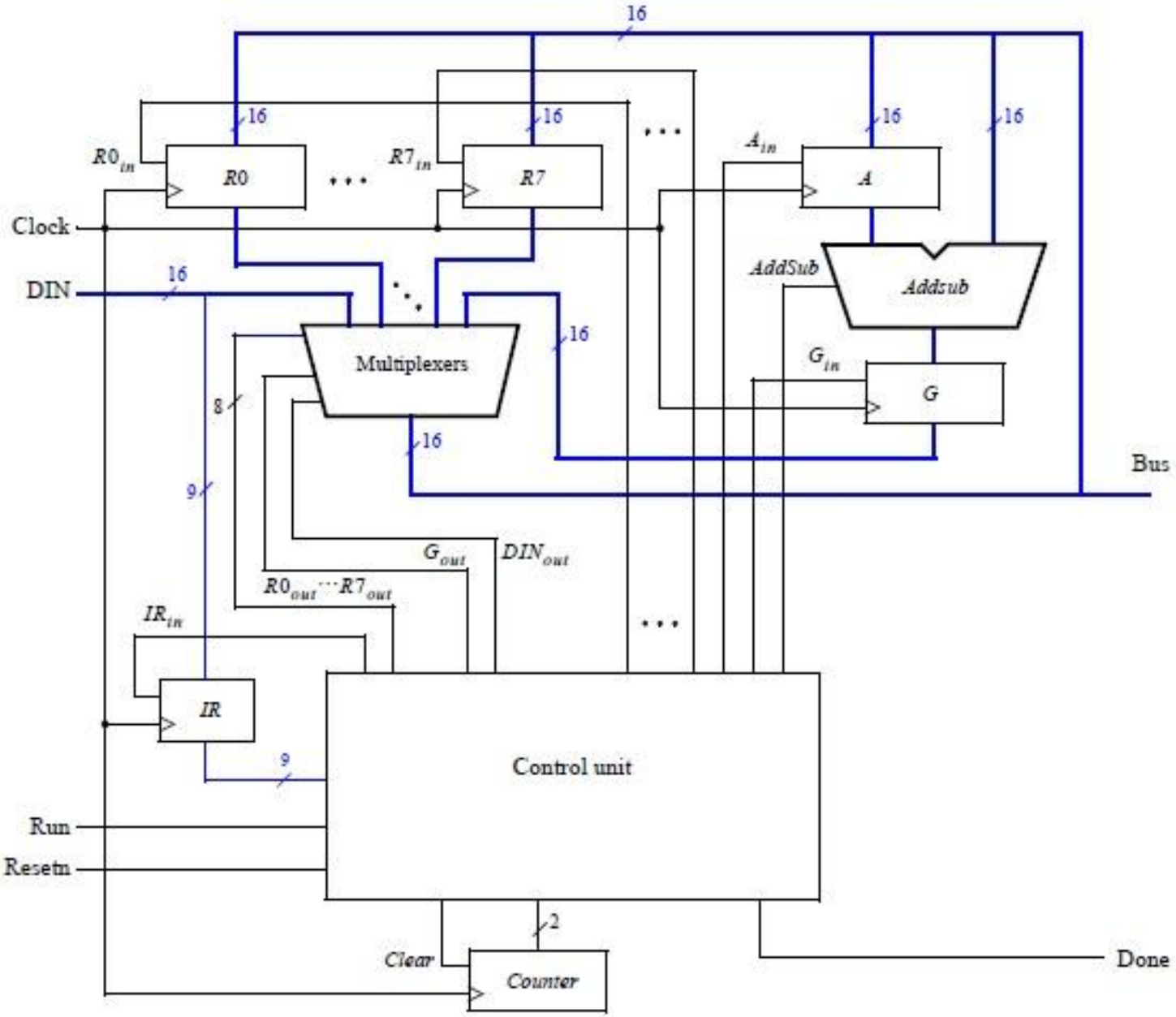
A Simple Processor

Xiaolu Wang

Abstract

- This project bases on the Laboratory Exercise 9&10
- The designed processor can perfrom a lot of functions including IIR Filter (shown below).


