6.004 Design project implementation notes (continued 2nd sheet)

Exception Handling implementation

- We no longer need the wasel signal, since a BNE(R31,0,XP) is inserted into the pipeline which has rc = XP on its own. We do compute_wasel (aka is_ILLOP) in RF stage to know if ILLOP exception happened.
- Note in the computations for IRSrc signals, we give higher authority to IRSrc[1] (for BNE due to exception) by using BNE BNE NOP ↓ to select instruction, since we must stop at first instruction that causes an exception

BNE	BNE	NOP	↓
11	10	01	00

$IRSrc^{IF}[0] = branch\_taken + reset + except\_RF + except\_ALU + except\_MEM + except\_WB$

$IRSrc^{IF}[1] = except\_IF$

$IRSrc^{RF}[0] = reset + except\_ALU + except\_MEM + except\_WB + stall$

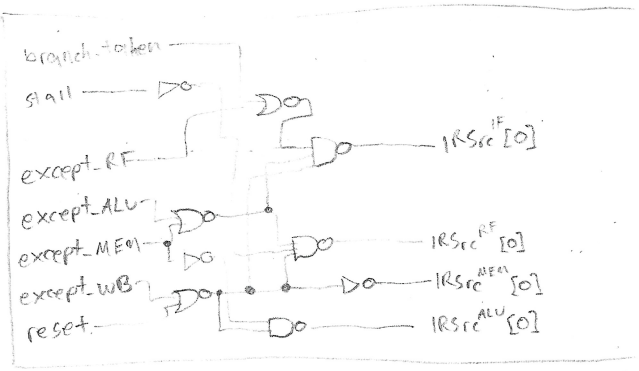
$IRSrc^{RF}[1] = except\_RF$

$IRSrc^{ALU}[0] = reset + except\_MEM + except\_WB$

$IRSrc^{ALU}[1] = except\_ALU$

$IRSrc^{MEM}[0] = reset + except\_WB$

$IRSrc^{MEM}[1] = except\_MEM$

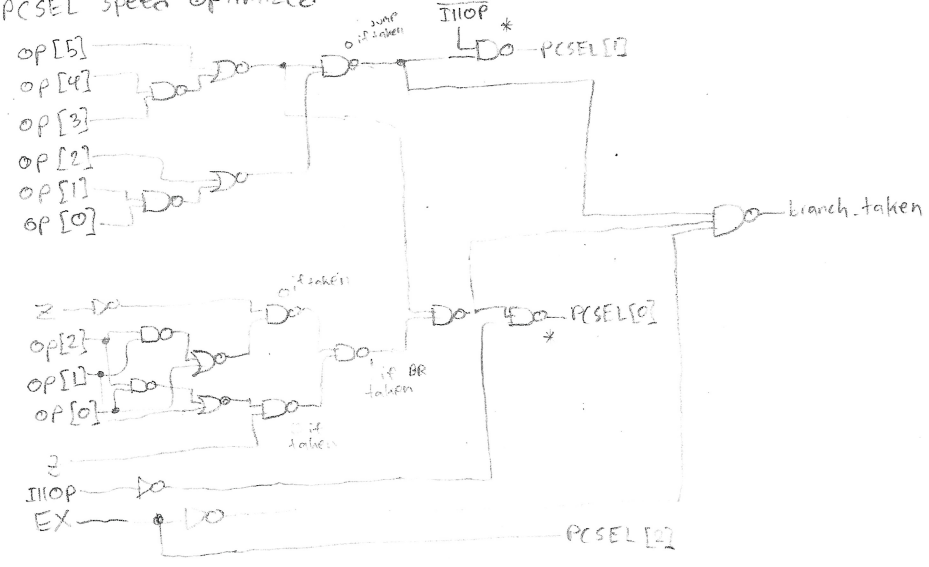


- except_RF = interrupt + ILLOP_RF (since interrupt handled like regular exception, but let all previous instructions finish)
- all other exceptions are constant 0 in this version of Beta B
- Since we handle interrupts in RF stage (and use a BNE) no other stage ctl. signals depend on IRQ, so I removed this redundant dependence from the CTL computations.

Critical Path optimization

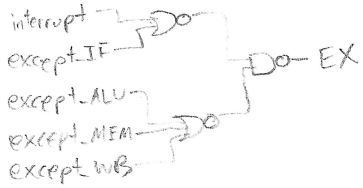
- swapped the OR3 for a NOR3 + INV.

PCSEL speed optimized (considering 2 as latest input)



* can remove these inputs for an inverted pcsel[1:0] that has not taken is_ILLOP into account. For the most optimized PC (see opposite side of page)

EX is non-ILLOP exception (including interrupt)



NOTE: This implementation dangerously assumes the PC implementation used in this writeup, where PCSEL[2] alone is checked for selecting Xaddr, PCSEL of 0x4, 0x5, 0x6, 0x7 are all valid signals for Xaddr, which has higher authority than ILLOP or other lower signals.