Futurebus+ Offers Innovative Protocols

Silicon Support Emerging from Three Vendors

By John Theus
TheUs Group, Sherwood, OR

In the previous two articles of this series (see µPR 5/27/92, p. 17 and 6/17/92, p. 17), we took a high-level look at Futurebus+ and its protocols and profiles. In the first part of this final article, some of the more significant Futurebus+ technologies—packet mode, cache coherency, and BTL—are described in greater detail. The last part of this article reviews the announced transceiver and protocol silicon.

Packet Mode

The Futurebus 1987 standard provided a compelled data transfer mode where every word transferred required a handshake between the master and slaves. While this protocol makes for very reliable transfers and is technology independent, it is also slower than required for many of today's applications. The development of a non-compelled protocol went through many stages of design and review—including its final name of packet mode. All of the approaches had a common feature in that they were source synchronous—the sender of the data provided the synchronization. The first proposal added a clocking signal to send with the data. While this approach eliminated the delays due to the handshake, analysis showed that performance was still inadequate.

An improvement on the first proposal was to use a clock per byte. This approach limited the skew to a single byte-wide transceiver chip, but it added lots of new signals and was a solution based on the day's technology. The next stage looked at embedding a clock in each bit in the data stream. While this approach is common on serial links with long blocks of data, the amount of logic to do this on a parallel bus along with the problem of extracting the clock when a data block is only a couple of words long made it impractical.

The breakthrough was realizing that it is not necessary to send a clock at all. The everyday example is a UART. As long as both ends know the clocking rate, start and stop bits are sufficient control information to transfer a stream of data. To solve the clock rate problem, a system's boot software uses the compelled data transfer mode to configure the hardware. Two clock speeds are allowed on an operating backplane, and a choice is made on a transaction-by-transaction basis. A slow speed is chosen that is the lowest common speed, and a higher speed can be selected for a subset of the modules. Modules must provide an internal clock with 0.01% accuracy.

A packet is transmitted by clocking a normal register at the agreed-upon frequency. The data is preceded by a start bit and is encoded as NRZI. The data is followed by a stop bit that returns the signal to its quiescent state and acts as a longitudinal parity bit. The transmitter is allowed to introduce skew across the word and edge-to-edge jitter into the data stream up to specified maximums. Packets are limited in length to 64 words so that the error introduced by the difference in internal clock frequencies does not become a problem.

The front end of the receiver treats each signal that makes up the data word independently. The start bit on each signal is phase compared with the internal clock and the difference recorded. This difference is then used to set the sampling window for the subsequent data bits. When all the bits across a word have arrived, the word is clocked out of the receiver into a normal register.

Although an individual packet is limited in size, multiple packets can be sent during a single transaction. The main restriction is that packets must occupy an ascending contiguous address space. When a packet transfer is started, the compelled data handshake signals are used for controlling the request and approved/denied protocol for the subsequent packets. This protocol runs asynchronously with respect to the packet transfer and is allowed to run ahead so it can build up a queue of packets to send. This multiple-packet-mode protocol can be used for both cached and non-cached data.

Hierarchical Cache Coherency

When the parallel protocols for Futurebus were being designed in the mid-1980s, copyback (writeback) cache coherency was not a design goal. A year later, cache coherency became a major issue, and the decision was made to add it. However, its late addition into the parallel protocols resulted in several kludges. Futurebus+ was designed from day one to support cache coherency, and therefore it has a cleanly-integrated set of coherency protocols.

Although that first implementation left a lot to be desired, the underlying foundation—laid in 1986—was a major advancement in understanding cache coherency. Prior to this work, there were many competing proposals for coherency protocols, and they all looked very different from one another. The major discovery

was that all these protocols were built around a common model—a fact which had been obscured by the use of inconsistent nomenclature.

The model, called MOESI (for the states: Modified, Owned, Exclusive, Shared, and Invalid) describes the attributes (valid vs. invalid, exclusive vs. shared, and modified vs. unmodified), states, and permitted state transitions for a cache line (see µPR 6/20/90, p. 12). Many different cache coherency protocols are supported by the model, all of which are coherent with one another. (Note that some manufacturers have misappropriated the term MOESI to label their own five-state coherency protocols.)

