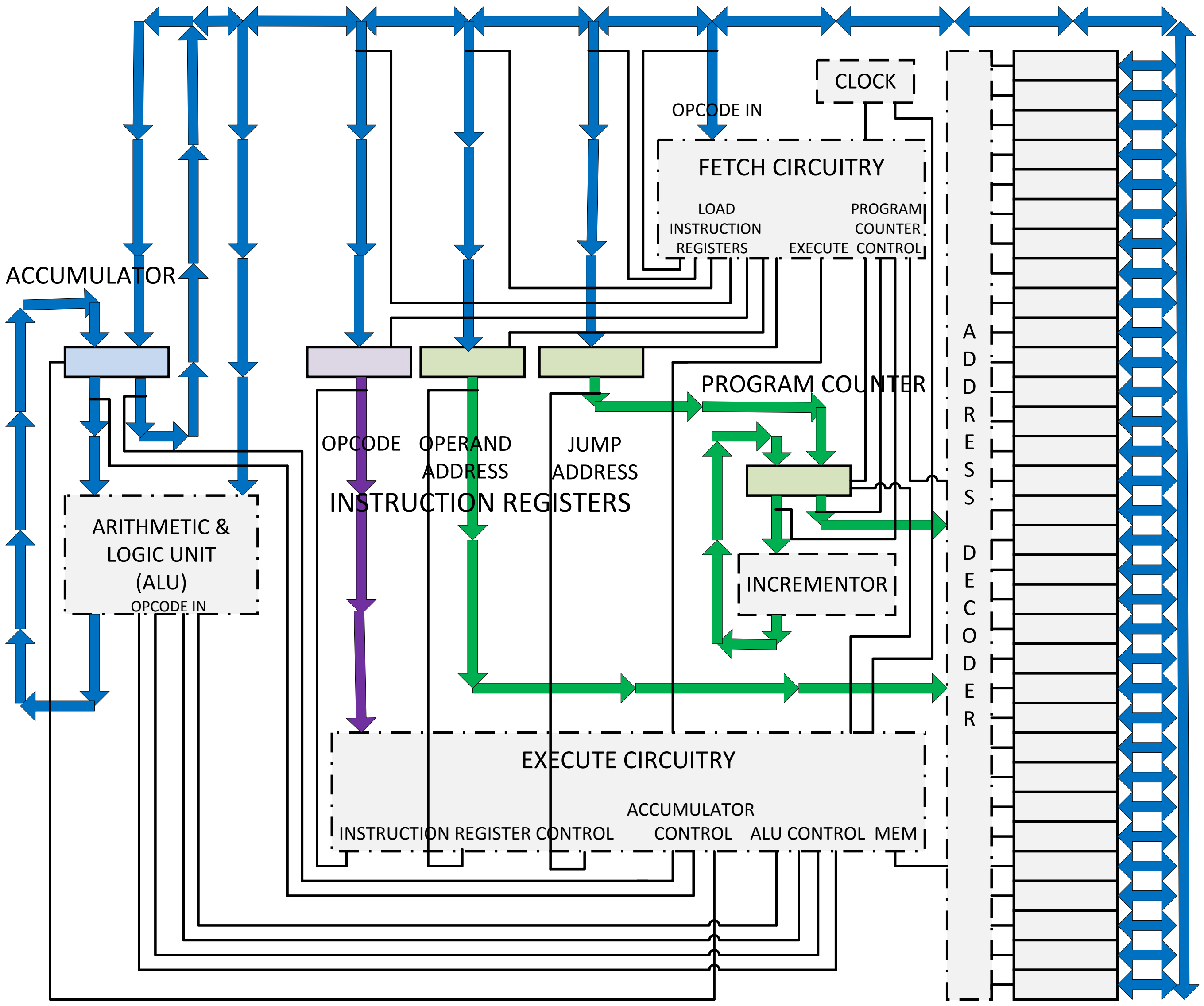
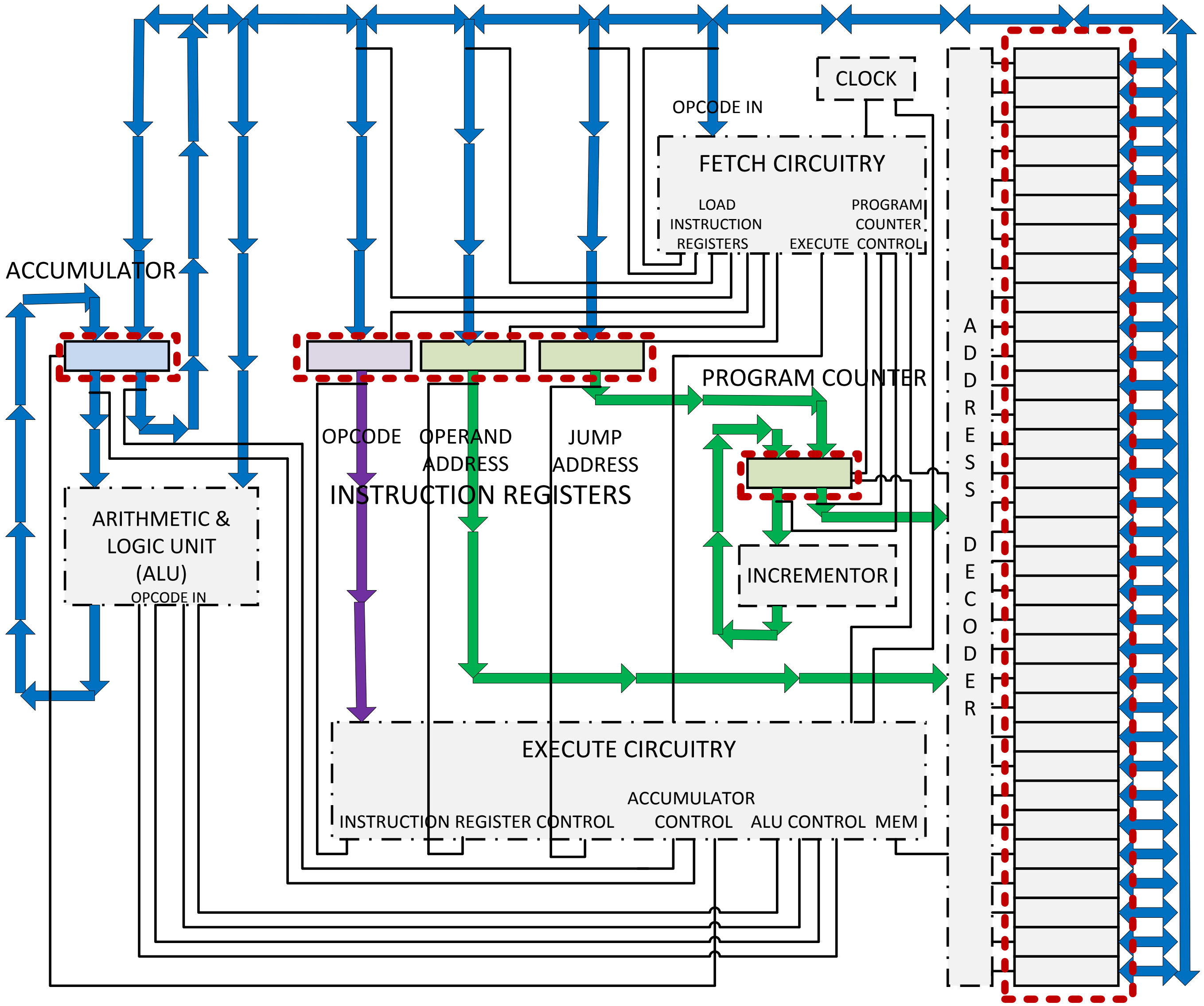

CLOCK
OPCODE IN
FETCH CIRCUITRY
LOAD INSTRUCTION REGISTERS
PROGRAM COUNTER CONTROL
EXECUTE
ACCUMULATOR
A D D R E S S D E C O D E R
PROGRAM COUNTER
OPCODE
OPERAND ADDRESS
JUMP ADDRESS
INSTRUCTION REGISTERS
ARITHMETIC & LOGIC UNIT (ALU)
OPCODE IN
INCREMENTOR
EXECUTE CIRCUITRY
INSTRUCTION REGISTER CONTROL
ACCUMULATOR CONTROL
ALU CONTROL
MEM