16
16
16
16
16
$R0_{in}$
R0
$R7_{in}$
R7
A_{in}
A
Clock
DIN
16
AddSub
Addsub
Multiplexers
16
G_{in}
G
8
16
Bus
9
G_{out}
DIN_{out}
$R0_{out} \cdots R7_{out}$
IR_{in}
IR
9
Control unit
Run
Resetn
2
Clear
Counter
Done

Run signal is asserted=>the processor starts

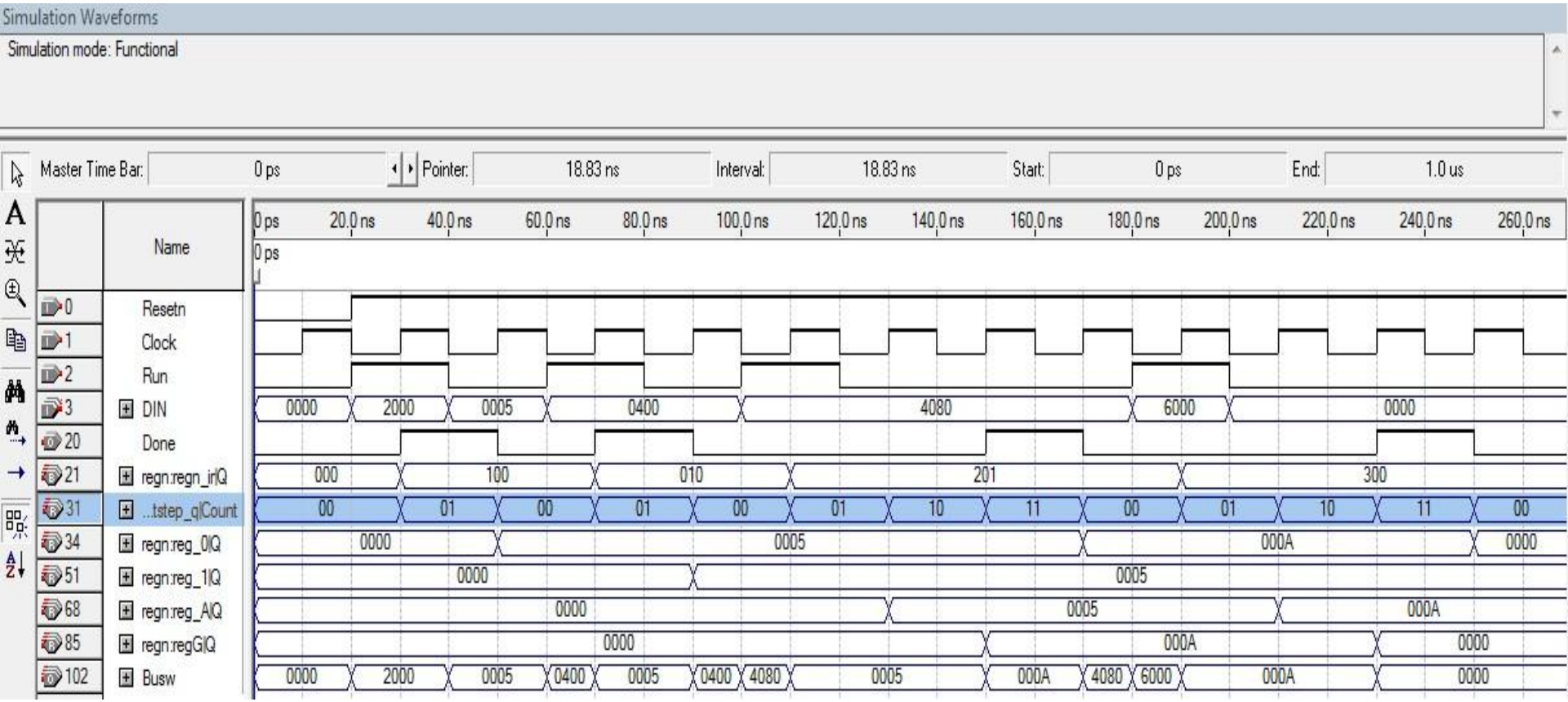
In time step 0, only IR is asserted.