With the model in hand, a Futurebus five-state protocol for a single-bus system was developed that utilized the new parallel protocols called intervention and reflection. Intervention occurs when a cache holding a modified line intercepts a read request to memory, and the cache supplies the line itself. The memory is not involved in the data transfer protocol. Reflection is intervention with the additional requirement that the memory participate in the data transfer protocol and receive and store the modified line.

Futurebus+ built upon the work done for Futurebus by integrating and extending the coherency protocol to work across a hierarchy of buses. The parallel protocol supports split transactions, which are necessary for efficient communication across bus bridges. Intervention was retained, but expanded to work with split transactions. Reflection—which was difficult to implement—was replaced by a more general protocol called snarfing.

Snarfing lets a bystander (neither the master or the addressed slave) grab a transaction's data as it passes by. An example would be a cache that is waiting to win the bus so it can request a line read. While waiting, the cache is snooping bus traffic. If the cache sees the address corresponding to its pending read, the cache snarfs the data and retires its read request. While on the surface this seems like a rare occurrence, around hot spots, such as semaphores, snarfing can save significant bus bandwidth.

Considerable work went into finding a protocol within the MOESI model that was appropriate for a hierarchy of caches across bus bridges. The resulting four-state write-invalidate protocol is called MESI. The owned (shared modified) state, write broadcast, and partial-line writes were all excluded to reduce complexity and lower cost. The exclusive unmodified state is limited to single-bus systems so that bridges can always see the transition to the modified state.

BTL

Backplane Transceiver Logic was the first significant technology developed to solve a Futurebus need. Before BTL, the only TTL-compatible transceivers were either 74LS245-class devices, which had insufficient drive for backplane impedances, or 26S12-class devices, which could sink sufficient current to drive the center of a 50-ohm line but when used at every slot in a backplane dropped the impedance of the line below 25 ohms. The end result was that neither could provide incident wave switching—cleanly passing through the threshold on the first signal edge, and never reentering the threshold region due to subsequent reflections.

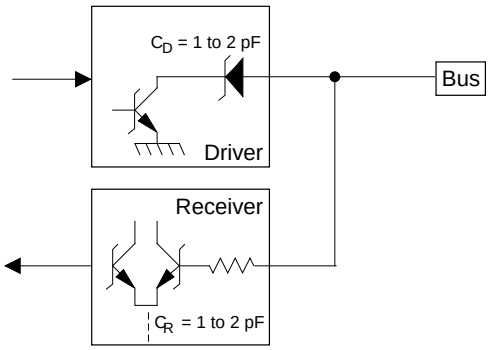

Figure 1. BTL interface circuits.

The obvious solution for an incident wave transceiver was to put bigger output transistors in the driver to sink more current. Doing so increased the capacitance of the output, however, which further depressed the loaded backplane impedance, which then required bigger transistors. The breakthrough solution in 1984 was to design an output stage that could sink high current and still have a low capacitance when inactive. This was accomplished by putting a Schottky diode in series with an NPN output transistor in an open-collector structure, as shown in Figure 1. The very low capacitance of the diode made the total capacitance of the output in the 1 to 2 pF range.

The other design goals for BTL where to improve the noise margin, lower the crosstalk potential, and reduce power consumption in the drivers and the terminators. All of these led to a reduced-amplitude signal of about 1 V and a very tightly controlled threshold (see

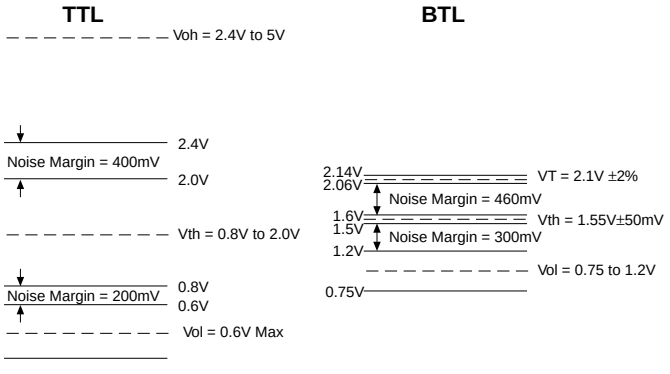

Figure 2. TTL vs. BTL signal levels.

