Design Specification

YUCHAO HAN   yh7483

1. State diagram

My jobs to do with state diagram mainly are 3 parts.

a. Add subroutine initialize state. It is shown in the right. I use two signal to detect interrupt: [INT] & PSR[15]. That means only in the user mode when I detect a [INT] signal, I can start a subroutine initialize only once (because I change into supervisor mode in subroutine initialize state). In the subroutine, I save the current PSR, set the correct vector for interrupt/exception, change mode, save USP and set R6 as SSP, push old PSR and correct PC into user stack successively. And finally, get the subroutine PC and go to state 18 to start my subroutine program.

b. RTI instruction state. I pop the PC and PSR of the user program. Restore the R6 to the USP.

c. Change the state of LDB/LDW/STB/STW. Protection exception only occurs in these 4 instructions and unaligned access exception occurs in STW/LDW. In order to save the number of state, I merge these 4 instructions into one state to check if there occurs protection exception or unaligned access exception. If so, I have a state to tell whether it is protection exception or unaligned access exception. And then jump to subroutine initialize state to start an exception handler. If there's no exception, I use IR[15:12] and use IRD to separate the 4 instructions.

2. Datapath

a. The part related to R6
I use a mux to select one from SSP, USP, R6+2, R6-2. This structure is to support state 34, 37, 39, 25, 40, 43.

b. The part to generate EXCBR(exception branch) signal for control.
This branch signal is for protection exception or unaligned access exception. It is a control signal in microsequencer. Also EXCBR signal is generated by or X and Y signal. X signal is high when a protection exception happen (bus value smaller than 0x3000 in user mode). Y signal is high when a unaligned access exception. Note that Y signal equals to BUS[0] in STW/LDW and is always '0' in STB/LDB.

c. The part related to Vector
This structure is to generate the MAR for subroutine. First I choose which vector to specify. Then I left shift it by 1 and combine it with 0x02(as high 8 bits) to make the address.

d. Add structure for PSR and change structure for CC
PSR15 has may have two inputs: BUS[15] or '0'. It has the ability to drive bus through GatePSR15, which is to support state 55, 10, 11, 49. As the same, CC may have two inputs: from logic or from BUS[2:0], and has a gate to drive bus.

e. Changes made to DRMUX and SR1MUX
Add a select choice from R6.

3. New signals in microinstruction

COND2:	in microsequencer help to specify next state
LD.PSR15:	load PSR[15] signal
LD.EXCBR:	load EXCBR signal
LD.X:	load X signal
LD.Y:	load Y signal
LD.VECTOR:	load Vector signal
LD.SSP:	load SSP signal
LD.USP:	load USP signal
GatePSR15:	PSR15 drive bus (when push PSR)
GateCC:	CC drive bus (when push PSR)
GateVECTOR:	drive bus to give out the address of subroutine
GatePC2:	drive bus use PC-2
GateR6:	the output of R6MUX to drive bus
DRMUX1:	an added signal for DRMUX, used to specify R6 as DR
SR1MUX1:	an added signal for SR1MUX, used to specify R6 as SR1
R6MUX:	select from 4 inputs: SSP, USP, R6+2, R6-2
YMUX:	select input of Y
PSRMUX:	select input of PSR15
CCMUX:	select input of CC
VECTORMUX:	select input of Vector
MYIRD:	in microsequencer, help to separate LDB/LDW/STB/STW

4. Microsequencer

I add a MYIRD signal to choose one input of the IDR MUX, which take effect when IRD signal is high (state 32, state 23). Note that I use "$1,1,\overline{IR[15]},\overline{IR[14]},\overline{IR[13]},\overline{IR[12]}$ ". This is because in the next state of LDB/ LDW, they are accessing memory, so they are waiting for ready bit of memory. That means the 2nd bit of state should be 0. So I invert the IR[15:12].
And also, I add COND2 to help specify the next state because of EXCBR/INT/PSR[15] signal.