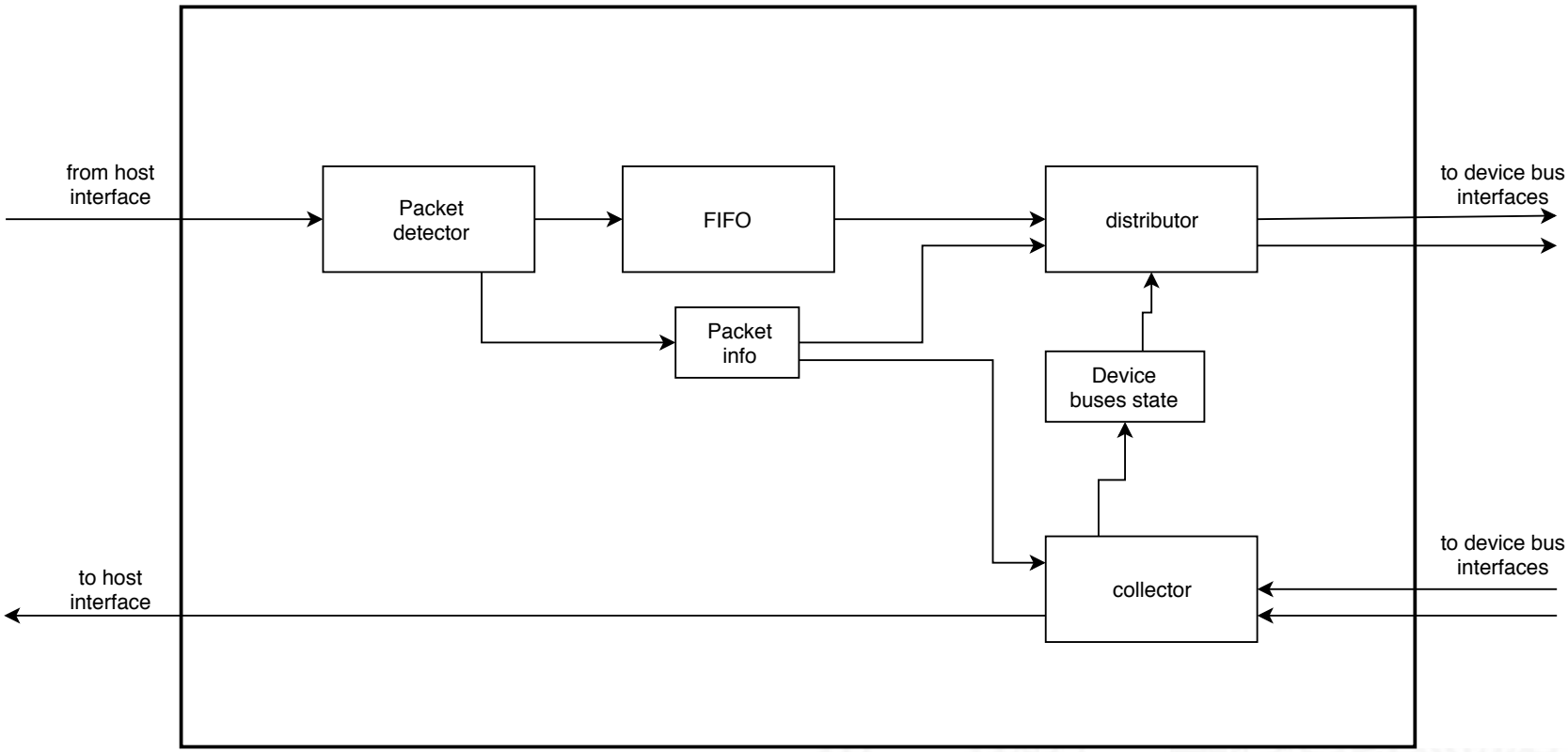


Figure: Block diagram of the packet processor implementation showing the most important components.

Tooling

- Build system (based on SCons): Build controller from YAML description
- Unit tests with VUnit + GHDL
- System Test Simulations: VUnit/C++/Python framework