Vendor	Part #	Type	Bits	User I/O	Functional Logic
National	DS3883	Data path	9	Bused	Buffers only
	DS3884A	Control	6	Split	Buffers only; 3 Programmable wire-OR glitch filters with both filtered and non-filtered signals available
	DS3885	Arbitration	9	Bused	Competition latch and logic; Win and greater than detection; Parity checker; All asserted detection
	DS3886A	Data path	9	Bused	Driver has edge triggered latches; Receiver has transparent latches
Signetics and TI	FB2031	Data path	9	Bused	Buffered, registered and latched modes for both driver and receiver paths
	FB2032	Arbitration	9	Bused	Competition latch and logic; Win detection
	FB2033	Control or data path	8	Split	Buffered, registered and latched modes for both driver and receiver paths
	FB2040	Control	8	Split	Buffers only
	FB2041	Control	7	Split	Buffers grouped into a 3-3-1 arrangement; Groups have shared control signals
	FB2042	Control	8	Split	Non-inverted version of FB2040
	FB2043	Control	7	Split	Non-inverted version of FB2041
Signetics	FB2030	Data path	9	Bused	Driver has buffered and latched modes; Buffered receiver path
	FB2034	Control or data path	9	Split	Buffered, registered and latched modes for both driver and receiver paths

Table 1. BTL transceiver chips.

Figure 2), which was implemented using a voltage comparator and a band-gap reference.

Over the last couple of years, most of the development work on BTL devices has centered on adding logic functions, lowering propagation delay, lowering and specifying inter- and intra-package skew, and solving the problems of live insertion and withdrawal. This last topic has two principal areas of interest. The first is the more obvious: glitch-free operation when power is applied or withdrawn. The second is less obvious and in many ways more critical.

At the instant a Futurebus+ board mates or demates with the active bus signals, an impedance change occurs as the bus slot goes from unloaded to loaded or vice versa. This change does not occur on all signal lines at the same time due to the mechanical tolerances. An inserted signal line can present an additional problem due to current inrush as the transceiver, board trace, and connector pin charge themselves to match the bus voltage. These problems are reduced in BTL devices by using an input pin to bias the transceiver's bus output pins into the 1.62 to 2.1 V range prior to insertion. These characteristics of BTL are now specified in IEEE 1194.1.

Transceivers

For several years, BTL transceivers have been the only commercially-available integrated circuits for Futurebus. National Semiconductor produced the first generation of BTL, and they held the trademark for several years. They relinquished their trademark rights for the good of the industry. The first generation consisted of two DIP packaged parts—an octal bidirectional bus transceiver and a quad split-user-I/O transceiver. One unique aspect of the first generation was the use of a resistor voltage divider connected to V_{cc} to set the threshold voltage.

National was also the first to produce a second generation part which had a quad split-user-I/O structure that added a band-gap reference, PLCC packaging, and separate logic, reference, and driver grounds. Texas Instruments joined the BTL market by producing a functionally equivalent part of their own design. During this period, Philips Signetics was also producing BTL-like parts for the IBM-specified Pi bus.

We are now well into the third generation. National, Signetics and TI all produce a full line of BTL transceivers compliant with IEEE 1194.1, as shown in Table 1. Signetics and TI signed a joint development/alternate source agreement in October 1990. At that time Signetics was well into the development of its own family of chips, so some parts are unique to each company.

All three companies have divided their BTL parts into three classes—data path with bused user I/O, control path with split user I/O, and a distributed arbiter part. All the data path parts are 8 or 9 bits wide. Signetics' and TI's parts offer more register, latch, and flow-through options than do National's.

National's 6-bit control-path part is unique because it includes three programmable wire-OR-glitch filters. Placing these required filters in the transceiver lets them see the runt pulses caused by the wire-OR effect, without the distortions introduced by the off-chip TTL buffers used in the Signetics/TI approach.

Both types of arbitration transceivers include the arbiter competition logic—the speed of which is critical to distributed arbitration performance—and where Signetics and TI have a big performance advantage. National picks up points by including parity error and arbitration-number greater-than comparison logic on-chip.

The National parts are offered in both 44-pin PLCC

and PQFP packages, while Signetics and TI use a 52-pin PQFP. Four of the pins on these latter packages are allocated for JTAG, but no JTAG functions have been implemented other than wiring serial in to serial out.

