Reducing CRT display overheads in Z80 systems using refresh-mode transfer

A method of greatly reducing the overhead for display maintenance with Z80-based VDUs is outlined. **Thomas Alexander and M V Subramanyam** use a special feature of this microprocessor to reduce processor workload from a normal 15-25% to less than 5%.

CRT displays   Z80   refresh-based transfer

A CRT screen needs to be refreshed once every frame, 50 or 60 times a second. This involves the transfer, per frame, of up to 2 kbytes of data, and hence can use a lot of available CPU time. LSI display controllers using DMA transfer can reduce this overhead to reasonably low levels, but will still need 15–25% of bus time.

Usually, one of three methods is used to fill the CRT controller row buffers from memory

- shared memory, for instance in the 6847 controller
- DMA transfer using a DMA controller
- processor-controlled (software) DMA transfer

The first two require considerable hardware; the latter entails memory space and processing time.

Z80 REFRESH MECHANISM

This microprocessor is unique in that it directly supports dynamic RAM refresh. An internal 7-bit counter (the R register) emits a refresh address during the T_3 state of an opcode fetch machine cycle. The counter is incremented automatically and its contents are placed on A_0–A_6 of the address bus (see Figure 1). A control signal, $\overline{RFSH}$, indicates when an $\overline{RAS}$-only refresh can be carried out. During this period, the data bus is tristate, all other control lines inactive (high) and the interrupt register (I register) contents appear on A_8–A_{15}. (A_7 reflects the 8th bit of the R register, and can be set or reset by program control.)

Hence in every 128 successive opcode fetch cycles 128 consecutive 16-bit addresses (produced by concatenating the I and R registers) are emitted by the Z80, along with the $\overline{RFSH}$ signal.

CRT DISPLAY REQUIREMENTS

An 80 x 24 character display, with a 6 x 10 matrix for each character, needs 80 bytes to be supplied to the CRT controller every 600 μs (10 horizontal retrace periods of 60 μs each). Hence one byte should be transferred every 7 μs, on the average.

OPERATING PRINCIPLES

The basic idea is to use the 16-bit refresh address supplied by the CPU during T_4 to load the row buffers of a CRT controller (here, an 8275) from display memory. Between successive refresh periods, the refresh address will be automatically incremented; hence successive display characters are loaded into the buffers. Since the interval between successive T_3 states never exceeds 7 μs (worst case: 6 μs) the display will always be maintained. In addition, as the instruction decode does not require the system bus, the transfer is entirely transparent to the CPU. The only remaining work load is the reinitialization of the refresh counter and interrupt register at the start of each character line, once every 600 μs.

HARDWARE DESCRIPTION

The extra hardware performs the following.

- An interrupt is generated at the leading edge of the 8275 DMA request signal, issued at the start of each character line.
- The refresh address emitted is allowed to access the display RAM, causing data to appear on the bus soon after $\overline{RFSH}$ is asserted low.
- Data is strobed into the 8275 buffer using its $\overline{DACK}$ (DMA grant acknowledge) and $\overline{WR}$ lines.
- When DREQ of the 8275 goes low again (80 bytes transferred), the transfer logic is reset.

The realization of the above is fairly straightforward. The implementation required five TTL chips, all 74-series SSI gates, flipflops and monostables.

Since the display RAM (two 6116's) should be accessible to both the Z80 and the 8275, the $\overline{RD}$ for the chips is obtained as an OR or the CPU $\overline{RD}$ line and the 8275 DACK

