TO: Prof. J. Saltzer

FROM: Roger Schell

DATE: March 15, 1971

SUBJECT: Reconfiguration in Follow-on Hardware

From reading preliminary specifications and from discussions with engineers, I have concluded that reconfiguration will be made difficult by portions of the GE655 design to be reflected in the Multics follow-on hardware. These features are pointed out as an aid in planning for the use of the follow-on hardware.

The current (GE635) system controller can signal an "interrupt present" to a processor even though the port to the processor is disabled: this causes the processor to "hang-up" since it cannot access the controller via the disabled port to determine the interrupt number. This problem is particularly difficult for an operator to debug. An "MIT only" fix is being made to the current hardware, but the follow-on controller will bring back the bug.

In the current Multics "service system" the program controlled port enable registers in the system controller are used to determine the configuration of active modules. This allows reconfiguration without an (error-prone) operator changing port enable switches on a system controller that is being used. All port enable switches are always in the program control position -- this approach works because whenever the controller or the system is initialized (viz. at bootload time) all ports are enabled. In the GE655 controller the initialization condition is disabled. Any controller using only program controlled port enable registers cannot be bootloaded, since all ports to active modules (including the bootload processor) are disabled. Thus, in the follow-on hardware the operator will be required to risk setting port enable switches manually for bootloading and setting them to the program control position on the "service system" controllers before doing reconfiguration.

The lack of read-alter-rewrite (RAR) instructions in the follow-on hardware interacts badly with the processor appending mechanism and dynamic memory removal. In the GE655, per se, all

RAR needs for control of conflicts over shared memory locations can be resolved by an additional "lock" variable and a single "conditional store" instruction since all references to memory are explicit. However, the appending mechanism makes implicit references to memory (with no way to make conditional on a "lock") to set the modified bit in page table words, and these page tables are shared data to the reconfiguration procedure. The modified bit is used to prevent incorrect versions of pages being retained on secondary storage (leading to potential "time bombs"). When moving wired down pages the current design for reconfiguration not only uses the modified bit to detect changes while moving a page but also must leave a correct modified bit in the page table word to trigger secondary storage updating. The modified bit at the beginning of the move is saved and restored at the end with an "OR to storage" RAR instruction. Without the RAR, another processor can set the modified bit after the "OR" instruction reads storage only to have it turned of when the result of the "OR" is stored back into memory.