PROJECT: DESIGN OF A SIMPLE PROCESSOR IN XILINX VIVADO

INTRODUCTION:

Processor do all the computational tasks of a hardware system. It is divided into some subsystems to do all the work such as control unit, ALU, general purpose register, instruction register, memory, input-output port etc.

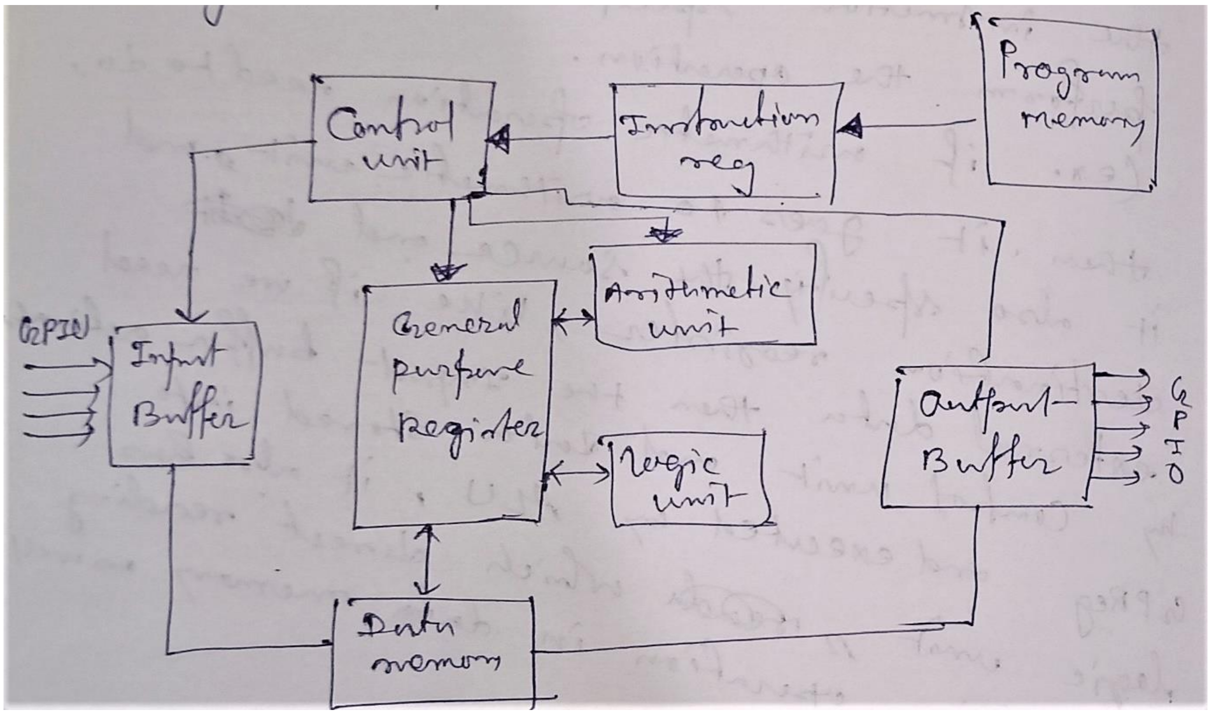

Figure 1: Block diagram of Processor sub-system

The main purpose of the control unit is to fetch the instruction from instruction register and by decoding and executing the instruction, it will send the signal to different blocks to perform the specified operation (like if we need to do arithmetic operation then it will go to the arithmetic block, to do logical operation it will go to the logical block or if data needed from external port, then it will be stored in data memory and then perform the task.

All the specification of operation with the source and destination register is provided in an Instruction. It is 32-bit size in our processor. The instructions are stored in program memory and the instruction register stored the next instruction, which will be executed by the processor.

The data memory is used to store the temporary data and as well as data coming from input port or going to output port. As we can not access the general-purpose register directly from external, we use data memory.