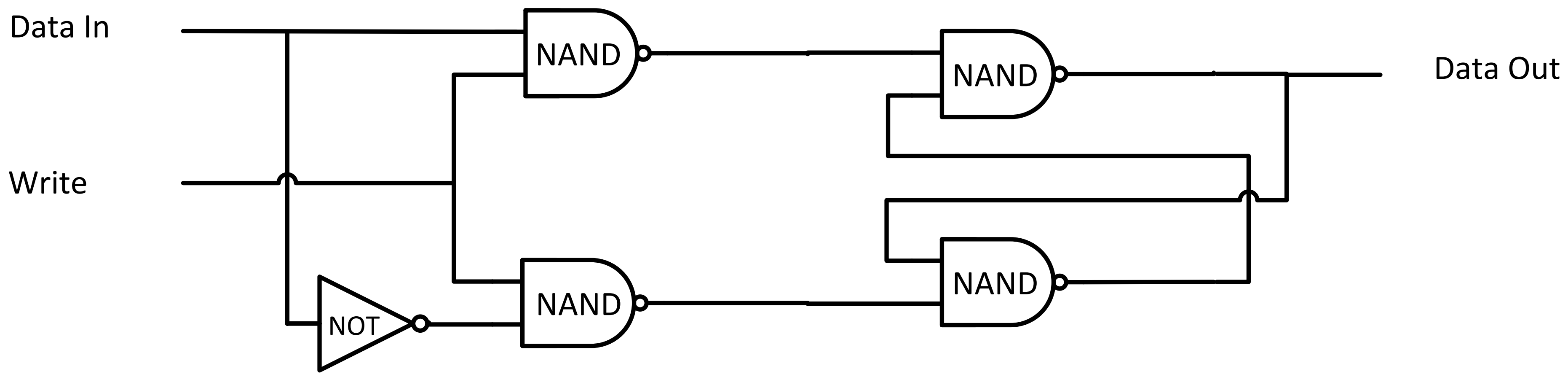
Memory



CLOCK
OPCODE IN
FETCH CIRCUITRY
LOAD INSTRUCTION REGISTERS
PROGRAM COUNTER
EXECUTE CONTROL
ACCUMULATOR
ADDRESS DECODER
PROGRAM COUNTER
OPCODE
OPERAND ADDRESS
JUMP ADDRESS
INSTRUCTION REGISTERS
ARITHMETIC & LOGIC UNIT (ALU)
OPCODE IN
INCREMENTOR
EXECUTE CIRCUITRY
INSTRUCTION REGISTER CONTROL
ACCUMULATOR CONTROL
ALU CONTROL
MEM

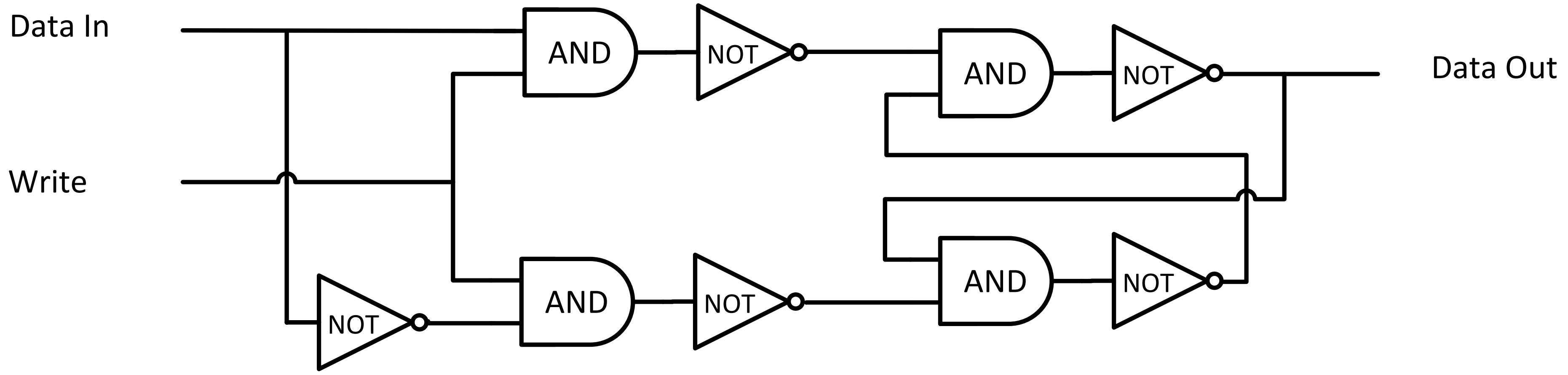
Memory or Register Bit

based on a “D Flip-Flop” --- other designs are more efficient



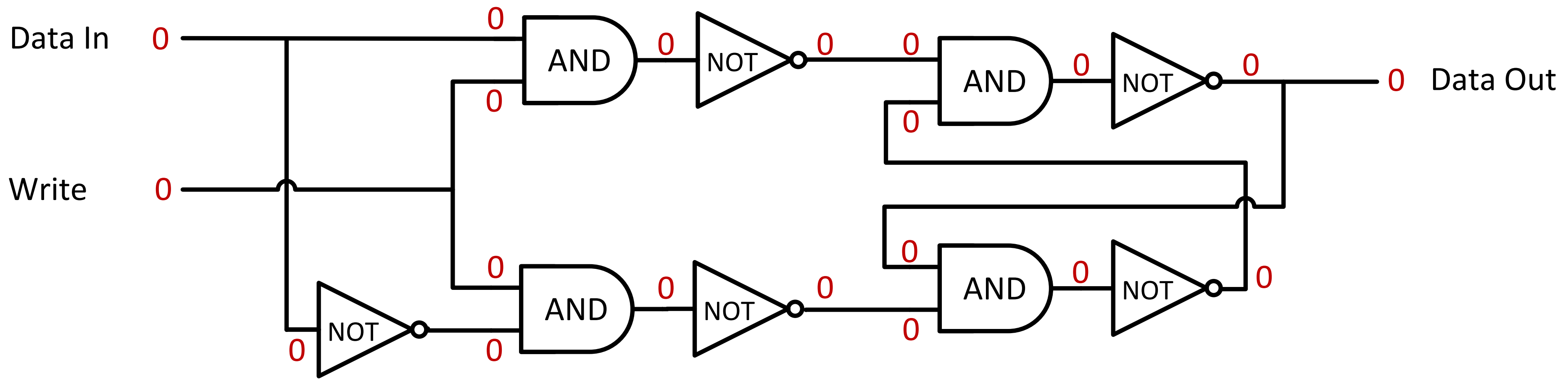
Memory or Register Bit

based on a “D Flip-Flop” --- other designs are more efficient



Memory or Register Bit

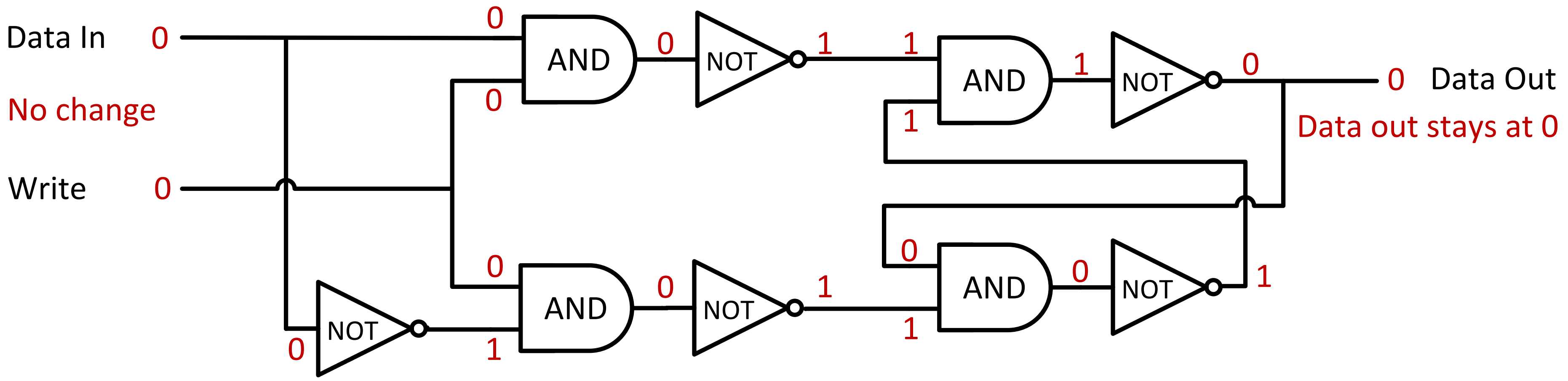
based on a “D Flip-Flop” --- other designs are more efficient



Power is just switched on – everything is at 0 momentarily.

