Pipelining RI_RISC-V Data_path Architecture

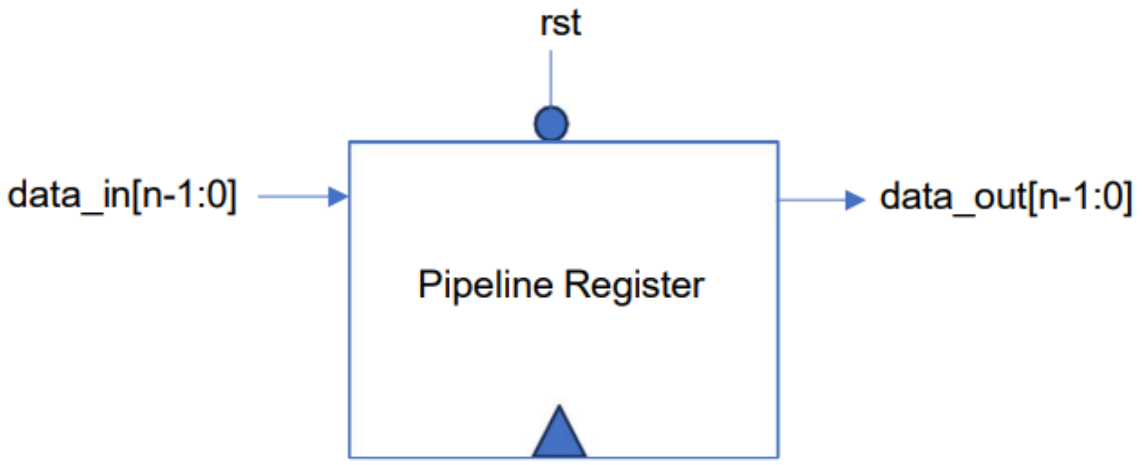
NCDC

Submitted by: **Muhammad Farhan Shah**

Pipelining:

A pipelined data_path needs to "separate" the five stages so that each stage can process data from different instruction. To obtain this functionality, intermediate states between different stages must be stored in a state element i.e. a pipeline register.

A pipeline register may have an n-bit input, a reference clock, a reset signal and a n-bit output. A reference block diagram in shown in the figure below.



Benefits:

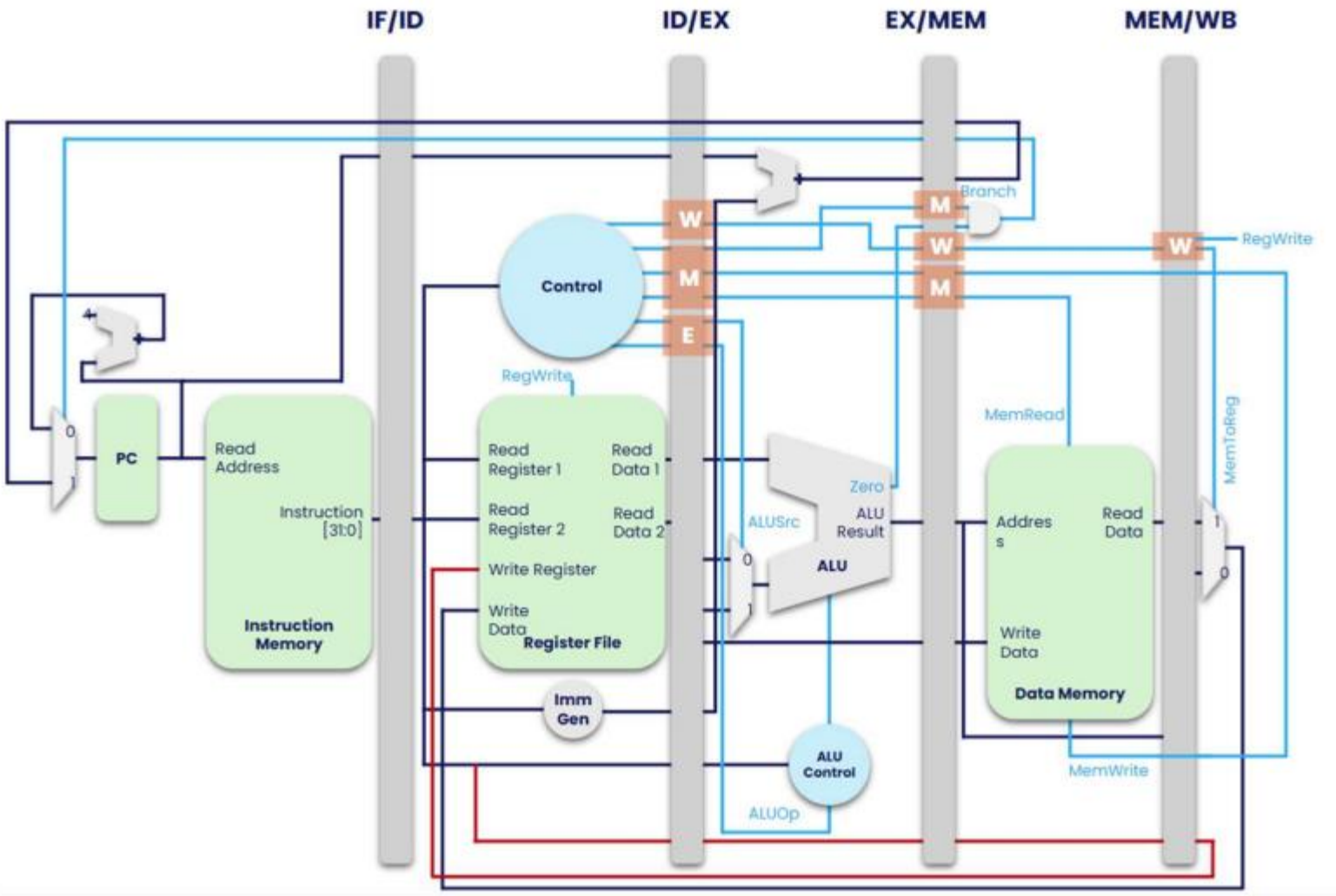
There are two main benefits of pipelining the data and control paths

1. Reduces the **critical path**, and hence increases the maximum **operating frequency.**
2. Allows **instruction level parallelism**, i.e. multiple instructions are being executed in different stages of the processor.

Although these benefits may be at the cost of increased execution time of a single instruction, the true effect of pipelining can be observed **in longer executions** of entire programs

4

Integration of Pipeline Registers in Datapath :