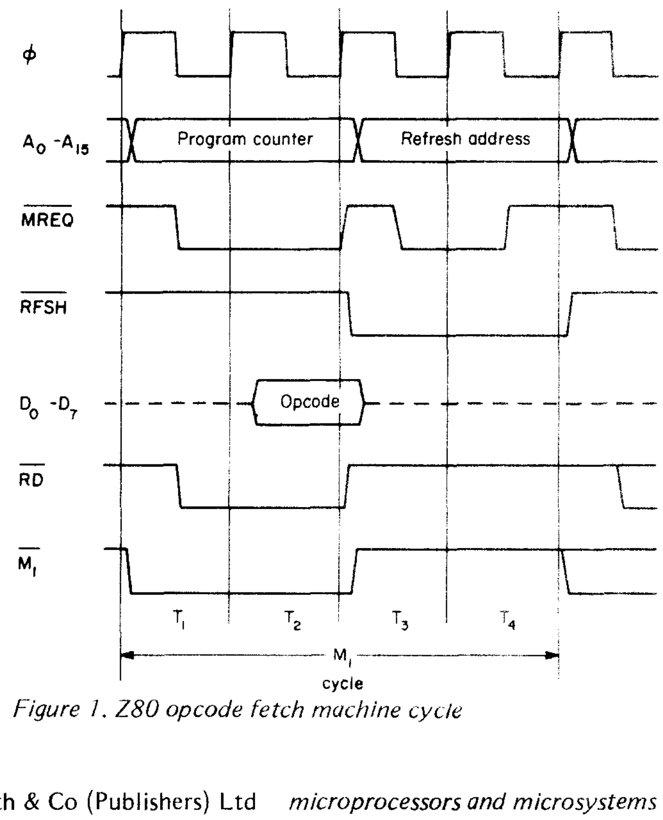

Figure 1. Z80 opcode fetch machine cycle

Department of Electrical Engineering, Indian Institute of Technology, Madras 600036, India