Memory or Register Bit

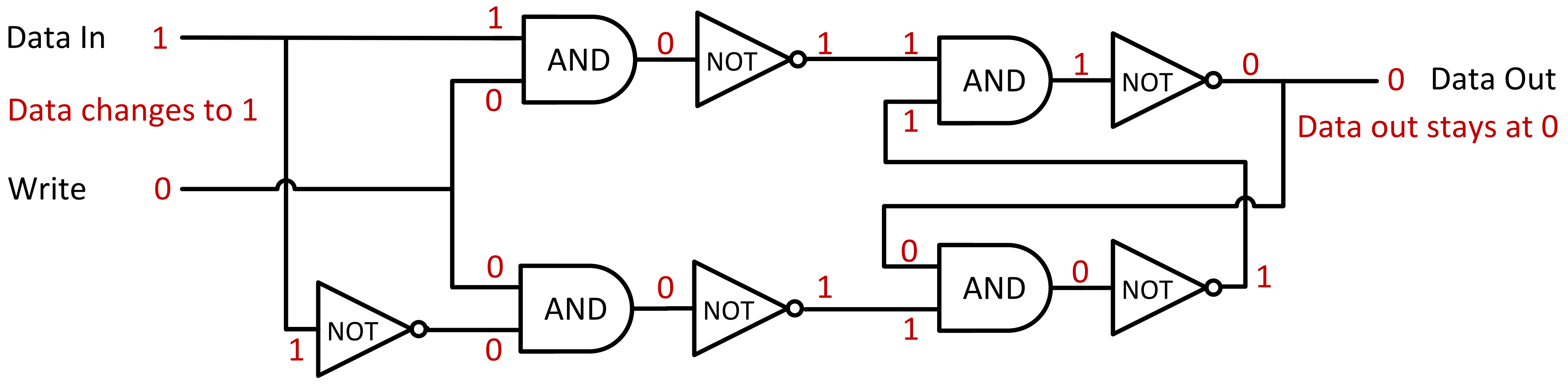
based on a “D Flip-Flop”



Logic gates activate – no input yet, and no change in output

Memory or Register Bit

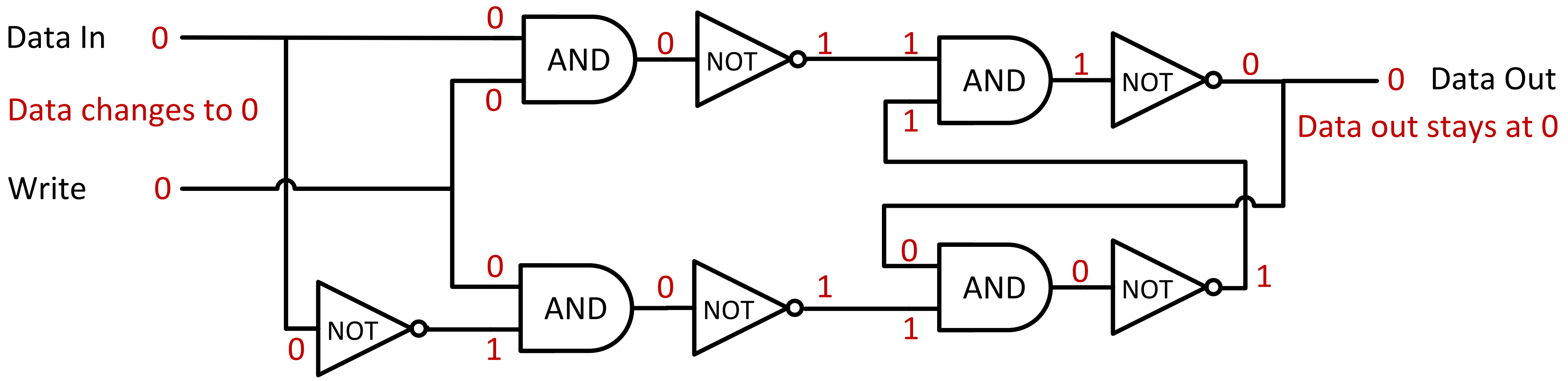
based on a “D Flip-Flop”



Data In changes, but Write stays at 0 -- no change in output

Memory or Register Bit

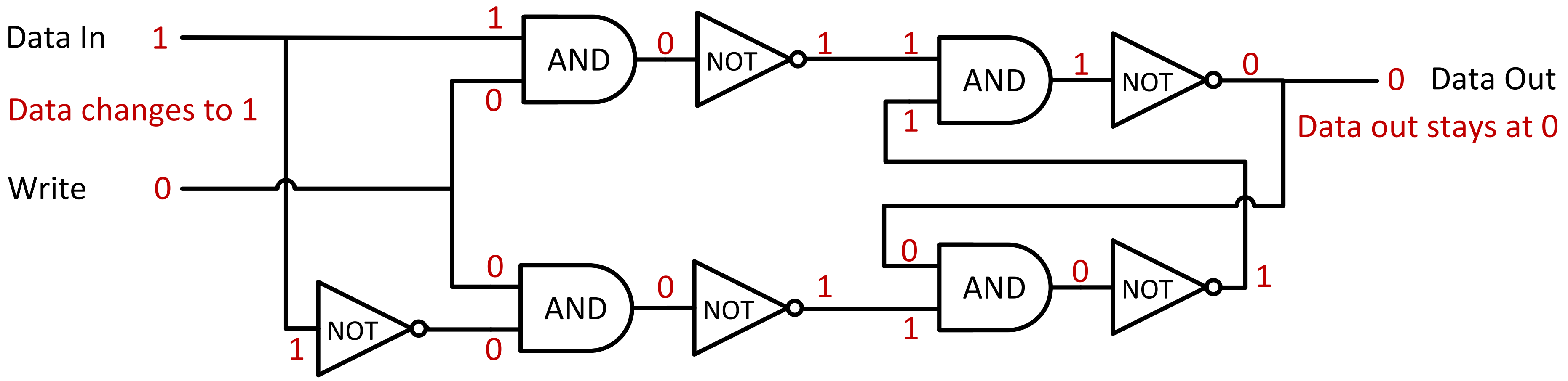
based on a “D Flip-Flop”



Data changes back, but Write stays at 0 -- no change in output

Memory or Register Bit

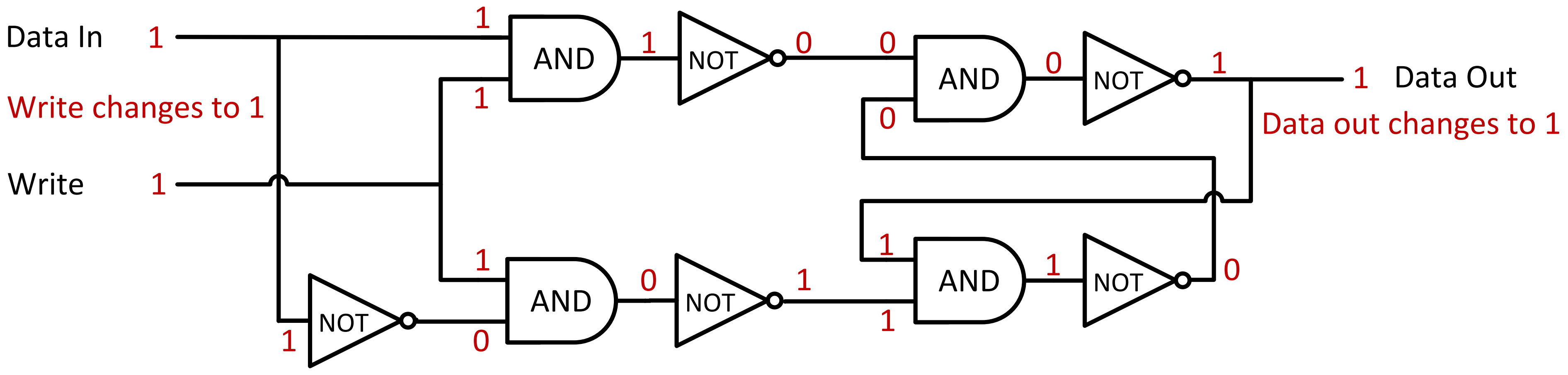
based on a “D Flip-Flop”



