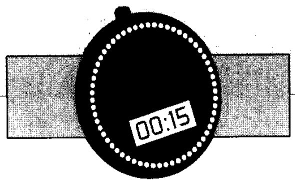

Short Note

Dream Chip #1: A Timed Priority Queue

Real-time programs often need a "time-tagged" priority queue that must be updated on the tick of a clock. A prime example of such a queue is a list of filter coefficients that must be changed on a given sample clock. This short note proposes the architecture for a chip to be used in conjunction with a DSP processor and a fast static memory.

Mark Kahrs

Rutgers University

Digital filters are ubiquitous in digital audio. In a real-time audio system, these digital filter coefficients must be changed quickly in response to a real-time event, perhaps the turning of a potentiometer knob or the pushing of a button. Failure to rapidly change all of the coefficients will result in an audible error.

Now, consider how to build a real-time system that must update numerous filter coefficients on sample boundaries (such as a note in a musical score). This time constraint limitation is known as the parameter update problem.[1] Two earlier solutions from the audio domain are Samson's synthesizer,[2] which uses the notion of a presorted list of parameter changes (but already sorted ones), and Moorer's real-time implementation,[3] which implements a hardware version of a linked list.

I propose a single chip that manages a time-tagged priority queue. The queue connects either to a static RAM or directly to a DSP microprocessor. The SRAM would be part of the address space of a DSP microprocessor executing digital filter code.

Figure 1 (next page) illustrates a typical FIFO. In such a FIFO (for example, the AMD Am7200) data is written into the chip on the falling edge of write enable pin Wr–. Additionally, data is read out of the "other side" of the chip by the falling edge of read enable pin Rd–. A FIFO has several status outputs:

- *Full* indicates the chip cannot accept more inputs,
- *Empty* indicates that reads will fail, and
- *Half* indicates that the FIFO is half full (and conversely half empty) and that perhaps an interrupt is required.

The proposed priority queue departs from a basic FIFO architecture since each datum in the queue will now have an attached event time. See Figure 2.

An on-chip timer (counter) keeps track of the sample number. As the on-chip timer matches the event time of a datum, a memory cycle writes the datum into an external static memory (with another write strobe pin called We–). This arrangement eliminates the need for the processor to get involved with the ready cycle of the queue. If the external static memory is dual-ported, the attached DSP processor will invisibly get its filter coefficients updated at the right sample time.

Architecture

Inputs to the queue include a Reset– pin to clear the timer and a clock pin that will increment the clock counter on the rising edge. An event is a pair of inputs: <alarm_time, data>. An Alarm input gives the time when the event will happen. Obviously, the width of this input depends on the maximum timer value before overflow. To get an hour of use before overflow, the timer must be at

0272-1732/93/0800-0049$03.00 © 1993 IEEE

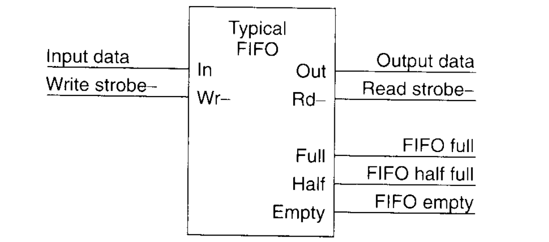

Figure 1. Typical FIFO.

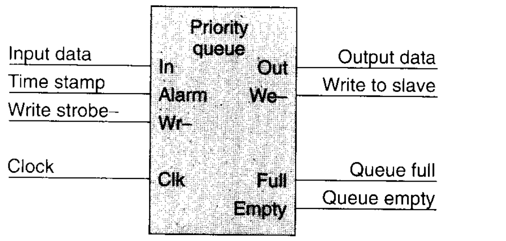

Figure 2. Proposed priority queues.

