State Diagram:

Address Translation:

10: Store value at MAR in VA

11: Store PTBR + LSHF(VPN, 1) in MAR

45: Load PTE from physical address (MAR)

47: Buffer state to test for protection and page fault

61: Set reference and write bits (if necessary)

44: Store the new PTE into physical address (MAR)

46: Store PFN.append(VA[8:0]) in MAR. Clear Write and TRAP flags. Next state determined by NS register.

Datapath:

The GateVA structure is used to pass appropriate values into the MAR. It connects 2 new registers – PTBR and VA to adders which eventually pass into a MUX that is connected to the bus. The first adder calculates PTBR + LSHF(VA, 1) and the second adder calculates the PFN appended with the offset in the VA. In addition, the VA can be loaded with values from the bus.

The NS structure is used to correctly to calculate next state in case of address translation. A mux is connected to the register’s input and next states can be loaded in by selecting specific lines.

Two additional registers were added TRAP and W, for special purpose address translation. W flag indicates if the modified should be set in a PTE and the TRAP flag indicates if the address being translated is used for a TRAP instruction. These can be cleared when address translation is finished.

The GateMAR structure is used to be able to pass the value in MAR onto the bus.

New Control Signals:

LD.TRAPFLAG: Set the trap flag

TRAPFLAGCLR: Clear the trap flag

LD.WFLAG: Set the write flag

WFLAGCLR: Clear the write flag

LD.VA: Load the VA register

LD.NS: Load the next state register

LD.MR: Used to set bits 1 and 0 of the MDR. Specifically used during address translation.

GateVAMUX: Allow value from VAMUX to be loaded onto the bus

GateMAR: Allow value in MAR to be loaded onto the bus

VAMUX: Select which adder’s output to pass onto bus

0: PTBR + LSHF(VA, 1)

1: PFN.append(VAoffset)

NSMUX: Select which value to load into the NS register

000: 29

001: 25

010: 23

011: 24

100: 33

101: 28

Microsequencer:

I modified existing and added additional condition bit combinations. Currently,

0110: unknown exception

0111: unaligned

1000: protection or page fault

In other words, these indicate in a state that it’s possible for these exceptions to occur. However, they still all result in state 63. I chose to this because only certain exceptions can occur at any time, and having all the exceptions end up in a single state reduces states used significantly. Other condition bit combinations remain the same as Lab 4.