Data changes again, but Write stays at 0 -- still no change out

Memory or Register Bit

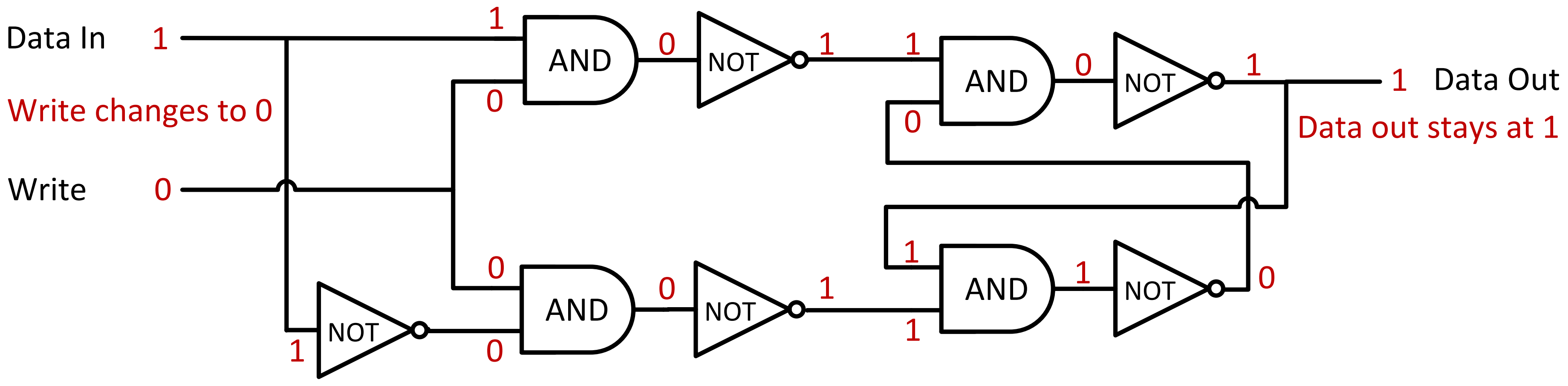
based on a “D Flip-Flop”



Now Write changes to 1 --- the output changes to 1

Memory or Register Bit

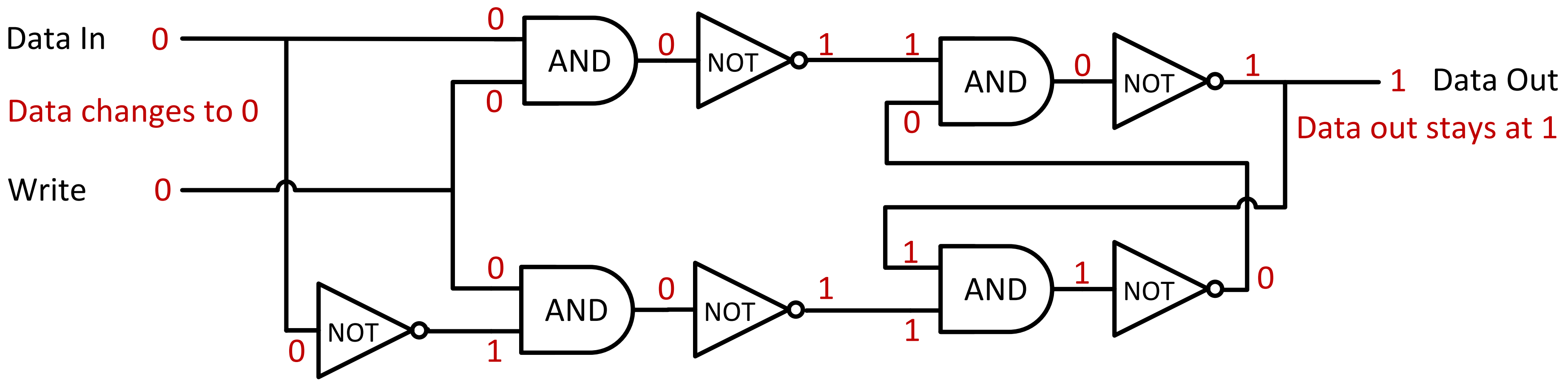
based on a “D Flip-Flop”



Write changes back to 0 --- the output stays at 1

Memory or Register Bit

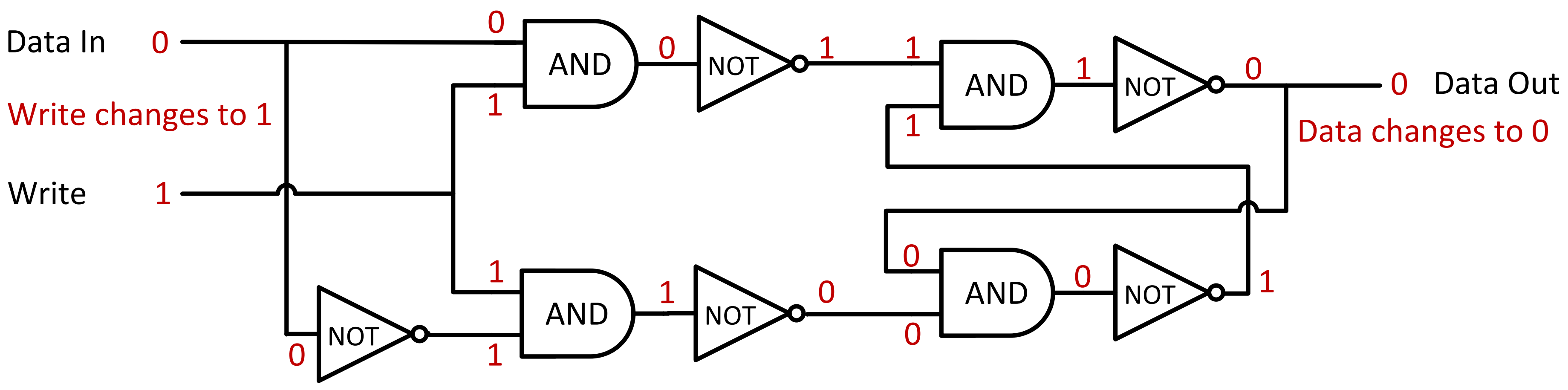
based on a "D Flip-Flop"



Data In also changes back to 0, but the output stays at 1 because Write is still 0.

Memory or Register Bit

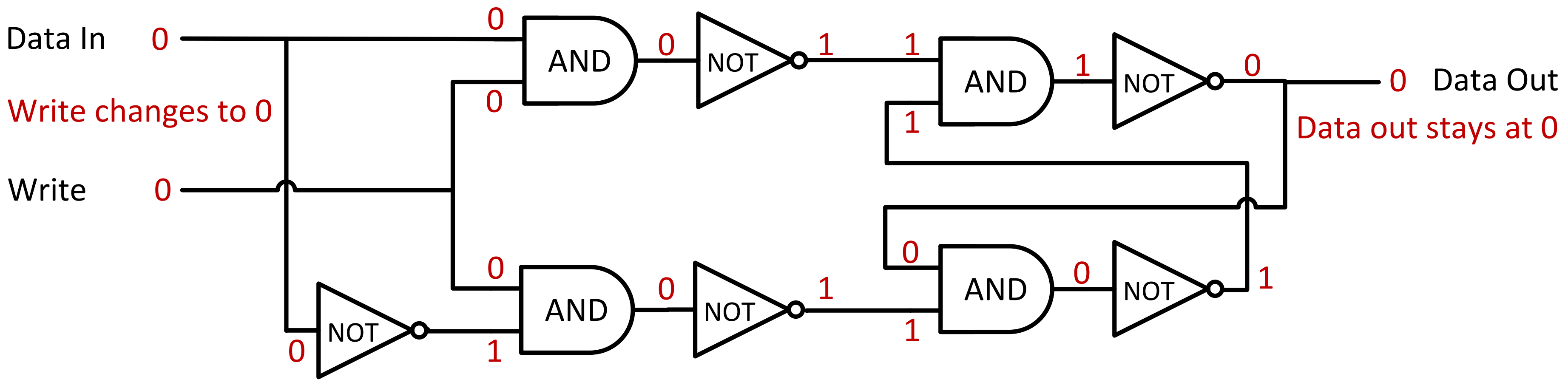
based on a “D Flip-Flop”



When Write changes to 1 --- the output changes to the input 0

Memory or Register Bit

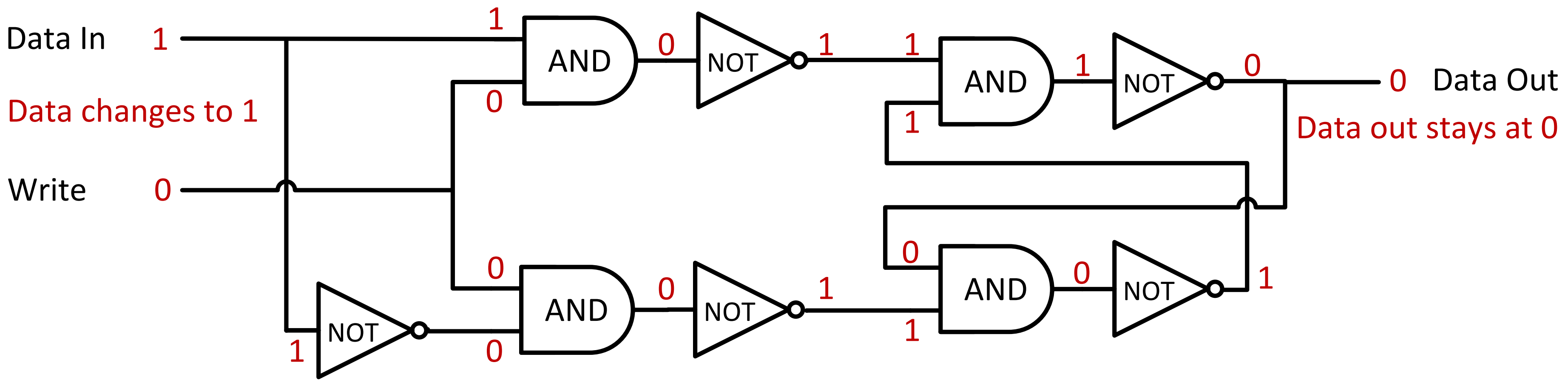
based on a “D Flip-Flop”



Write changes back to 0 --- the output will continue to hold it’s value, whatever it is.

