Name	RTL Instruction	Active Control Signals
Fetch	$IR = M(PC)$	MR, IRce
Decode	$A = RS1$, $B = RS2$ $PC = PC + 1$	Ace, Bce, S2sel[1], S2sel[0] PCce, add
Alu	$C = A$ op B	S1sel[0], Cce
TestI	$C = (A$ rel $imm)$	S1sel[0], S2sel[0], Cce, test, Itype
AluI(add)	$C = A$ + imm	S1sel[0], S2sel[0], Cce, add, Itype
Shift	$C = A$ shift sa $sa = 1, (-1)$	S1sel[0], Cce DINTsel, shift (,right)
Adr.Comp	$MAR = A + imm$	S1sel[0], S2sel[0], MARce, add
Load	$MDR = M(MAR)$	MDRce, Asel, MR, MDRsel
Store	$M(MAR) = MDR$	Asel, MW
CopyMDR2C	$C = MDR(\gg 0)$	S1sel[0], S1sel[1], S2sel[1], DINTsel, Cce
CopyGPR2MDR	$MDR = B(\ll 0)$	S1sel[1], S2sel[1], DINTsel, MDRce
WBR	$RD = C$ (R-type)	GPR_WE
WBI	$RD = C$ (I-type)	GPR_WE, Itype
Branch	branch taken?	
Btaken	$PC = PC + imm$	S2sel[0], add, PCce
JR	$PC = A$	S1sel[0], S2sel[1], add, PCce
Save PC	$C = PC$	S2sel[1], add, Cce
JALR	$PC = A$ $R31 = PC$	S1sel[0], S2sel[1], add, PCce GPR_WE, jlink

Table 7.3: The active control signals in each state

It is possible to almost perfectly combine the advantages of precise decoding and imprecise decoding by performing imprecise decoding. Every instruction execution has a "commit" stage (namely, write-back, store, or changing of the PC in a jump). One could condition the commit stage of execution upon the legality of the instruction. The legality of the instruction is computed while the instruction is executed (not in the decode stage), and so time is not wasted on checking the legality of legal instructions. We will not consider this more sophisticated option, and use only imprecise decoding.

The conclusion is that we suggest to do an imprecise decoding of the instructions. That mean that some illegal instructions are considered as legal encodings of other instructions. We leave it to you to choose which instructions are decoded in such cases.

The usage of imprecise decoding means that we cannot reach the Halt state using an illegal instruction. We therefore, add a HALT instruction that causes a transition to the Halt state.