INSTRUCTION SPECIFICATION:

The instruction is segmented in different field, from MSB side first 5-bit[31:27] used to specify the type of operation, next 5-bit[26:22] specify the destination register address, [21:17] bit specify the 1st source register address, [16] bit specify the mode selection, if mode selection bit "1" then immediate operation will be done and rest 16 bit [15:0] will act as immediate data or address and if it is "0" then register based operation will be performed, [15:11] bit will specify the 2nd source register address and [10:0] bits are left unused. Since the operation type is specified by 5-bit, we can do 32-types (2^5) of operation in our processor. The address is also 5-bit length, so depth of general-purpose register will be 32 (i.e., 00000-11111) and data size is 16-bit so we need a general-purpose register of depth 32 and each location is of 16-bit size.

Table 1:Instruction field (Mode selection :0)

[31-27] bit (oper_type)	[26-22] bit (rdst)	[21-17] bit (rsrc1)	16th bit (imm_mode = 0)	[15-11] bit (rsrc2)	[10-0] bit unused

Table 2 Instruction field (mode selection:1)

[31-27] bit (oper_type)	[26-22] bit (rdst)	[21-17] bit (rsrc1)	16th bit (imm_mode = 1)	[15-0] bit immediate data/address (isrc)

Table 3 Instruction field (jump condition)

[31-27] bit (oper_type)	[26-16] bit unused	[15:0] jump address

FLAGS:

CARRY (C): Used in addition operation, the MSB of addition of temp_sum.

ZERO (Z): If all the bits are zero then zero flag is set otherwise reset.

SIGN (S): The MSB of result. If sign bit is "1" then result is negative, if sign bit is "0" then result is positive.

OVERFLOW (OV): If the result is not as expected then overflow bit is set, otherwise it is reset.

OPERATION TABLE:

OPCODE NAME	OPCODE	OPERATION
MOVSGPR	5’B00000	MOVE DATA SGPR TO GPR
MOV	5’B00001	MOVE DATA FROM REG TO REG IN GPR
ADD	5’B00010	ADD TWO DATA
SUB	5’B00011	SUB TWO DATA
MUL	5’B00100	MULTIPLY TWO DATA
ROR	5’B00101	BITWISE OR
RAND	5’B00110	BITWISE AND
RXOR	5’B00111	BITWISE XOR
RXNOR	5’B01000	BITWISE XNOR
RNAND	5’B01001	BITWISE NAND
RNOR	5’B01010	BITWISE NOR
RNOT	5’B01011	BITWISE NOT
STOREREG	5’B01101	STORE CONTENT OF REGISTER TO DM
STOREDIN	5’B01110	STORE CONTENT OF INPUT TO DM
SENDOUT	5’B01111	SEND DATA FROM DM TO OUTPUT
SENDREG	5’B10001	SEND DATA FROM DM TO GPR
JUMP	5’B10010	JUMP TO SPECIFIC LOCATION
JCARRY	5’B10011	JUMP WITH CARRY = 1
JNOCARRY	5’B10100	JUMP WITH CARRY = 0
JSIGN	5’B10101	JUMP WITH SIGN = 1
JNOSIGN	5’B10110	JUMP WITH SIGN = 0
JZERO	5’B10111	JUMP WITH ZERO = 1
JNOZERO	5’B11000	JUMP WITH ZERO = 0
JOVERFLOW	5’B11001	JUMP WITH OVERFLOW = 1
JNOOVERFLOW	5’B11010	JUMP WITH OVERFLOW = 0
HALT	5’B11011	STOP

FLOW CHART OF CONTROL IN PROCESSOR [FSM]