Memory or Register Bit

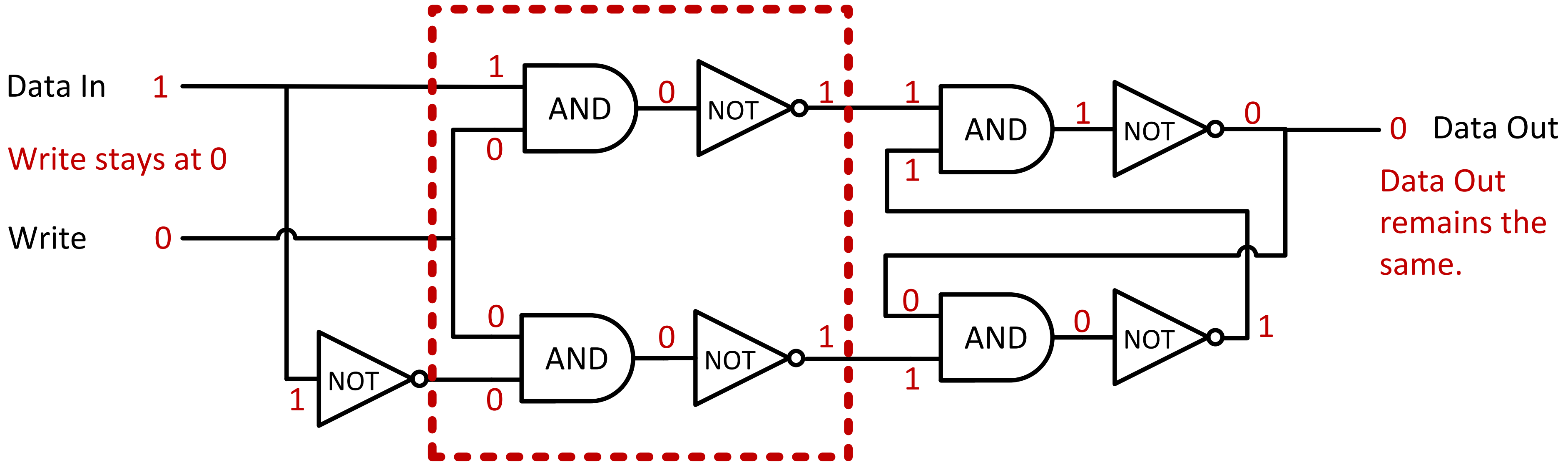
based on a “D Flip-Flop”



Even though Data In changes to 1 the output continues to hold it’s previous value.

Memory or Register Bit

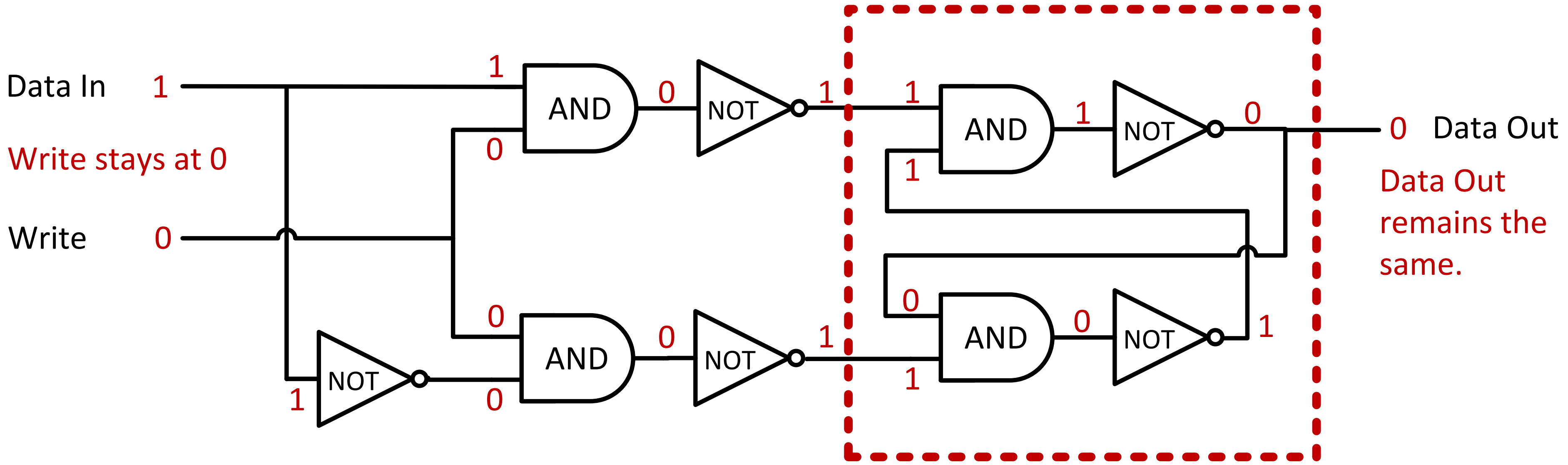
based on a “D Flip-Flop”



If Write stays at zero, these two pairs of AND and NOT gates will always output 1, no matter how the Data In changes. This means the Flip-Flop will hold its previous value.

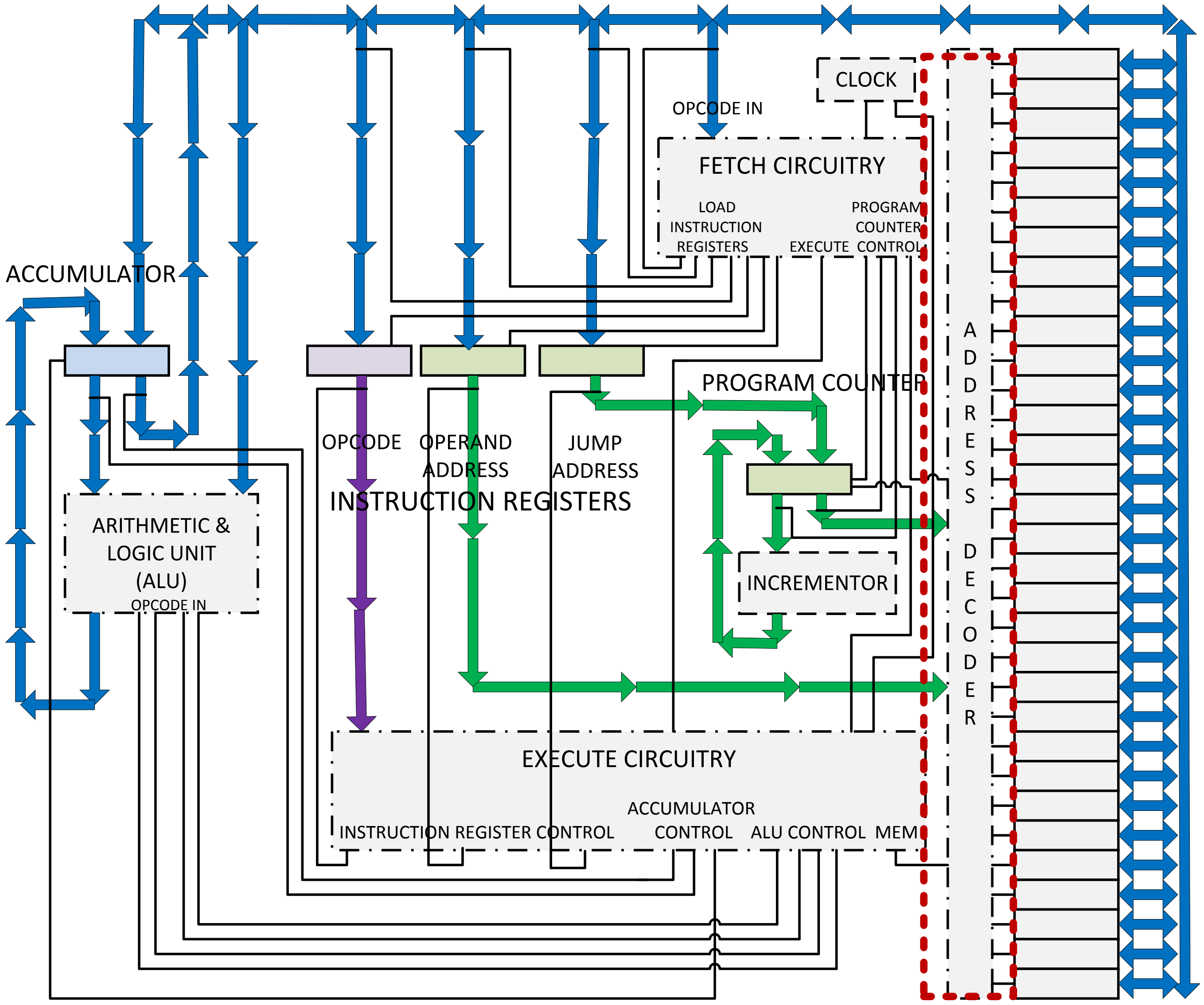
Memory or Register Bit

based on a “D Flip-Flop”



