These charts are incomplete and represent only one possible design point. You should use these as a starting point for your design; many optimizations are possible to improve the CPI performance of your implementation.

State registers

- haltff_current <= '0'

- int_on_next <= int_delay_current

- iop_enable_next <= "100"

- cont is a front panel switch. Depending on your front panel design, you might be doing things differently here.

- haltff_current <= '0'

- int_delay_next <= '0'

- alu_sel <= alu_zero

- loadma <= '1'

- ce_bar <= '0'

- jms0 <= '1'

- loadir <= '1'

- data_sel <= sel_pc

- alu_sel <= alu_max

- loadpc <= '1'

- read <= '1'

- ce_bar <= '0'

- data_sel <= sel_ir

- alu_sel <= alu_max

- irc_next <= bus_out

- data_sel <= sel_pc

- alu_sel <= alu_inc

- loadpc <= '1'

- puts PC into MA and MB

- increment the PC

- read instruction into IR

- puts zero on the address bus

- turns off arrays

- puts the instruction out for decode by the controls
alu_sel <= sel_mux
irc_next <= bus_out

F2.2

F2.1

data_sel <= sel_ir
alu_sel <= sel_mux
irc_next <= bus_out

put IR on the bus

splice the address

F3

data_sel <= sel_ea
alu_sel <= sel_mux
loadma <= '1'
ma_next <= bus_in

F3.1

load mem(MA) -> IR

data_sel <= sel_ma
alu_sel <= sel_mux
loadir <= '1'
ce_bar <= '0'
read <= '1'

F3.2

load IR into MB

data_sel <= sel_ir
alu_sel <= sel_mux
loadmb <= '1'

irc_current(3) <= '0'

first direct then autoindex check

T

E0

T

E0
ma_current(0 to 8) = "00000001"

(altoindex)

data_sel <= sel_mb
alu_sel <= sel_inc
loadmb <= '1'

write it back

F6

data_sel <= sel_ma
alu_sel <= sel_mux
read <= '0'
ce_bar <= '0'

(data_sel <= sel_mb)
(alu_sel <= sel_mux)
loadma <= '1'
ma_next <= bus_in

F6

(irc_current(0 to 2) = OP_JMS)
(irc_current(0 to 2) = OP_JMP)

F4.1

data_sel <= sel_ma
alu_sel <= sel_mux
loadma <= '1'
ma_next <= bus_in

E0

F4.auto

data_sel <= sel_mb
alu_sel <= sel_inc
loadmb <= '1'

increment the contents

F7

data_sel <= sel_ma
alu_sel <= sel_mux
loadma <= '1'
ce_bar <= '0'

F7.1

(data_sel <= sel_ir)
alu_sel <= sel_mux
loadmb <= '1'

E0
irc_current(0 to 2)
right <= '1'
left <= '1'
data_sel <= sel_mb
alu_sel <= alu_add

loadpc <= '1'
data_sel <= sel_ma
alu_sel <= sel_mux

write MB to memory

data_sel <= sel_ma
alu_sel <= alu_mux
read <= '0'
ce_bar <= '0'

irc_current(0 to 2)
data_sel <= sel_ac
alu_sel <= alu_zero
right <= '1'
left <= '1'

haltff_next <= sing_inst_in
OP_IOT

OP_OP

OP_DCA

E0

E1.IR

E_OP1

int_delay_next <= '0'
int_on_next <= '0'
irc_current = "11000000001"

F1
iop_enable_next(0) <= '0'
iop_enable_next(1) <= iop_enable_current(0)
iop_enable_next(2) <= iop_enable_current(1)
iop_control_next(0) <= IOSKIP
iop_control_next(1) <= ACCLR
iop_control_next(2) <= ORAC
loadmb <= '1'
datasel <= sel_ac;

data_sel <= sel_pc
loadpc <= '1'

loadir <= '1'

loadmb <= '1'

E_IOP3

data_sel <= sel_ir
left <= '1'

alu_sel <= alu_or
right <= '1'

iop_control_current(2) <= '1'

E_IOP5

iop_control_current(3) <= '1'

E_IOP4

alu_sel <= sel_zero
right <= '1'

left <= '1'

iop_control_current(2) <= '1'

E_IOP2

iop_enable_next(3 to 4) = "10"
irc_current(3 to 8) = "000100"
read <= '1'

E_IOP1

iop_enable_next(3 to 4) = "10"
irc_current(3 to 8) = "000100"
read <= '0'

E0