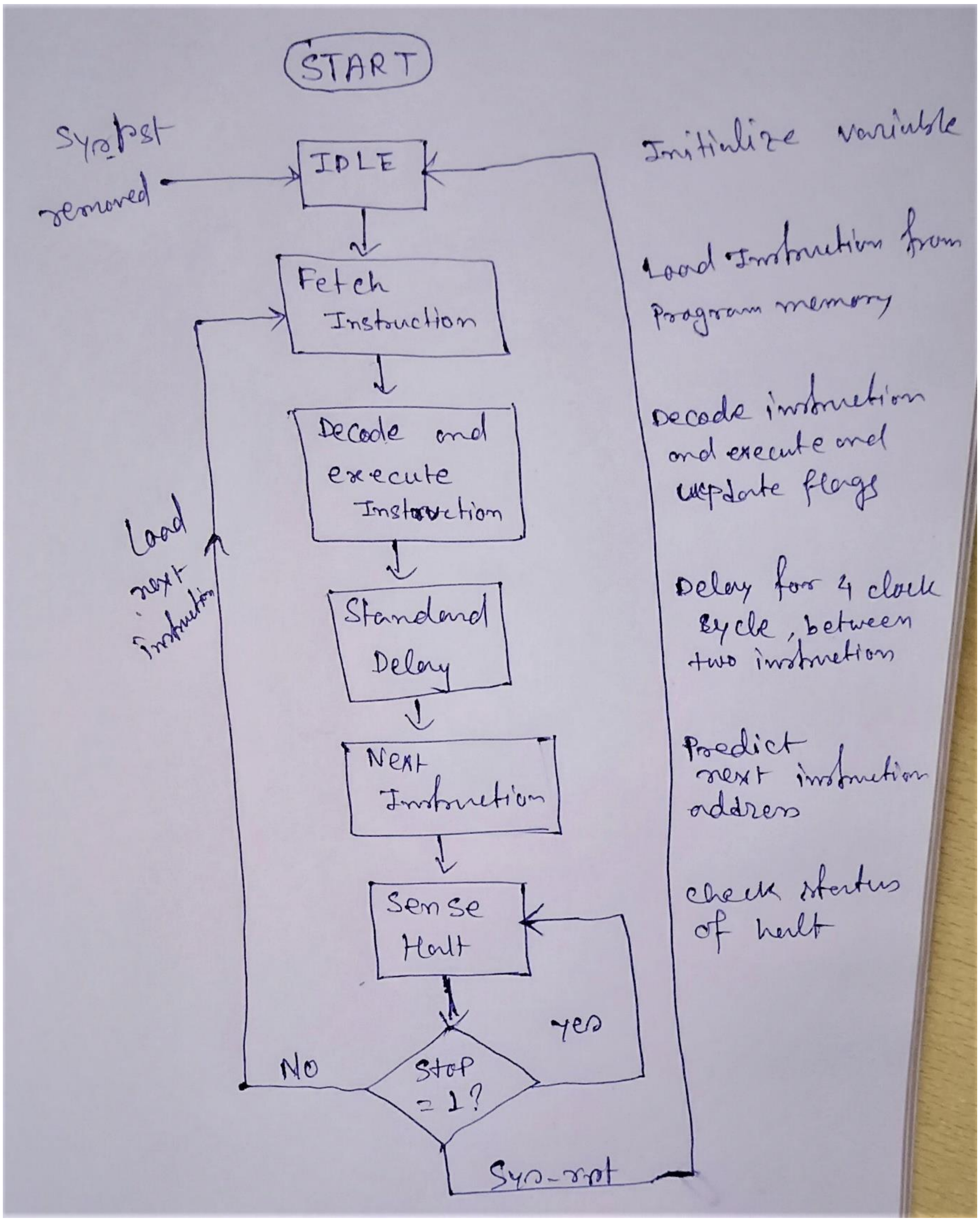
START
Sys_rst removed
IDLE
Initialize variable
Fetch Instruction
Load Instruction from Program memory
Decode and execute Instruction
Decode instruction and execute and update flags
Load next instruction
Standard Delay
Delay for 4 clock cycle, between two instruction
Next Instruction
Predict next instruction address
Sense Halt
check status of halt
yes
NO
Stop = 1?
Sys-rst

Figure 2: FSM of control unit

To design the control unit, we segmented the whole process into some states. At first at the positive edge of the clock if system reset is on then control unit will be in idle state and the instruction register will set to zero and the program counter is pointed to the 0th location of program memory. Now when the reset signal is removed the control signal start the processing and it will move to instruction fetching state. In this state the instruction of 0th location of program memory, (where the program counter pointed) will loaded to IR and it will move to decode and execute state. To do the decoding and executing comfortably I introduce a delay block. It will add delay the four-clock cycle. After that control unit move to new instruction state. Here it will look for any jump condition or not. If jump condition is satisfied then it will move the program counter to that address otherwise it will increase the program counter by one and move to last state of the state machine i.e., sense halt. Here if the stop signal is 0 then it move to fetch instruction state and if stop 1 then it will stop the process and wait for next reset signal.