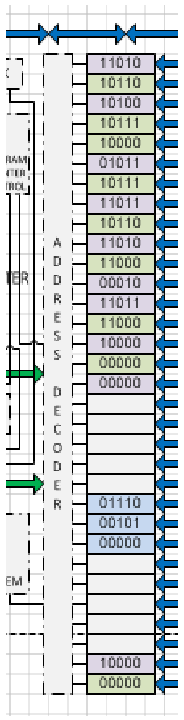
This is the Flip-Flop part.
As long as the two inputs from the NOT gates to the left are both 1's, the Flip-Flop will not change it’s output.

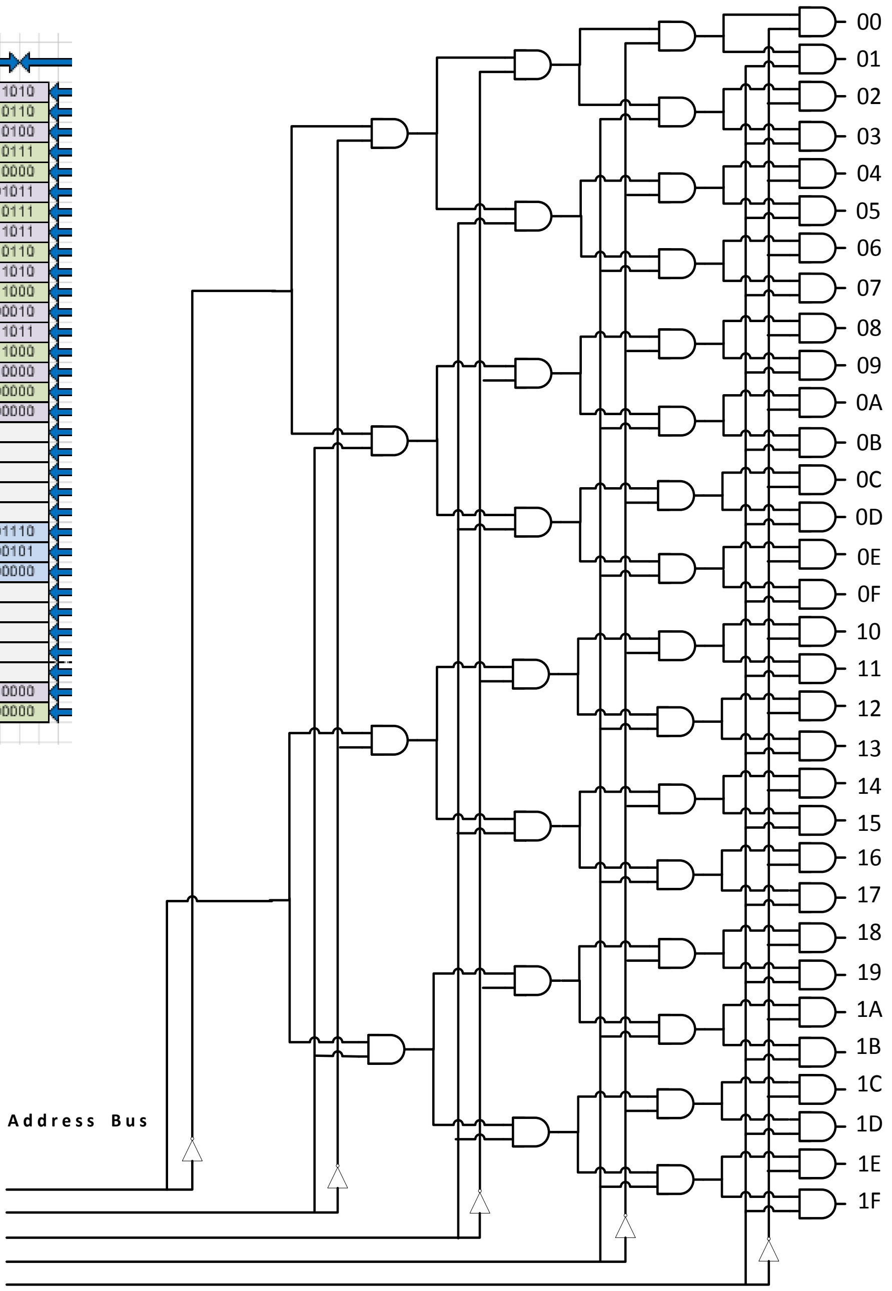
Memory Address Decoding


CLOCK
OPCODE IN
FETCH CIRCUITRY
LOAD INSTRUCTION REGISTERS
PROGRAM COUNTER
EXECUTE
CONTROL
ACCUMULATOR
PROGRAM COUNTER
OPCODE
OPERAND ADDRESS
JUMP ADDRESS
INSTRUCTION REGISTERS
ARITHMETIC & LOGIC UNIT (ALU)
OPCODE IN
INCREMENTOR
ADDRESS DECODER
EXECUTE CIRCUITRY
INSTRUCTION REGISTER CONTROL
ACCUMULATOR CONTROL
ALU CONTROL
MEM

