FDL, a Deterministic 100 Mbytes/sec Data Link

J. Bovier, J-F. Gilot
Creative Electronic Systems

Abstract

Data transfers in real time applications needs more than just a speed increase to meet the new constraints of data acquisition and control systems. Data gathering from multiple sources, labelling, transfer scheduling and data scattering to multiple destinations are of primary importance.

Over the past few years the power of processing units used in real time data acquisition and control systems has increased by a factor of 100 (from 1 mips to 100 mips) thanks to the availability of RISC-based microcontrollers with clock rates well above 100 MHz. Instructions are now executed in less than 10 ns if cache memory is used. In order to make the best use of this power, external agents are needed to move data to and from processors' memories.

Intelligent I/O boards have been provided to relieve the processing units of direct instrument control and data transfer protocol management. They pack data in buffers ready to be grabbed for processing. The missing piece in this scheme is an autonomous system able to gather data from I/O boards and scatter it to processors' memories at speeds matching those of the processors.

The Fast Data Link (FDL) is a set of hardware and software tools designed to transfer data "intelligently" between multiple sources and destinations.

The backbone of the FDL is a multimaster/multidrop (up to 15 nodes) copper link synchronised at 50 MHz. The physical support is a cable composed of 25 twisted pairs (16 data - 2 parity - 7 control signals) extending over a maximum distance of 30 meters.

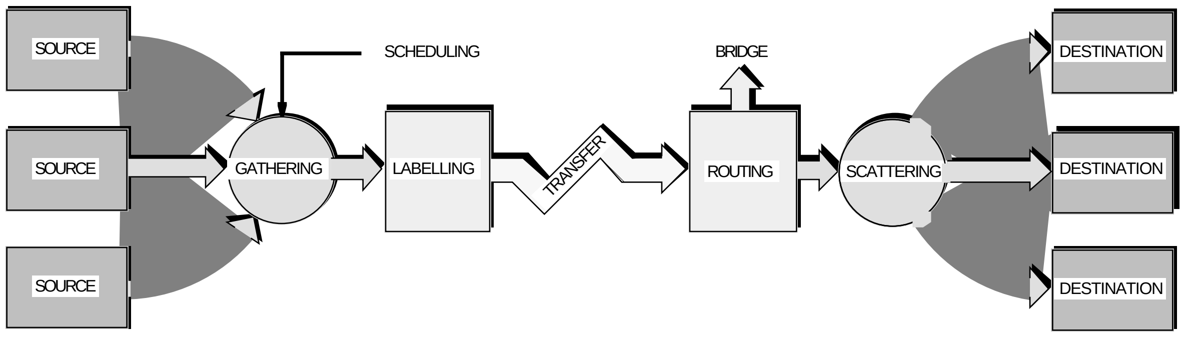

Figure 1. FDL General Concept

The client/server approach has been chosen in designing FDL interfaces. Upon request of a client or occurrence of an external event, the interface gathers data from data acquisition busses, dual ported memories or registers and stores it locally in an intermediate buffer. Information is then added to the data packet before transfer over the link so that each data packet contains routing parameters. In the destination nodes data packets are routed to their final destinations.

A point-to-point fibre optic link has been designed to interconnect at full speed local FDL networks over long distances. Up to 15 optical connections can be implemented from a local network.

Information is transmitted over the FDL in units of "cells". A cell is a sequence of 18 words (of 16 bits) emitted by the current master every 20 ns.

The link is source-synchronised. The current master drives the clock signal and sends one data word every clock cycle. The cell synchronisation signal indicates to the slave nodes the beginning of an 18 word cell. The first two words of every cell contain routing information (destination node identifier, cell identifier, destination buffer identifier, etc.) They are generated by the FDL master port and decoded by the FDL slave port. The payload of a cell is 32 bytes.