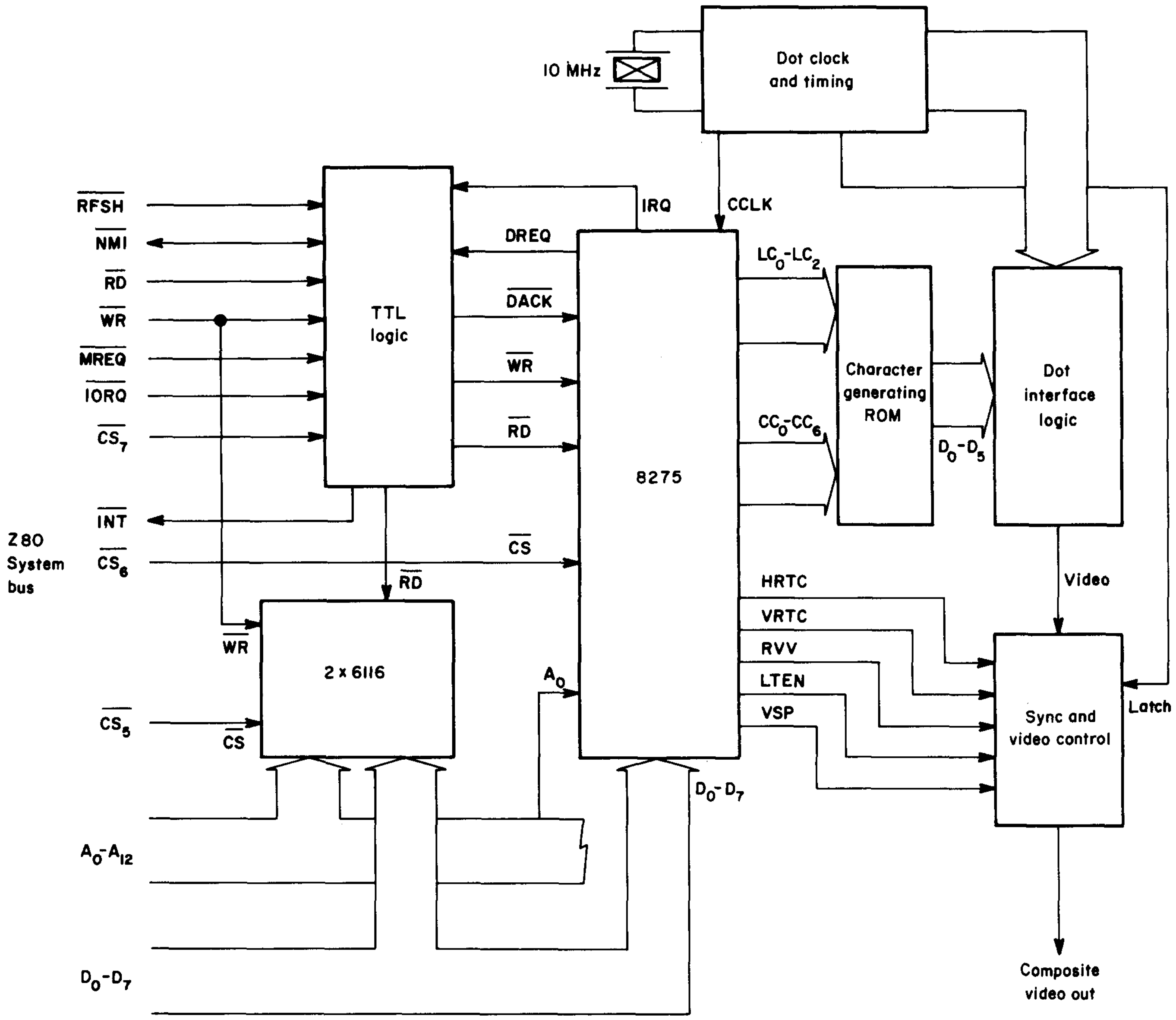

Figure 2. Hardware block diagram

line. The $\overline{WR}$ pulse for the 8275 is generated one-half clock cycle after the $\overline{DACK}$ is asserted, to fit the 8275 timing requirements. A spare chip select from the decoder was used to enable the transfer hardware after counter reinitialization (under software control). This was needed as the refresh counter increments continuously; it cannot be frozen. At the start of each character row, the DREQ signal is used to generate an $\overline{NMI}$ interrupt for the Z80.

The general configuration is shown in Figure 2.

SOFTWARE REQUIREMENTS

The software requirements are minimal. The $\overline{NMI}$ interrupt service subroutine increments the interrupt register every two character lines, reinitializes the refresh counter (complementing the eighth bit each time), enables the transfer hardware with a dummy read and saves the current display status. 8275 initialization is done as a part of the startup routine. A total of77+60 bytes were needed.

PERFORMANCE COMPARISON

A comparative analysis was made between the system as described, and two conventional methods, in terms of time available per frame (one frame is completed every 15660 μs).

DMA transfer scheme, using an 8275 DMA controller

- CPU time needed – 253 μs
- DMA transfer time – 2000 μs

Hence the CPU is available for 85% of the time.

Software transfer, using 40 POP instructions to transfer 80 bytes

- CPU time needed – 4400 μs

Hence availability = 72%.

DRAM refresh-based transfer

- CPU time needed – 920 μs
- Bus time needed – zero

Hence the availability is 95%.

POSSIBLE DISADVANTAGES

Every system has some drawbacks, this one is no exception.

- The interrupt vector register is tied to the display refresh. This means that the Z80 $\overline{\text{INT}}$ cannot be used with the powerful MODE 3 interrupt employed in Zilog peripheral chips.
- The refresh counter resets to 00H or 80H after only 128 counts; hence 48 bytes of display RAM must be left unused for every 80 bytes occupied. This is because the start and end of each character row must lie within the same 128 byte page. Hence around 3 kbytes will be needed to display 2000 characters.

POSSIBLE ADVANTAGES

- Possibility 1: A_8 of the display RAM could be supplied, via a 3-state buffer, from a flipflop toggled by the reset of the refresh counter to 00H. The display RAM can now be reduced to 2 kbytes (only 16 bytes wasted every three lines) with the added bonus of a 66% reduction in refresh overhead, and a 98% availability.
- Possibility 2: if only 64 characters are displayed per line, all character lines fall within page boundaries, and, in addition, overhead is reduced by 50%.
- Possibility 3: the extra 1 kbyte can be used to store system variables, or, more important, interrupt vector addresses. In a 256 byte page, 96 bytes will be free and can be used to store 32 jump instructions (loaded into the RAM at startup). These 96 bytes are copied into every 256 byte page in the RAM. Now, on a MODE 3 interrupt, it is only necessary to have the interrupting device supply the right address within the page; it does not matter where exactly the I register points to. So, up to 32 MODE 3 interrupts can be used.

CONCLUSIONS

A method of effectively using the autorefresh feature of the Z80 has been outlined. It has been shown that such a method can save up to 20% of processor time, a fair amount of software, and uses very little extra hardware. The operation is entirely transparent, both to the CPU and to the user, implying that user programs can be run at nearly the full speed while still maintaining the display.

The dot clock, character generator, etc. are made from the standard circuits: no change is needed on this side of the CRT controller.

A possibility exists for simple bit-mapped graphics, using a Z80 alone (no CRT controller) with refresh-based transfer. Here, the processor would execute NOPs (achieved by a software HALT) for each horizontal display line, and reset the registers during horizontal retrace. The vertical retrace period (1–2 ms) could be used for processing, and changing the display. Using a 16 kbyte RAM, a 300 x 300 pixel display could be obtained.