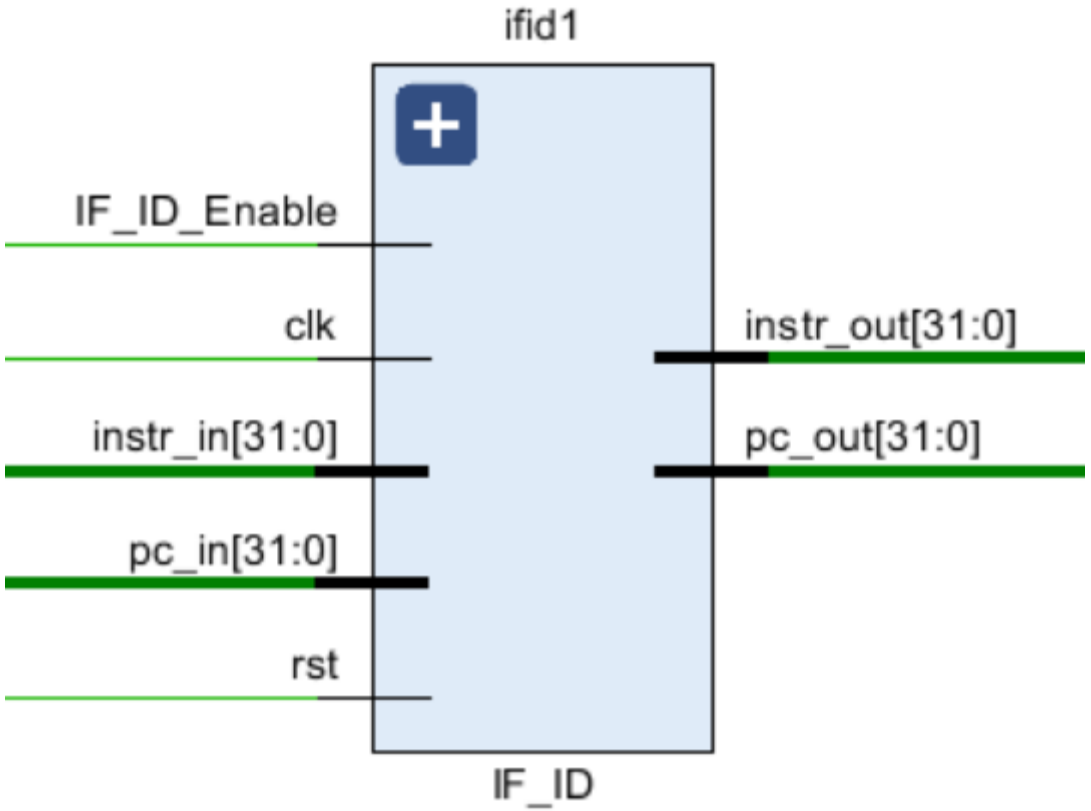
New modules to be integrated:

There are two main benefits of pipelining the data and control paths

1. **module IF_ID #(parameter WIDTH = 32)(**
2. **module ID_EX #(parameter WIDTH = 32)(**
3. **module EX_MEM (**
4. **module MEM_WB (**

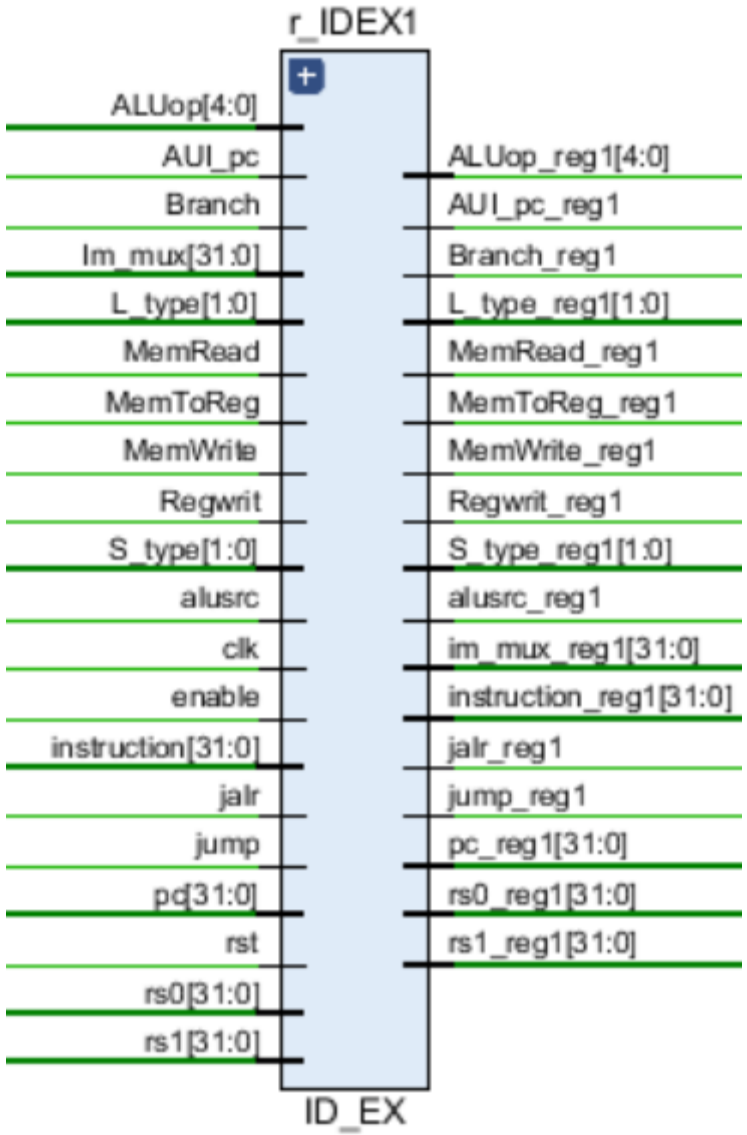
All these modules have control as well as the data path signals

