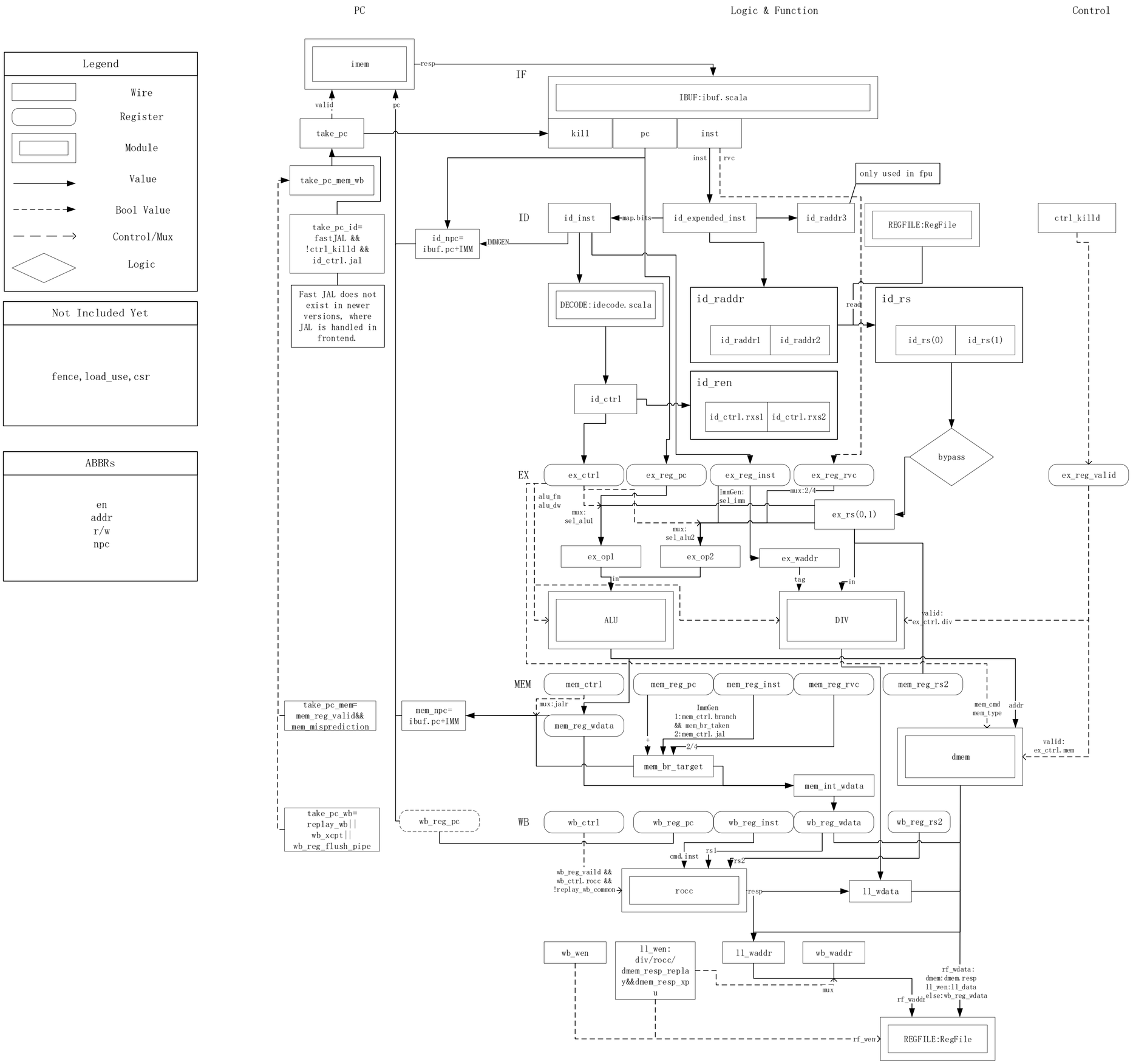
PC
Logic & Function
Control
Legend
Wire
Register
Module
Value
Bool Value
Control/Mux
Logic
Not Included Yet
fence,load_use,csr
ABBRs
en
addr
r/w
npc
imem
resp
IF
IBUF:ibuf.scala
kill
pc
inst
valid
pc
take_pc
take_pc_mem_wb
take_pc_id=
fastJAL &&
!ctrl_killd &&
id_ctrl.jal
Fast JAL does not exist in newer versions, where JAL is handled in frontend.
id_npc=
ibuf.pc+IMM
IMMGEN
inst
rvc
only used in fpu
ID
id_inst
map.bits
id_expended_inst
id_raddr3
REGFILE:RegFile
ctrl_killd
DECODE:idecode.scala
id_raddr
id_raddr1
id_raddr2
read
id_rs
id_rs(0)
id_rs(1)
id_ctrl
id_ren
id_ctrl.rxs1
id_ctrl.rxs2
bypass
EX
ex_ctrl
ex_reg_pc
ex_reg_inst
ex_reg_rvc
ex_reg_valid
alu_fn
alu_dw
mux:
sel_alu1
ImmGen:
sel_imm
mux:2/4
ex_rs(0,1)
mux:
sel_alu2
ex_op1
ex_op2
ex_waddr
in
tag
in
ALU
DIV
valid:
ex_ctrl.div
MEM
mem_ctrl
mem_reg_pc
mem_reg_inst
mem_reg_rvc
mem_reg_rs2
take_pc_mem=
mem_reg_valid&&
mem_mispredicition
mem_npc=
ibuf.pc+IMM
mux:jalr
mem_reg_wdata
ImmGen
1:mem_ctrl.branch
&& mem_br_taken
2:mem_ctrl.jal
+
2/4
mem_br_target
mem_int_wdata
mem_cmd
mem_type
addr
dmem
valid:
ex_ctrl.mem
take_pc_wb=
replay_wb||
wb_xcpt||
wb_reg_flush_pipe
wb_reg_pc
WB
wb_ctrl
wb_reg_pc
wb_reg_inst
wb_reg_wdata
wb_reg_rs2
cmd.inst
rs1
rs2
wb_reg_vaild &&
wb_ctrl.rocc &&
!replay_wb_common
rocc
resp
ll_wdata
wb_wen
ll_wen:
div/rocc/
dmem_resp_repla
y&&dmem_resp_xp
u
ll_waddr
wb_waddr
mux
rf_waddr
rf_wdata:
dmem:dmem.resp
ll_wen:ll_data
else:wb_reg_wdata
rf_wen
REGFILE:RegFile