Arbitration Controllers

National's DS3875 Arbitration Controller implements the distributed arbitration protocol—requiring an arbiter on each module—and has a fairly limited feature set. This part was in design before the Futurebus+ specifications had stabilized, and therefore it contains facilities for dead protocols. The 896.1 distributed arbitration specification was primarily written by a National employee, so they've had the advantage of far greater insight into this most-difficult-to-implement Futurebus+ protocol (this complexity is not the fault of the specification's author).

Signetics' FB2012 Central Arbitration Controller provides a single-chip implementation of the central arbiter function—only one central arbiter is required per backplane. The chip has support for only two out of the possible 256 priority levels, but most applications only need two. Signetics has taken advantage of its previously existing asynchronous arbiter technology. This technology resolves metastable conditions internally before an output is allowed to switch. This part has a 6.5 ns propagation delay from a request to a grant.

TI has designed two arbiters—the TFB2010 Arbitration Bus Controller and the TFB2011 Programmable Central-Bus Arbiter. The TFB2010 is a distributed-arbitration controller like the National part, but TI has a far richer feature set. It supports two active programmable priority levels, selected through the request pins. The part has a reasonable arbitration message implementation, and it has a JTAG boundary-scan port. The TFB2011—a superset of the TFB2010—has both central and distributed arbitration functions. With this feature set, an application can use either distributed or central arbitration. A system using distributed arbitration has a redundant central arbiter on every module, and has all 256 central arbiter priorities available. It is likely that many applications will never use the 2010's distributed arbiter functions included in the 2011.

Parallel and Cache Protocol Controllers

The DS3805 Protocol Controller is being codeveloped by National and Newbridge Microsystems (Ottawa, Canada). This chip is targeted for Profile B, which is for I/O applications, and it has no cache or packet support. Split transactions are supported as required by Profile B, but only one outstanding split transaction is permitted. The DS3805, along with 12 transceivers, implements a 64-bit Futurebus+ interface on one side and a 32-bit synchronous user bus interface on the other side.

The Signetics FB2000 Protocol Controller and the FB2020 Packet Data FIFO are a part of the Signetics building-block approach to the interface. These parts implement the most difficult and most performance-critical aspects of the parallel protocols, including maximum-length packet transfers. A user must supply additional protocol-related logic, but these parts don't prevent the user from implementing any set of features needed, and therefore are not oriented to any specific profile.

The 9-bit-wide packet part—the FB2020—contains a packet encoder and decoder, a 64-deep transmit FIFO, a 64-deep receive FIFO, and a 16-deep bypass FIFO. Together with the FB2000, the queued multiple-packet mode protocol can be executed, which yields the highest possible Futurebus+ transfer rates.

TI will offer two different two-chip sets—one for I/O applications and one for cache applications. The TFB2002 I/O Controller and the TFB2022 Data Path Unit make up the I/O chip set. The TFB2051 Data Path for Cache Controller and TFB2055 Data Path for Cache chip set is a superset of the I/O chip set. Each chip set implements a nearly complete 64-bit interface between the user bus and Futurebus+. Profile B is supported, although outstanding split transactions are limited to one, and the chip set can not initiate a split transaction. Some of the protocols required by Profile F are not supported.

Packet mode is supported in the data path parts, but packet length is limited to 32 words, and a single FIFO is shared for both transmit and receive data. The chip sets do not appear to support either multiple-packet mode or queued multiple-packet mode (based upon the preliminary data sheets).

The TFB2051/TFB2055 cache-coherent chip set implements the required coherency protocols. The cache tags snooped by the bus interface must be supplied by the user.

Conclusions

While all three vendors have announced chip sets, none has shipped a complete chip set, and in many cases the available documentation is sketchy. A truly meaningful evaluation of the different chip sets is therefore not yet possible. On paper, each is targeted for a slightly different set of applications. National/Newbridge is targeting the low-end, turn-key market, while TI is targeting the mid-to-upper range turn-key market. Signetics lets the knowledgeable user build whatever they need at the price of more pieces of silicon.

It is likely that all three chip sets will have numerous design bugs. Part of this will be due to complexity—TI has the highest risk here—and part will be due to the lack of previous Futurebus design experience, both at the protocol level and the system level. ♦