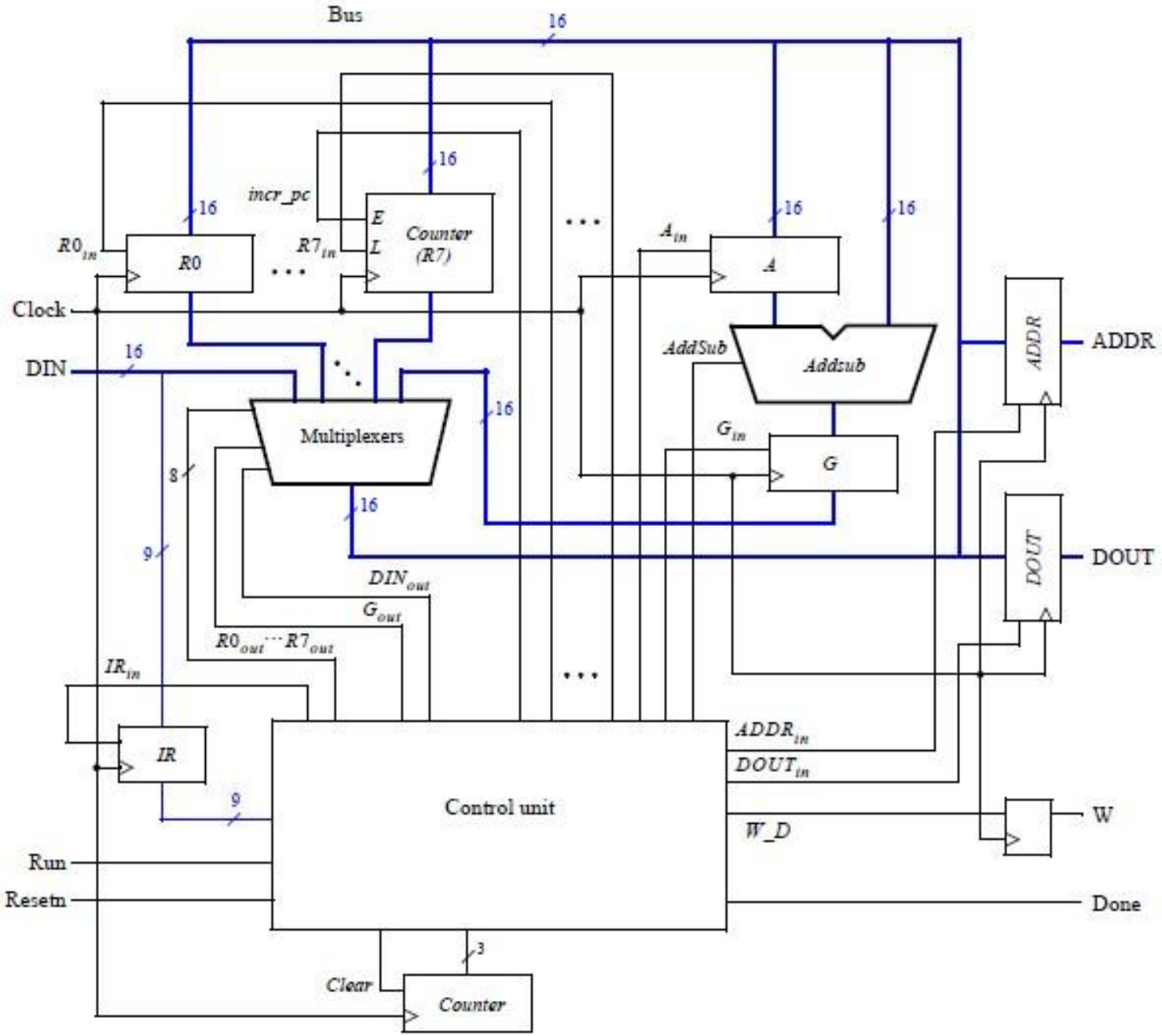
The processor finished=>Done output

Operation	Function performed
mv Rx,Ry	$Rx \leftarrow [Ry]$
mvi Rx,#D	$Rx \leftarrow D$
add Rx, Ry	$Rx \leftarrow [Rx] + [Ry]$
sub Rx, Ry	$Rx \leftarrow [Rx] - [Ry]$