IF_ID :

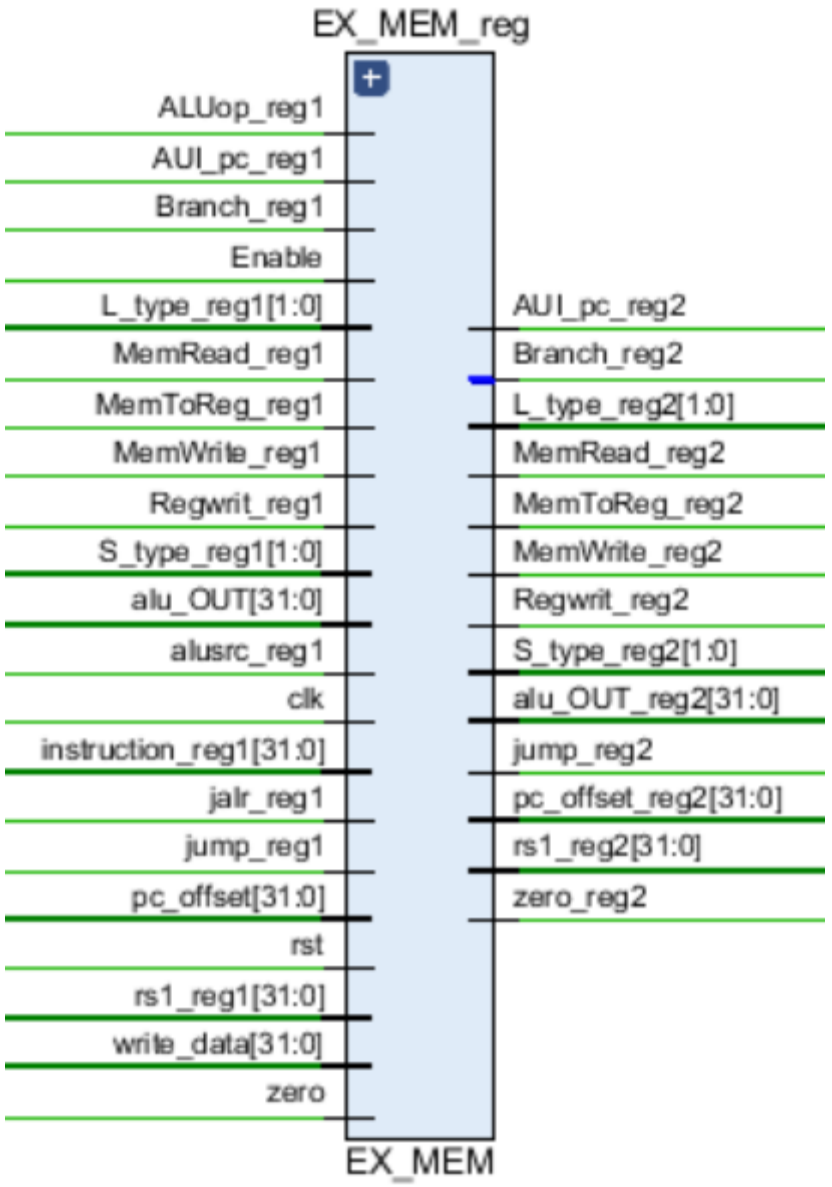

ifid1
IF_ID_Enable
clk
instr_in[31:0]
pc_in[31:0]
rst
instr_out[31:0]
pc_out[31:0]
IF_ID