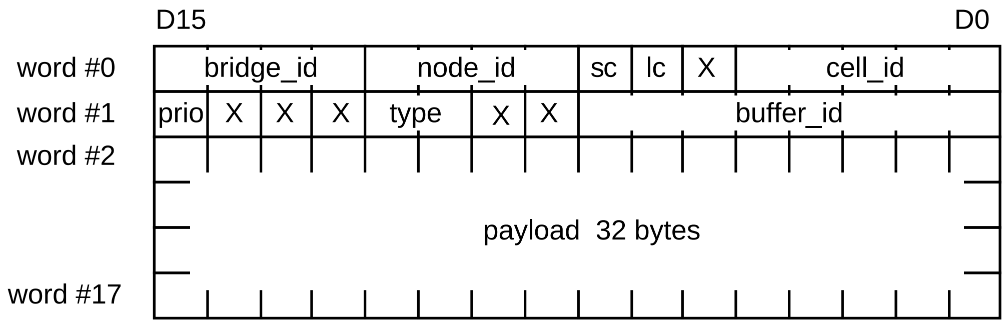

Figure 2. Cell Structure

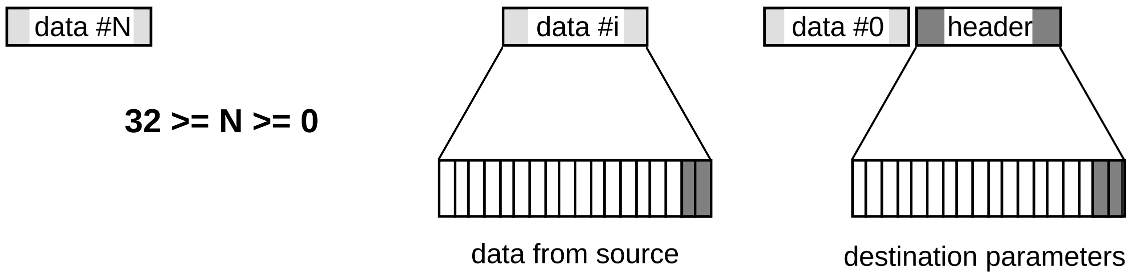

Figure 3. Data Packet Structure

For data transfer, cells are organised in packets of up to 1 kbyte (32 cells maximum) preceded by a header cell containing parameters for the final destination. Thus routing and administrative information imposes overhead of 14%.

Single cells are used for FDL management. The FDL offers two levels of priority for cell transmission to allow management cells to take precedence over data cells. When no node has cells to transmit, the current FDL master transmits empty cells to maintain link synchronisation.

The FDL is a deterministic link. The arbitration mechanism guarantees a defined data transfer rate and the maximum latency for bus mastership attribution.

To offer fair access to the link, bus arbitration is assumed by the current master. This implements a rotating arbitration. A requesting node asserts the Bus Request signal at the beginning of a cell; the current master grants the bus at the beginning of the next cell. This allows the next requesting node in the arbitration daisy chain to own link mastership before the end of the second cell following the Bus Request assertion. This mechanism fixes the latency for a change of master at 720 ns + time of flight between current master and requester (40-150 ns).

latency: < 1 µs

In addition the link shares ownership fairly amongst requesting nodes, thus guaranteeing minimum latency for data transfer completion. For a fully loaded link the overhead for arbitration is one time of flight every two cells. Depending on the physical length of the link the average arbitration overhead is then between 6 and 20%.

The nominal transfer rate of the link is 100 Mbyte/sec x (1.0-Overhead fraction). The average transfer rate is the nominal transfer rate divided by the average number of requesting nodes.