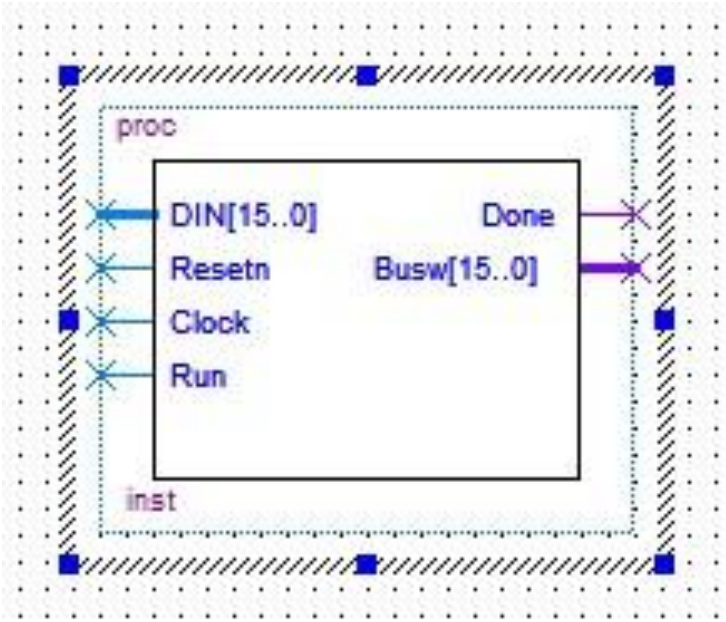
T_1	T_2	T_3
$RY_{out}, RX_{in},$ *Done*		
$DIN_{out}, RX_{in},$ *Done*		
RX_{out}, A_{in}	RY_{out}, G_{in}	$G_{out}, RX_{in},$ *Done*
RX_{out}, A_{in}	$RY_{out}, G_{in},$ *AddSub*	$G_{out}, RX_{in},$ *Done*

Then add Registers , Count , Addsub modules

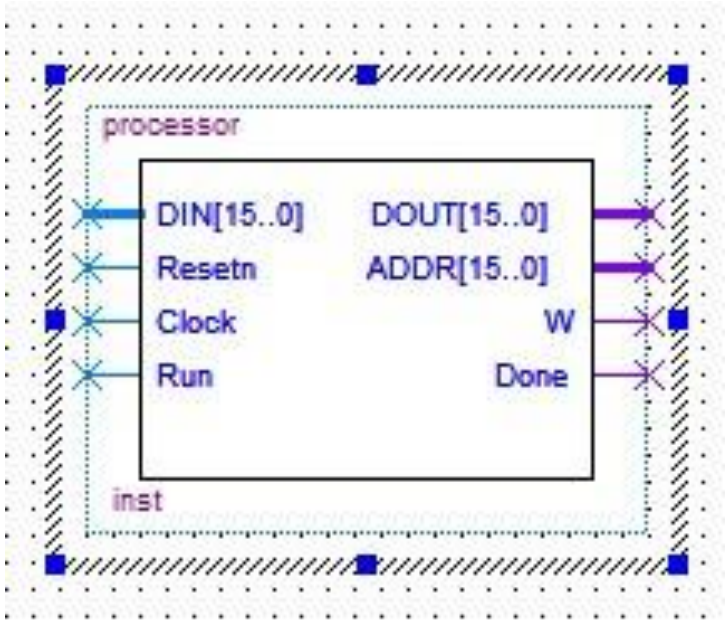



Bus
16
16
incr_pc
E
L
Counter
(R7)
16
$R0_{in}$
R0
$R7_{in}$
A_{in}
A
16
16
Clock
DIN
16
AddSub
Addzub
ADDR
ADDR
Multiplexers
16
G_{in}
G
8
16
9
DOUT
DOUT
DIN_{out}
G_{out}
$R0_{out}$ ··· $R7_{out}$
IR_{in}
IR
$ADDR_{in}$
$DOUT_{in}$
9
Control unit
W_D
W
Run
Resetn
Done
3
Clear
Counter