7

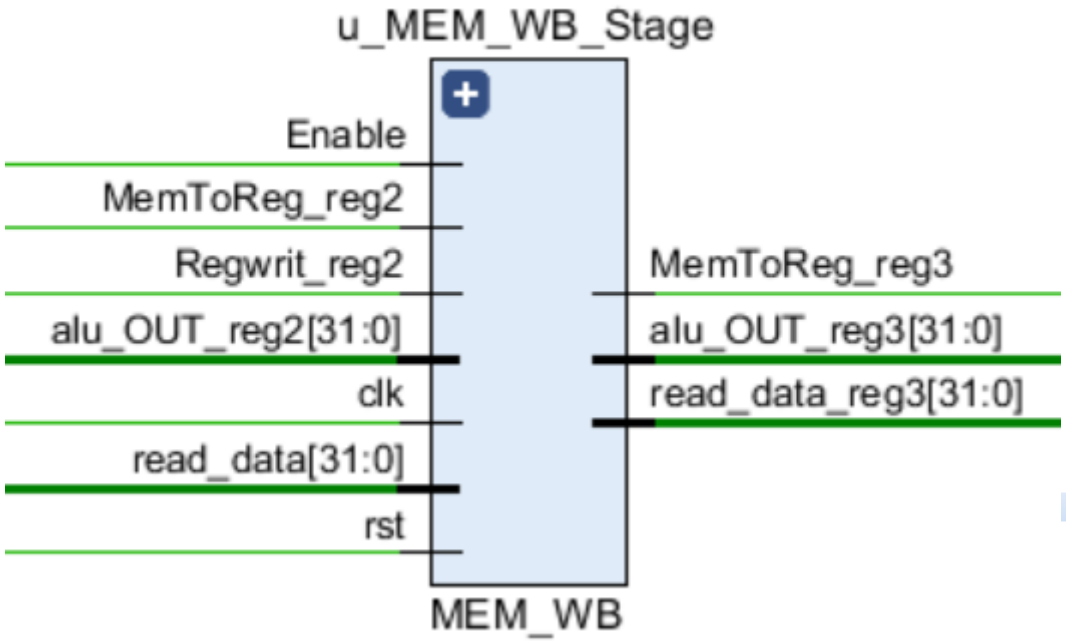
ID_EX :


r_IDEX1
ALUop[4:0]
AUI_pc
Branch
Im_mux[31:0]
L_type[1:0]
MemRead
MemToReg
MemWrite
Regwrit
S_type[1:0]
alusrc
clk
enable
instruction[31:0]
jalr
jump
pc[31:0]
rst
rs0[31:0]
rs1[31:0]
ALUop_reg1[4:0]
AUI_pc_reg1
Branch_reg1
L_type_reg1[1:0]
MemRead_reg1
MemToReg_reg1
MemWrite_reg1
Regwrit_reg1
S_type_reg1[1:0]
alusrc_reg1
im_mux_reg1[31:0]
instruction_reg1[31:0]
jalr_reg1
jump_reg1
pc_reg1[31:0]
rs0_reg1[31:0]
rs1_reg1[31:0]
ID_EX