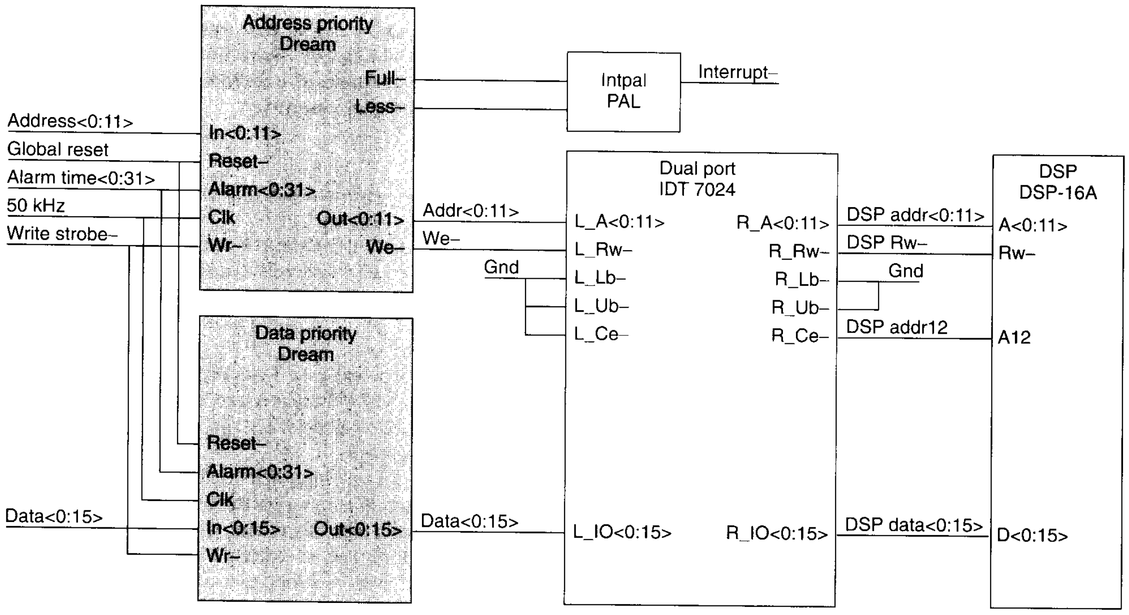

Figure 3. Applying the dream chip.

least 28 bits using a 20-μs clock (audio rate of 50 kHz). So, after rounding up to the next power of 2, we have 32 bits.

A 16-bit In data input may not be large enough to store the address and the data of the external SRAM. So, the SRAM must be expandable. We do this by realizing that priority queue chips in parallel would have the same timer value if they are reset at the same time and clocked with the same clock input. Therefore, we need only to divide the input data (including the slave memory address) into multiple priority queue chips.

When the internal timer (incremented by the clock signal) matches an alarm field in the internal priority queue, data are output and the slave write strobe cycles. Note that since we may have more than one datum with the same alarm time, the priority queue chip must be prepared to cycle the slave memory multiple times within one clock sample time. A prime example of this is the updating of multiple filter coefficients within a single sample time.

Outputs. The chip must control the external static memory and therefore must offer a write control called We– . Of course, the 16-bit-wide data output Out must also be present.

We also have some status output pins: Tick– means that an event has been matched for the first time after the clock increments. Less– indicates that the alarm_time just written is less than the current clock time. Full– indicates that the chip is full, and any attempt to write another datum would be ignored.

Internals. Outside of the internal clock register, the rest of the chip must implement the priority queue. We have several ways of doing this. One is to have an internal pointer memory; updates to the queue must insert the new pair into the pointer memory. At the worst case, this would take log n cycles, assuming the list is kept sorted. For a full list of 1,024 event pairs, updating would consume 10 cycles. Assuming each cycle is say, 50 ns, this would take 0.5 μs and result in a maximum of 40 coefficients at the 20-μs sample rate.

A second approach would be a linked list. However, updates could take a very long time since in the worst case the entire list might have to be searched. For music based on scores, an ordinary queue is fine: You know the times will be sorted when you enqueue the event or note. Alternatively, the memory could be self-sorting, such as a bubble sort.[4]

Applications

Here is a typical application of the proposed dream chip: The clock input is connected to the sample 50-kHz oscillator. The multiplexed RAM data/address pins connect to a fast dual-port RAM with an address latch (in this case, an IDT 7024). The L (left) port of the memory connects to the priority queue; the R (right) port connects directly to a 16-bit DSP processor. The external processor that knows very little about the real-time nature of this process controls the priority queue inputs! See Figure 3.

Related silicon

Related chips exist. AMD produced the 95C85 data manager, though it only stored a limited priority and was too slow for this application (at best 5 μs). MIT designed and built an experimental priority FIFO chip called the FIPO in 1979; but it's not an off-the-shelf item.

WITHOUT INTEGRATING THIS PRIORITY QUEUE on silicon, the board space consumed by the priority queue would be unreasonably large. A chip with space for 1,024 events would need an internal memory of [32 (time) + 16 (data) = 48 bits] 1,024 = 49,152 bits. Obviously, we can expand the space available with a larger die size (and accompanying cost increase). The total pin count is 69 pins, not including power and grounds—an easy fit into an 84-pin PLCC. Besides easing board space concerns, such a chip would also simplify the design of any real-time processor using time-stamped event queues.

References

1. J.A. Moorer, "Synthesizers I Have Known and Loved," *Computer Music J.*, Vol. 5, No. 1, 1981, pp. 4-12.
2. P. Samson, "A General-Purpose Digital Synthesizer," *J. Audio Engineering Society*, Vol. 28, No. 3, 1980, pp. 106-113.
3. J.A. Moorer, "A Flexible Method for Synchronizing Parameter Updates for Real-Time Audio Signal Processors," *Proc. 79th AES Convention*, 1985.
4. T. Kohonen, *Self-Organization and Associative Memory, 3rd ed.*, Springer-Verlag, Berlin, New York, 1989.

Mark Kahrs teaches courses in digital signal processing and computer architecture in the Department of Electrical and Computer Engineering at Rutgers University. He is also an amateur food engineer.

Direct questions about this article to the author at kahrs@winlab.rutgers.edu.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

Low 165 Medium 166 High 167