Symbol Generated

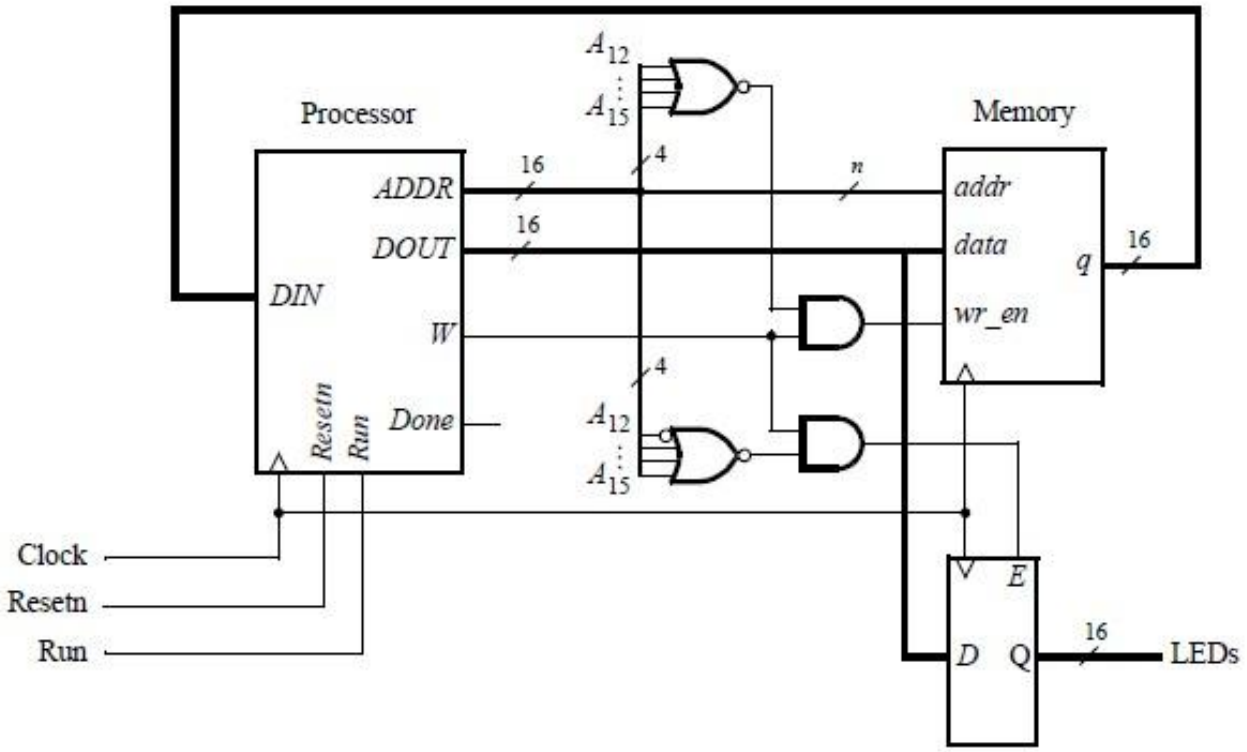


First processor



Enhanced processor

Synchronous memory



	T0	T1	T2	T3	T4	T5
mv	R7=>ADDR; R7+1	Wait for rising edge clock of memory	DIN=> IR	RY=>RX DONE		
mvi				DIN=>RX R7+1 DONE		
add				RX=>A	RX+RY=>G	G=>RX DONE
sub				RX=>A	RX-RY=>G	G=>RX DONE