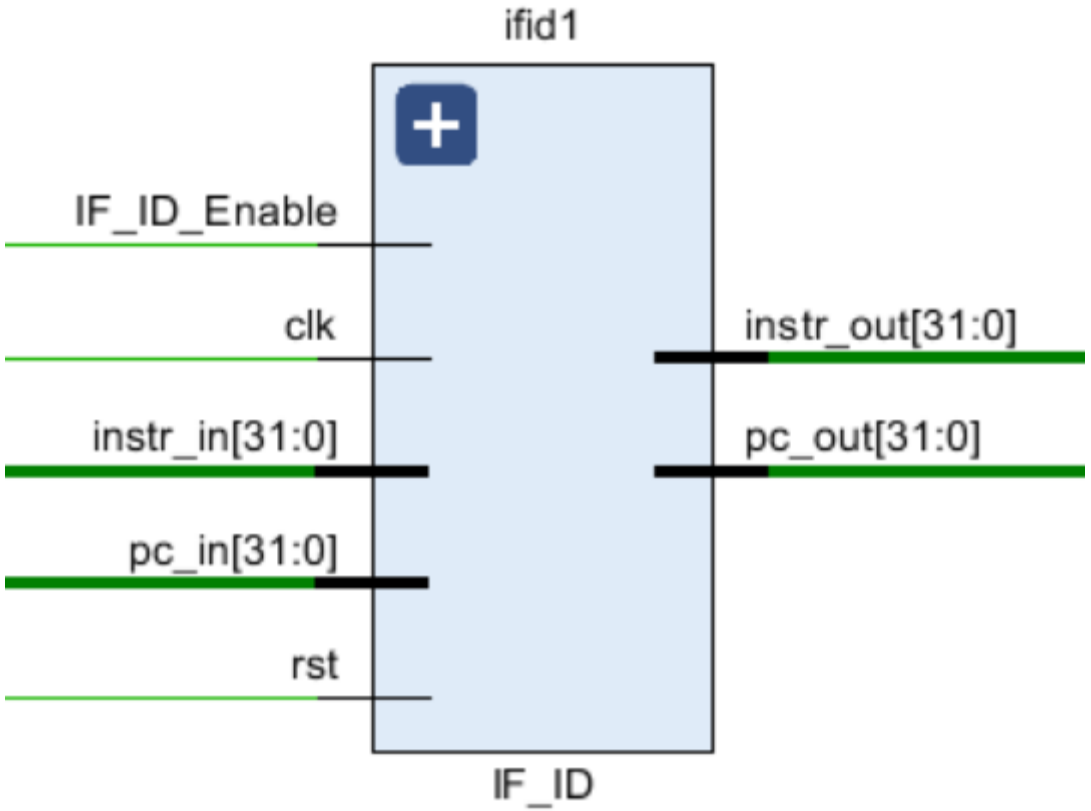
EX_MEM :


EX_MEM_reg
ALUop_reg1
AUI_pc_reg1
Branch_reg1
Enable
L_type_reg1[1:0]
MemRead_reg1
MemToReg_reg1
MemWrite_reg1
Regwrit_reg1
S_type_reg1[1:0]
alu_OUT[31:0]
alusrc_reg1
clk
instruction_reg1[31:0]
jalr_reg1
jump_reg1
pc_offset[31:0]
rst
rs1_reg1[31:0]
write_data[31:0]
zero
AUI_pc_reg2
Branch_reg2
L_type_reg2[1:0]
MemRead_reg2
MemToReg_reg2
MemWrite_reg2
Regwrit_reg2
S_type_reg2[1:0]
alu_OUT_reg2[31:0]
jump_reg2
pc_offset_reg2[31:0]
rs1_reg2[31:0]
zero_reg2
EX_MEM

MEM_WB :


u_MEM_WB_Stage
Enable
MemToReg_reg2
Regwrit_reg2
alu_OUT_reg2[31:0]
clk
read_data[31:0]
rst
MemToReg_reg3
alu_OUT_reg3[31:0]
read_data_reg3[31:0]
MEM_WB

IF_ID :


ifid1
IF_ID_Enable
clk
instr_in[31:0]
pc_in[31:0]
rst
instr_out[31:0]
pc_out[31:0]
IF_ID

Thank you

Muhammad Farhan Shah

farhaanshah336@gmail.com