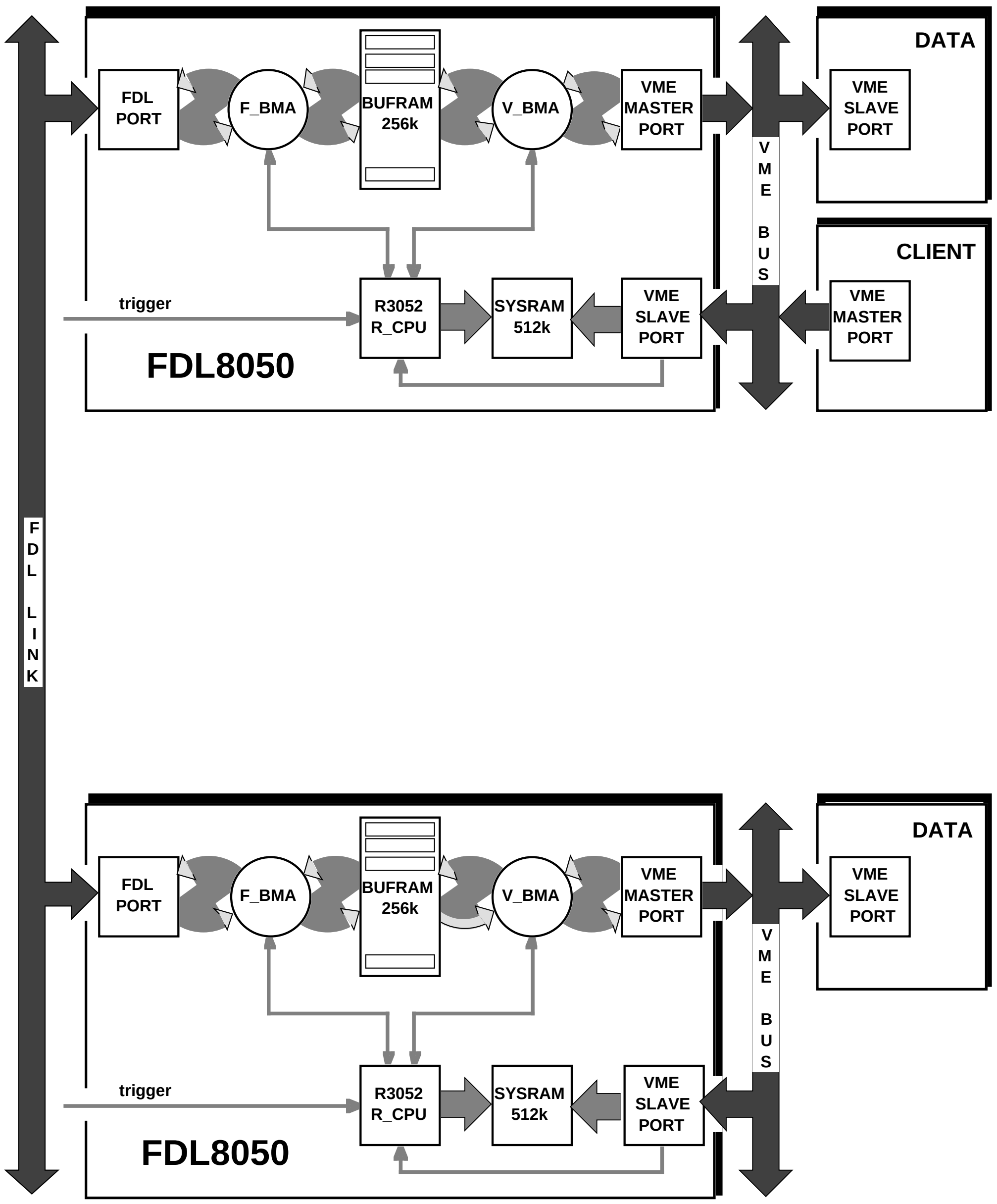

Figure 4. FDL8050 VME interface

The FDL8050 interface is a VME server designed to transfer data between VME crates at rates up to 70 Mbyte/sec. A client is any VME master able to access the FDL8050's VME slave port.