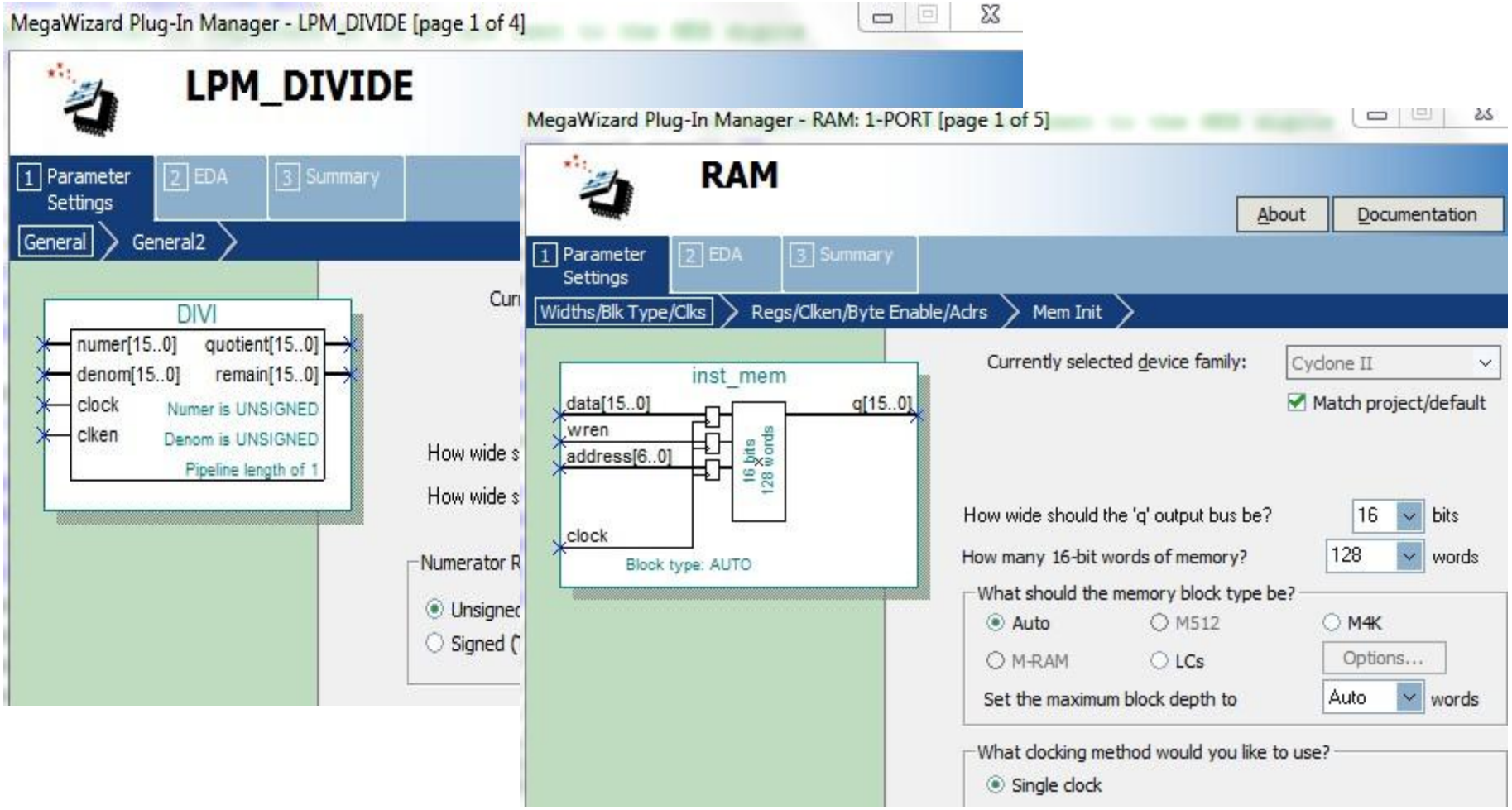
	T0	T1	T2	T3	T4	T5
ld	R7=>ADDR; R7+1	Wait for rising edge clock of memory	DIN=> IR	RY=>ADDR	Wait for rising edge clock of memory	DIN=>RX DONE
st				RY=>ADDR	RX=>DOUT; W	DONE; Wait for rising edge clock of memory
mvnz				Z=0,RY=>RX,ELSE 0 LOADED DONE		