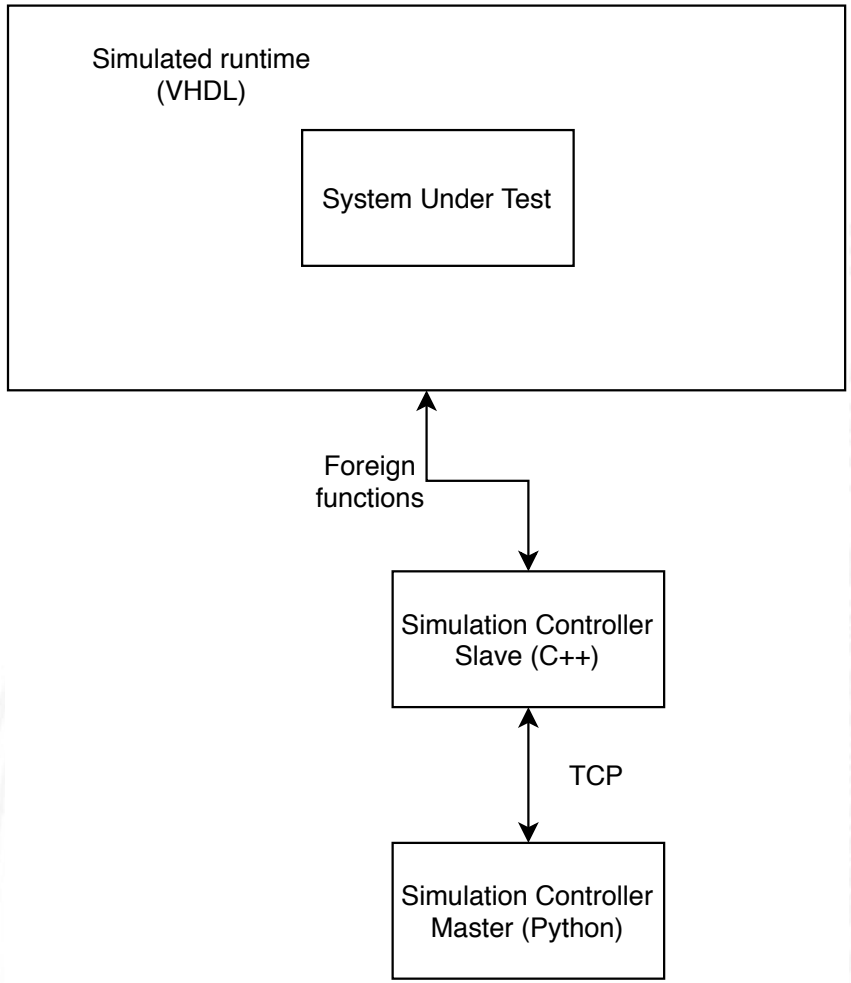


Figure: Communication of the components in system test simulations

Performance

Table: Mean update rates with SYNC_READ and SYNC_WRITE commands

No. buses	MBaud	Board	Update Rate (Hz)
1	1	Theoretical Max.	176
		R-DXL Single	132
		R-DXL Multi	125
		USB2DXL	153
		QUADDXL	149
		Dxlfpga	- [1]
	2	Theoretical Max.	352
		R-DXL Single	215
		R-DXL Multi	179
		USB2DXL	272
		QUADDXL	261
		Dxlfpga	**276**
	4	Theoretical Max.	704
		R-DXL Single	40
		QUADDXL	398
		Dxlfpga	**425**

Performance (cont.)

Table: Mean update rates with SYNC_READ and SYNC_WRITE commands

No. buses	MBaud	Board	Update Rate (Hz)
2	1	Theoretical Max.	336
		R-DXL Multi	185
		QUADDXL	285
		Dxlfpga	**277**
	2	Theoretical Max.	671
		R-DXL Multi	219
		QUADDXL	497
		Dxlfpga	**474**
	4	Theoretical Max.	1342
		QUADDXL	744
		Dxlfpga	**678**

Performance (cont.)

Table: Mean update rates with SYNC_READ and SYNC_WRITE commands

No. buses	MBaud	Board	Update Rate (Hz)
3	1	Theoretical Max.	461
		R-DXL Multi	224
		QUADDXL	390
		Dxlfpga	**365**
	2	Theoretical Max.	922
		R-DXL Multi	250
		QUADDXL	670
		Dxlfpga	**618**
	4	Theoretical Max.	1843
		QUADDXL	1003
		Dxlfpga	**837**

Performance (cont.)

Table: Mean update rates with SYNC_READ and SYNC_WRITE commands

No. buses	MBaud	Board	Update Rate (Hz)
4	1	Theoretical Max.	613
		QUADDXL	524
		Dxlfpga	**465**
	2	Theoretical Max.	1227
		QUADDXL	923
		Dxlfpga	**678**
	4	Theoretical Max.	2545
		QUADDXL	1373
		Dxlfpga	**889**

Resource Usage

- Logic elements:
  - de0_nano_ftdi_async_4dxl: 42%
  - per Dynamixel bus: 8%
- Block memory:
  - de0_nano_ftdi_async_4dxl: 13%
  - per Dynamixel bus: 3%
- Clock speed: 100Mhz

Conclusion

- Faster than microprocessor based approach
- Nearly as fast as serial converter based approach
- As flexible and extendable as microprocessor based approach
- Scales well for multiple buses

Future Work

- Design custom PCB
  - integrate into robot
  - FTDI synchronous FIFO interface
- FX3 USB controller as host interface
- Complete Dxl2 protocol support
  - `BULK_READ`, `BULK_WRITE`, `REG_READ`, `REG_WRITE`, `ACTION`
- Optimize implementation
  - Timing: reach 200MHz
  - Porting to other FPGAs
  - FIFO handling: introduce almost full/empty flags
- Finish IMU support
  - Try different IMU handling methods
  - `SYNC_READ` and `SYNC_WRITE` for multiple IMUs
- Support more actuators and sensors

Demo

Demo

References

[BGZ19] Marc Bestmann, Jasper Güldenstein, and Jianwei Zhang.

High-frequency multi bus servo and sensor communication using the dynamixel protocol.
In *RoboCup 2019: Robot World Cup XXIII*. Springer, 2019.
Accepted.

[Inc13] Terasic Technologies Inc.
DE0-Nano User Manual, 2013.

[Lim19] Future Technology Devices International Limited.
FT2232H datasheet, 2019.

References (cont.)

[Rob19a] Teams RoboCup.
Team description papers Robocup 2019 Humanoid KidSize league.
https://humanoid.robocup.org/hl-2019/teams/, 2019.
[Online; accessed 07-October-2019].

[Rob19b] Robotis.
E-Manual.
https://emanual.robotis.com/, 2019.
[Online; accessed 08-October-2019].

References (cont.)

[Rob19c] Robotis.
MX-106T / MX-106R (2.0) documentation.
http://support.robotis.com/en/product/actuator/dynamixel/mx_series/mx-106.htm, 2019.
[Online; accessed 03-December-2019].