The central part of the VME/FDL interface is a fast dual ported memory (BUFRAM) accessed by two block movers at 200 Mbytes/sec. The first one (V_BMA) gathers/scatters data from/to the VME bus. It packs the data which it reads into blocks up to 1 kbyte in size. The second one moves data packets between the BUFRAM and the link. It handles link protocol and packet fragmentation into cells.

The two block movers are controlled by a RISC microprocessor (R_CPU (mips R3052)) managing data transfer and communication with clients. The system memory (SYSRAM) is accessible from both the V_BMA and the VME bus. This feature is used to implement a 64 kbyte mirror memory. External trigger lines are connected to the microprocessor. They are used for event generation.

The transmission mechanism uses request fifos (of 256 words) to keep both BMAs busy with data packets for transfer.

To trigger a V_BMA data transfer, the R_CPU prepares a transfer descriptor in the SYSRAM and then posts the descriptor index in the V_BMA request fifo. The V_BMA is built around an R3051 microprocessor in order to handle complex VME transfer descriptors. It checks for block boundaries and recovers from bus errors in less than 1 µs. At the end of the transfer the V_BMA updates the descriptor with the transfer status and writes the descriptor index into the V_BMA status fifo. If required, control information such as sequence number, time stamp, etc. is added to the data packet.

The R_CPU builds the header cell with final routing parameters in the HEADRAM, and posts the cell index in the F_BMA request fifo. The F_BMA then transfers the header cell followed by the associated data cells over the link. At the end of the transfer the cell index is written in the F_BMA status fifo.

The destination node catches cells containing its own node identifier and stores them in the BUFRAM. The HEADRAM holds the header cell and the F_BMA status fifo the cell index. The R_CPU reads the header cell and builds a transfer descriptor for the V_BMA. It then triggers the V_BMA to transfer the data packet from BUFRAM to VME. A single cell containing an acknowledge from the data packet transfer is sent back to the transmitter for flow control.

A 12-stage transmission pipeline has been implemented to maximise data throughput while controlling arrival of data packets. Up to 8 VME transfers can be concurrently active.

High priority single cells (service cells) are transmitted over the link for system management; they take precedence over data cells for link access A dedicated F_BMA request fifo is used to post these cells, the HEADRAM contains room for 256 of them, which are directly handled by the R_CPU. These service cells are used for data transfer acknowledge, remote request, event generation and time synchronisation.

The FDL supports time stamping of data packets and transfer synchronisation (isochronous mode). To this end every FDL interface has a hardware clock running at 1 MHz. The interface maintains a calendar time accessible by VME clients. One node in an FDL system keeps the global time reference and every 10 ms it broadcasts this reference value to other nodes. Time synchronisation service cells are directly handled by the interface hardware.

Special timers (25 MHz) on FDL interfaces allow a time synchronisation of all nodes with an accuracy of 1 µs.

The FDL supports two scheduling modes for data transfers. In the asynchronous mode they are triggered by an external event (client request, VME interrupt, front panel trigger, etc.) and are handled by the server on a first-in first-out basis. In the isochronous mode data transfers are activated periodically at a user-chosen frequency. The two modes of data transfers can be mixed.

When the isochronous phase occurs, all asynchronous data transfers are suspended and a pre-defined transfer descriptor is given to the R_CPU for processing. At the end of the isochronous phase (the duration is programmable) asynchronous data transfers are resumed. A special F_BMA request fifo is used to post isochronous data transfers in order to allow instant switching between isochronous and asynchronous modes.

The isochronous mode has been implemented to allow all FDL interfaces to activate special data transfers periodically at the same time. This ensures the time coherence of the data collected by the system.