divide				RX=>DG	RY=>BUS RX/RY	RX/RY=>RX

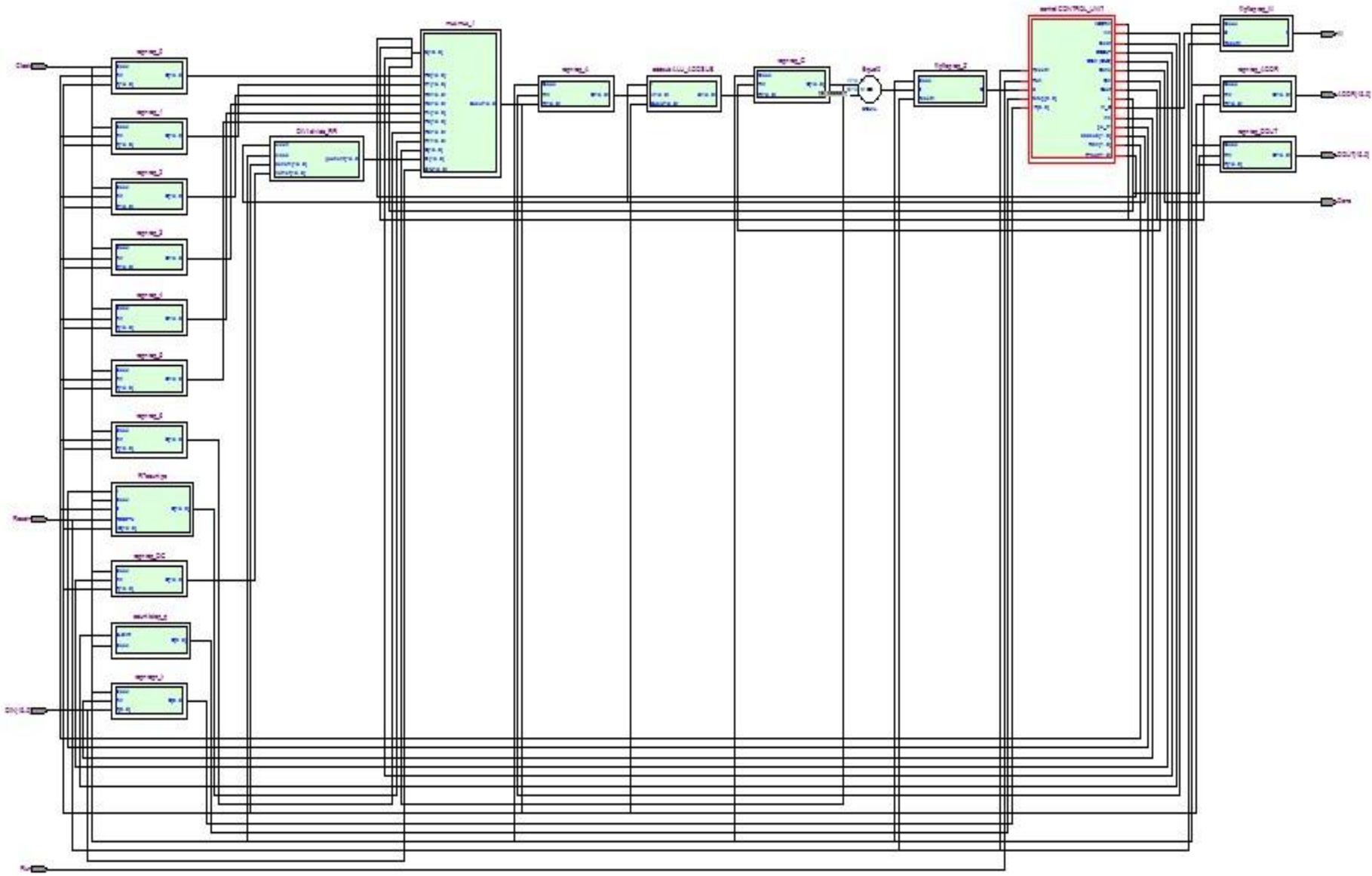
The instruction **mvnz** (move if
not zero) allows a **mv** operation to be executed only under a certain condition; the condition is that the current
contents of register *G* are not equal to 0.