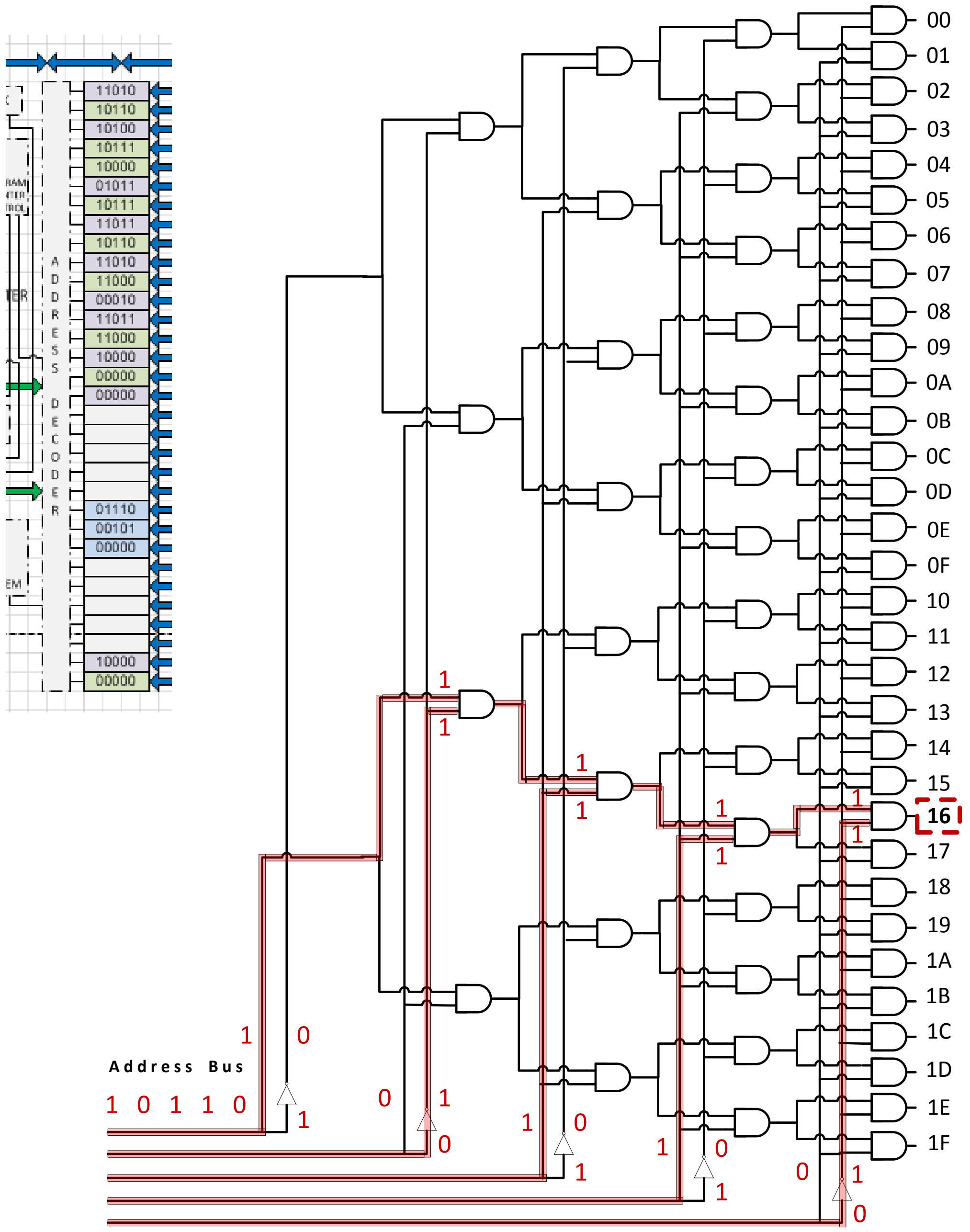
Address Decoder

Input from Address Bus

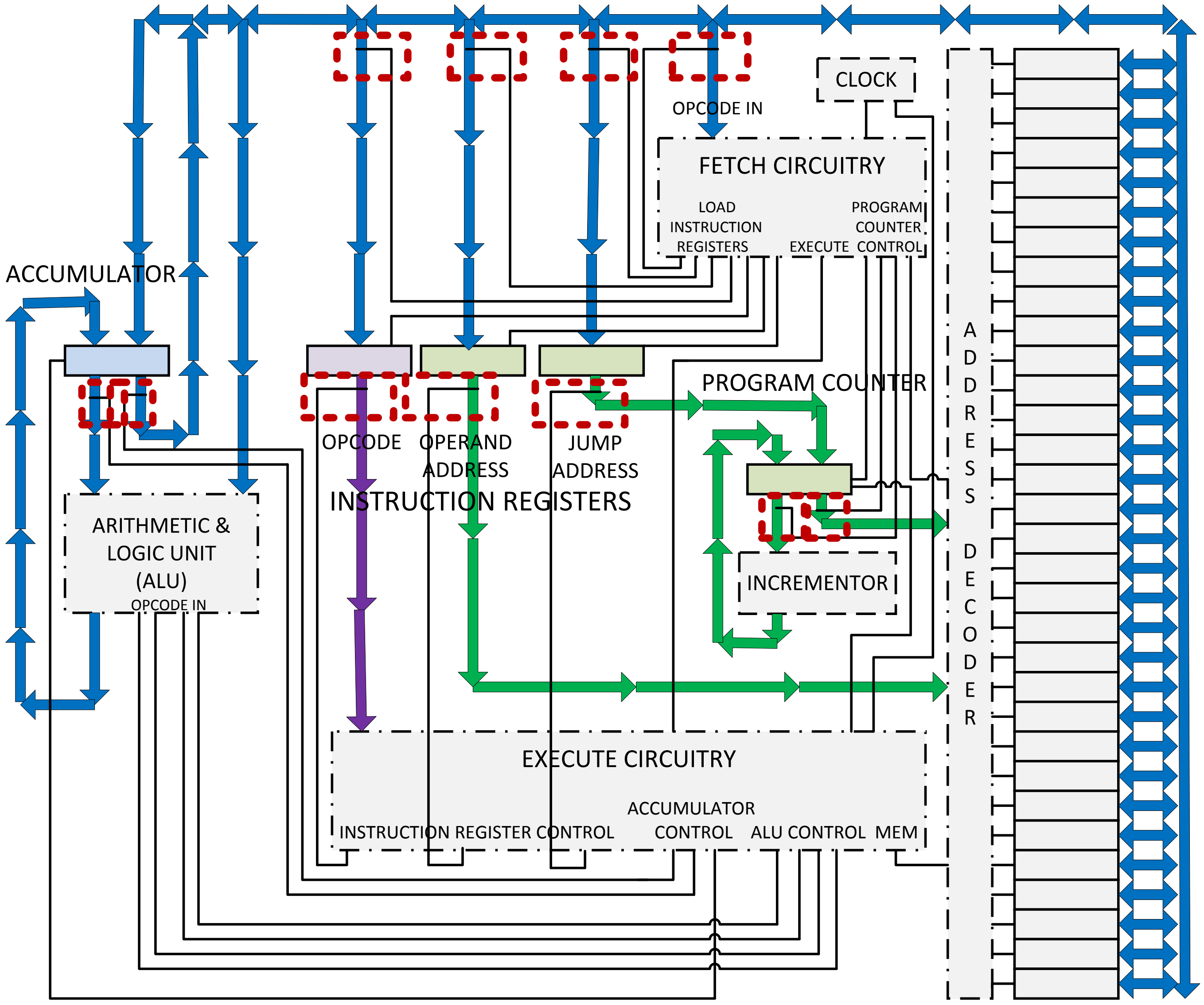




Address Decoder

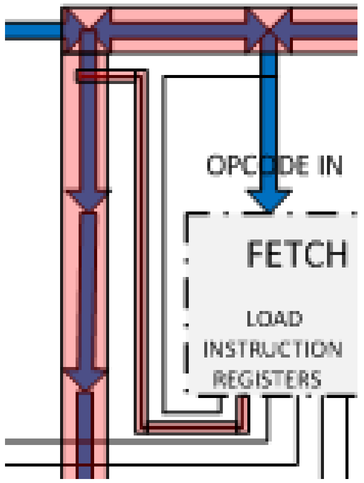


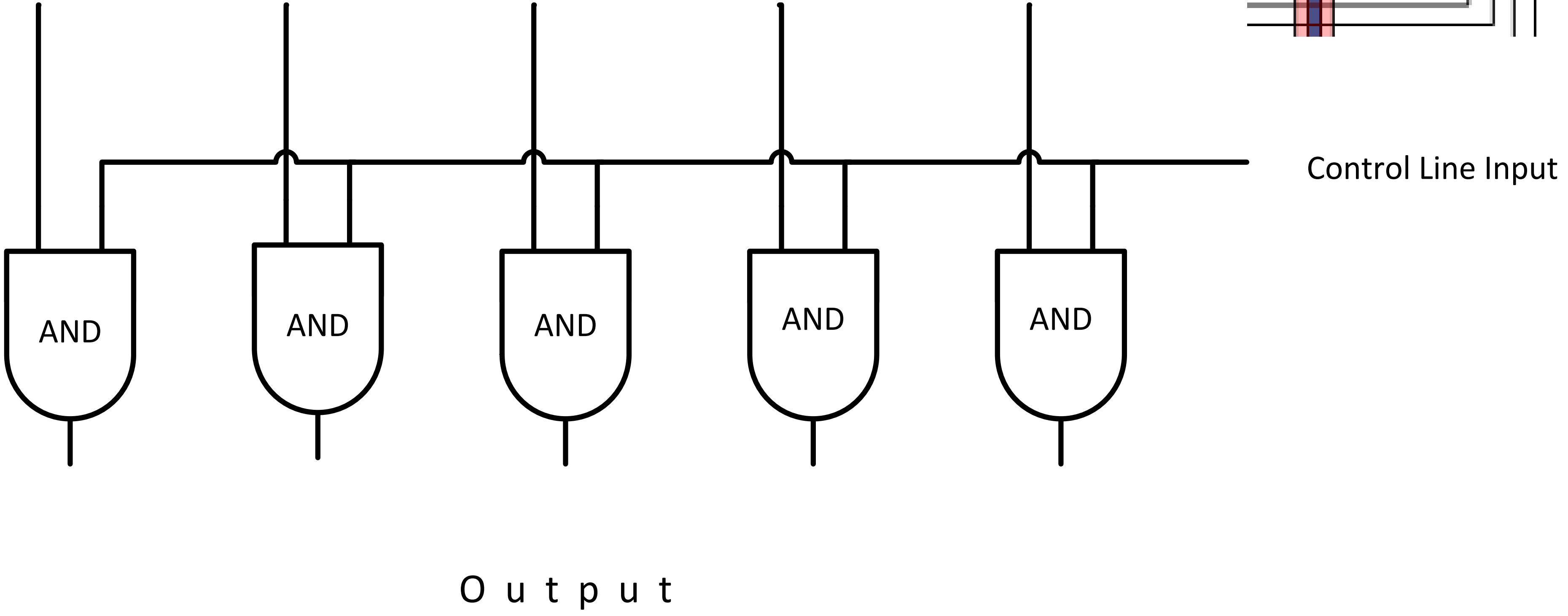
Bus Control



CLOCK
OPCODE IN
FETCH CIRCUITRY
LOAD INSTRUCTION REGISTERS
EXECUTE
PROGRAM COUNTER CONTROL
ACCUMULATOR
ADDRESS DECODER
PROGRAM COUNTER
OPCODE
OPERAND ADDRESS
JUMP ADDRESS
INSTRUCTION REGISTERS
INCREMENTOR
ARITHMETIC & LOGIC UNIT (ALU)
OPCODE IN
EXECUTE CIRCUITRY
INSTRUCTION REGISTER CONTROL
ACCUMULATOR CONTROL
ALU CONTROL
MEM