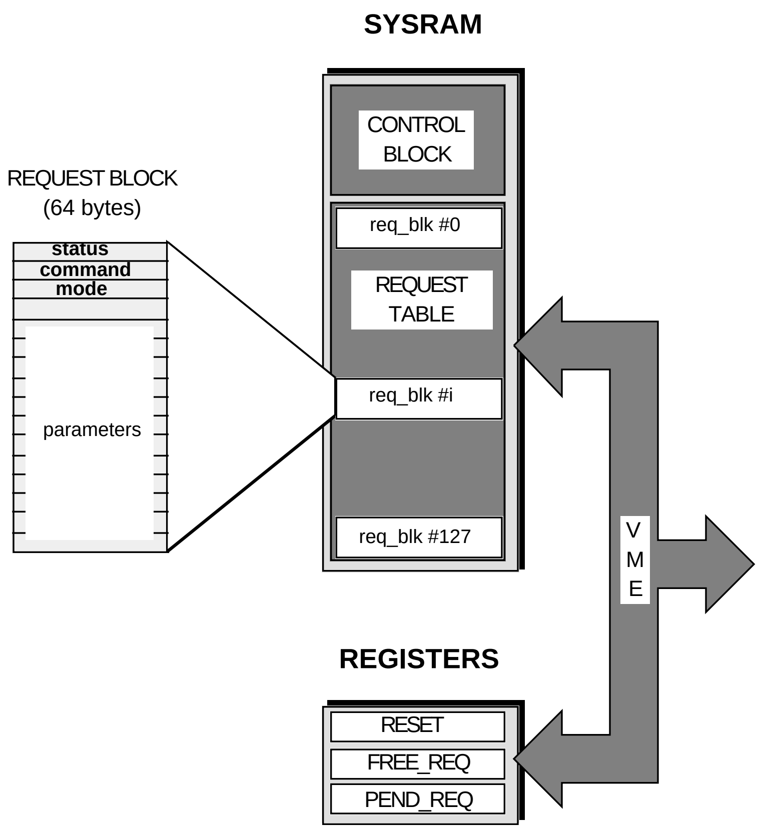

Figure 5. FDL8050 programmer's interface

The VME clients interact with the FDL interface through a shared data structure located in the SYSRAM. Two hardware registers (FREE_REQ and PEND_REQ) are used to control access to that structure. Remote reset of the interface is activated by writing in the reset register. Up to 127 connection channels can be opened at a time to these VME clients. Each communication channel is controlled by a 64 byte request block in which the client posts the transfer parameters.

The FDL firmware supports direct and indirect data transfers. For direct data transfers the client provides all routing parameters in the request block. In the indirect mode, the client provides the reference of a transfer descriptor. Transfer descriptors are linked lists of elementary transfer parameters (chained list of buffers, direct rings, indirect rings, etc.). These transfer descriptors can be pre-defined through a resource creation mechanism, or can be read by the FDL at transfer time. The FDL firmware supports the creation of remote transfer descriptors in order to handle indirect transfers in destinations.

The server supports 8 levels of priority for client requests and handles programmable time-out. Broadcast to multiple VME destinations has been emulated in order to implement protocols such as TCP-IP.

Device drivers for the most popular operating systems have been written to control the FDL8050 interface. If the driver is linked with the TCP-IP layer, a workstation can be used as a gateway to pass TCP-IP packets over the FDL. This allows a VME multiprocessor system to be controlled from a local area network while fast data transfers are going on between VME crates.

Interfaces to other busses such PCI, CAMAC, FASTBUS, etc. are under development in order to build complete data acquisition and control systems. We feel that PCI will be a standard in future workstations. The PCI/FDL interface will be used to push data directly into workstation memories under control of the operating system, so the data will be directly available to data processing software.

We will also provide embedded FDL interfaces to be directly implemented in instruments and detectors in which a register or a dual-ported memory will be used to post data for transfer over the link.

The performance of a real time system depends mainly on three parameters: the processing power, the interrupt response and the data transfer rate. The only way to maximise all these parameters simultaneously is to make them independent by dedicating specialised hardware to each of them. The FDL has been designed to maximise data transfer without compromising processing power or interrupt response.