- Divide Rx,Ry Rx=Rx/Ry

Operation	Function performed
ld Rx,$[Ry]$	$Rx \leftarrow [[Ry]]$
st Rx,$[Ry]$	$[Ry] \leftarrow [Rx]$
mvnz Rx, Ry	if G != 0, $Rx \leftarrow [Ry]$

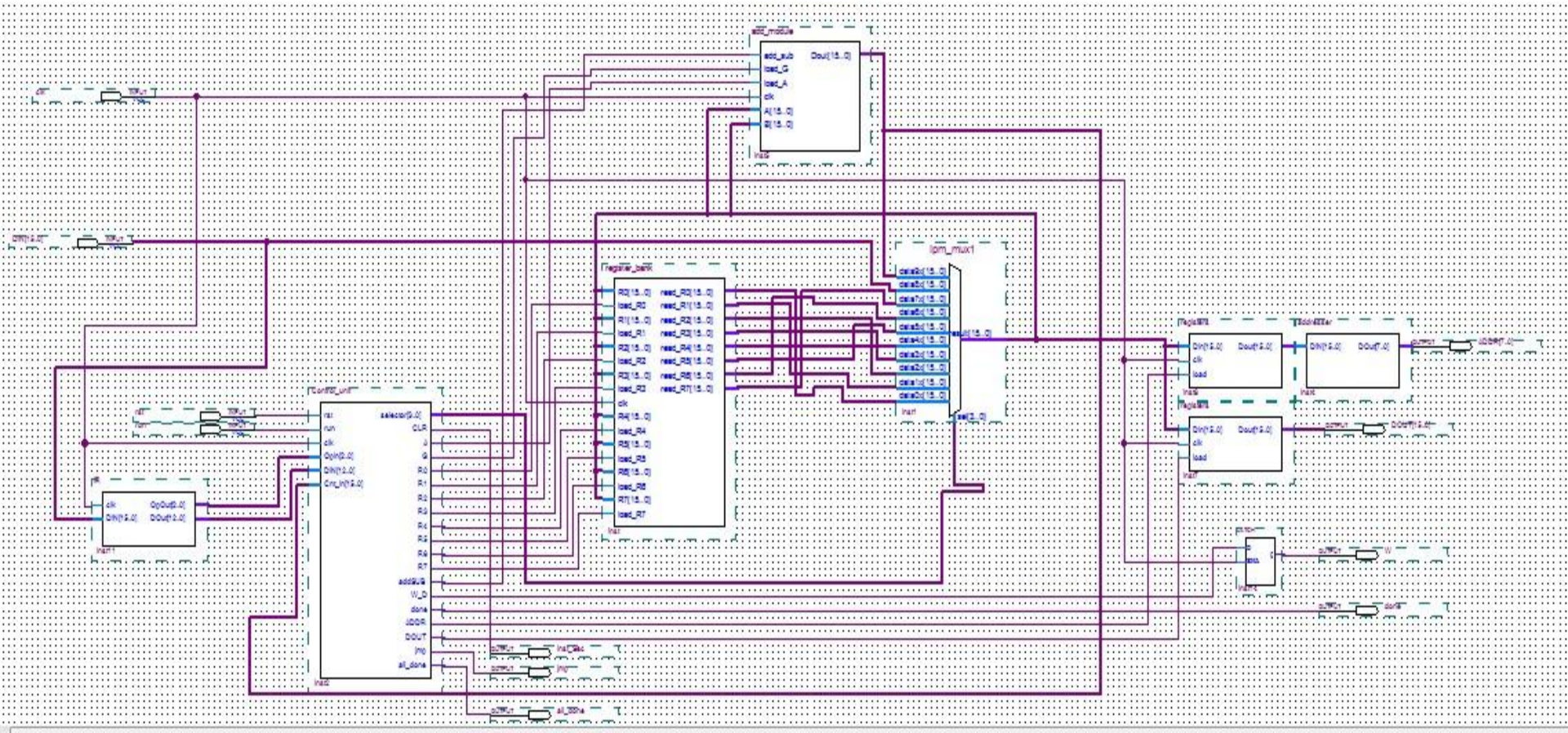
Momery & Divide





RTL View


1addsub.vhd
1control.vhd
1count.vhd
1flipflop.vhd
1multiplexer.vhd
1proce.vhd



Utilize the processor

- IIR low-pass filter

$$H(z) = \frac{z^{-1}}{1 - 0.75z^{-1}}$$

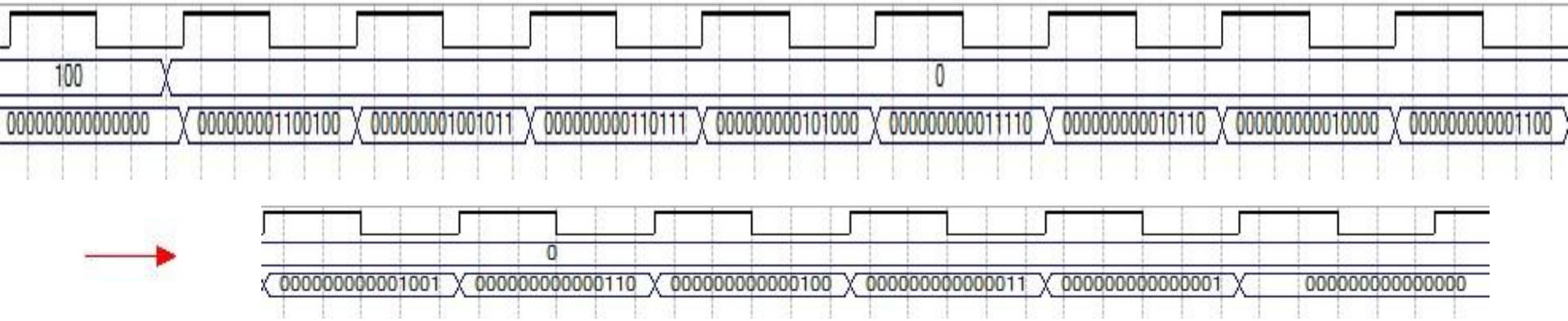
```
WAIT UNTIL clk = '1';
x  <= x_in;
y  <= x + y / 4 + y / 2;
```

Momery initialization file(.mif)

- 0011010000000000 MVI R5
- 0000000000000100 #4
- 0000110100000000 MV R3,R2
- 0100100100000000 ADD R2,R3
- 0100100110000000 ADD R2,R2
- 1110101010000000 DIV R2,R5

Expected result

- 0010100000000000 MVI R2
- 0000000001100100 #100



100 on 7segment

Yout on LED

Read from switches into momery

- The number read is as the delay of loop all the instructions in momery

- Shown on DE2 board:

100 Scrolling across the 7-segment display

Switches to control the filter speed

- Thank you