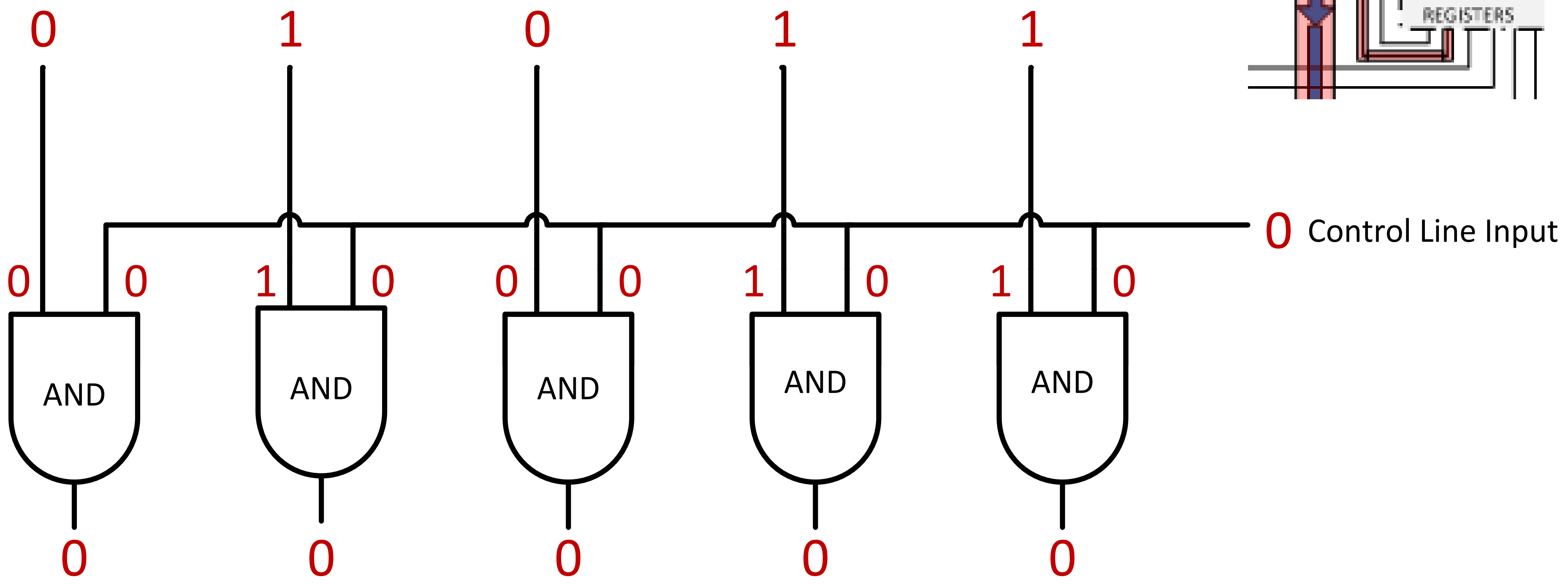
Bus Control

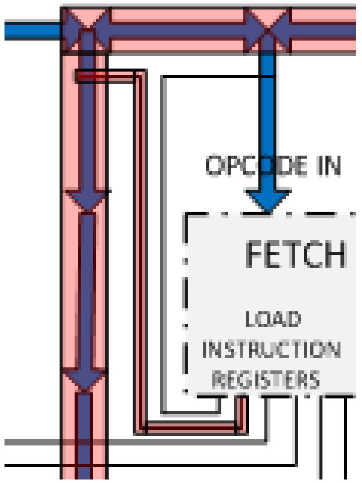




All output is 0 if Control Line is 0
Identical to Inputs From Bus if Control Line is 1

Bus Control



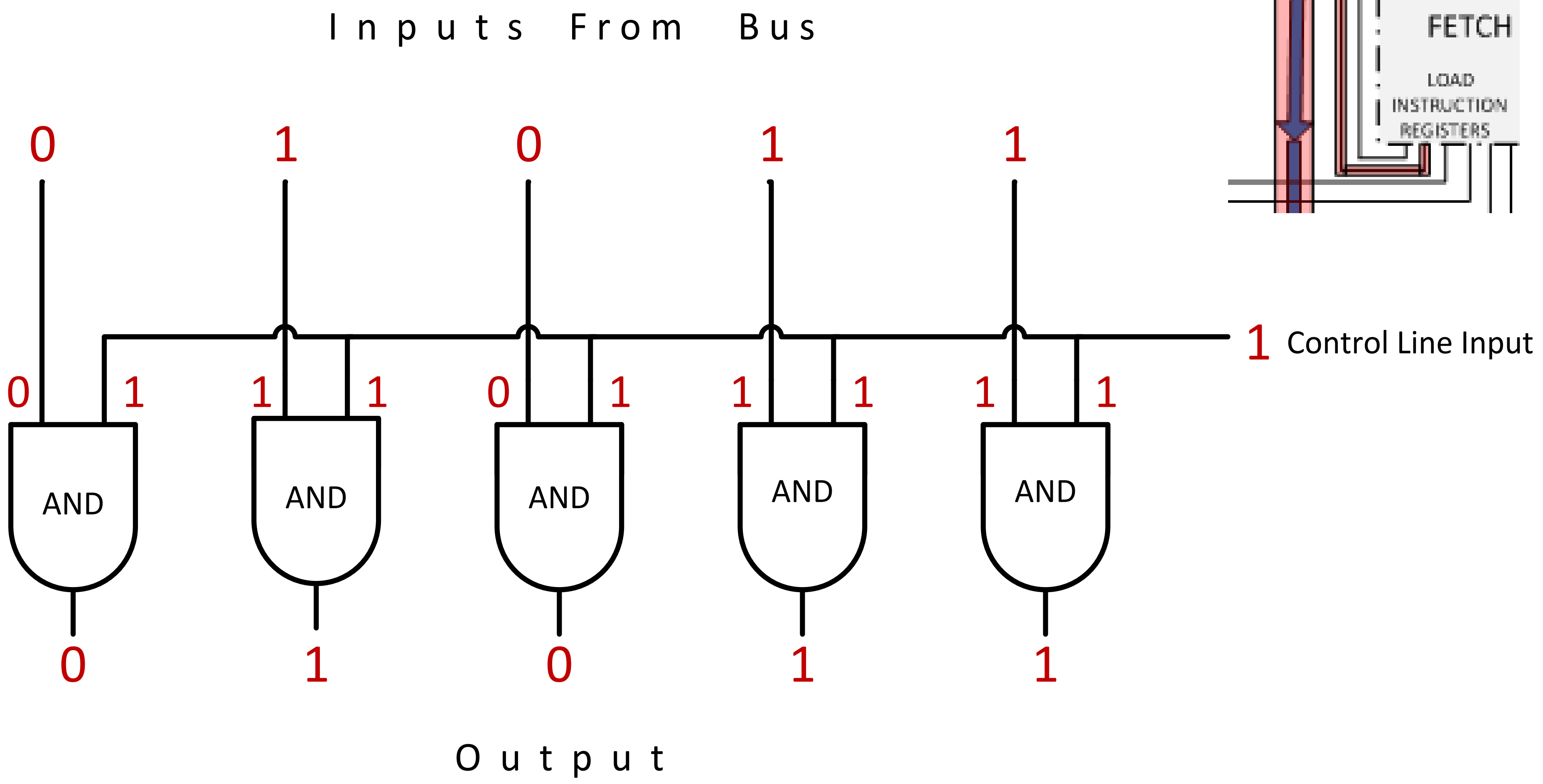


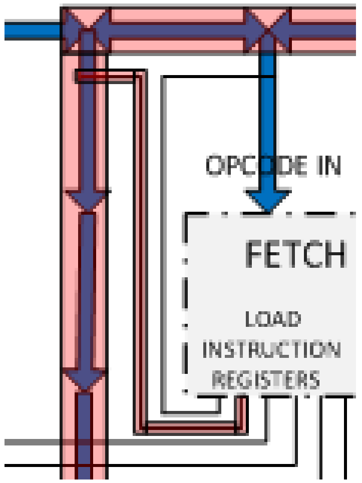
Output

All output is 0 if Control Line is 0

Identical to Inputs From Bus if Control Line is 1

Bus Control

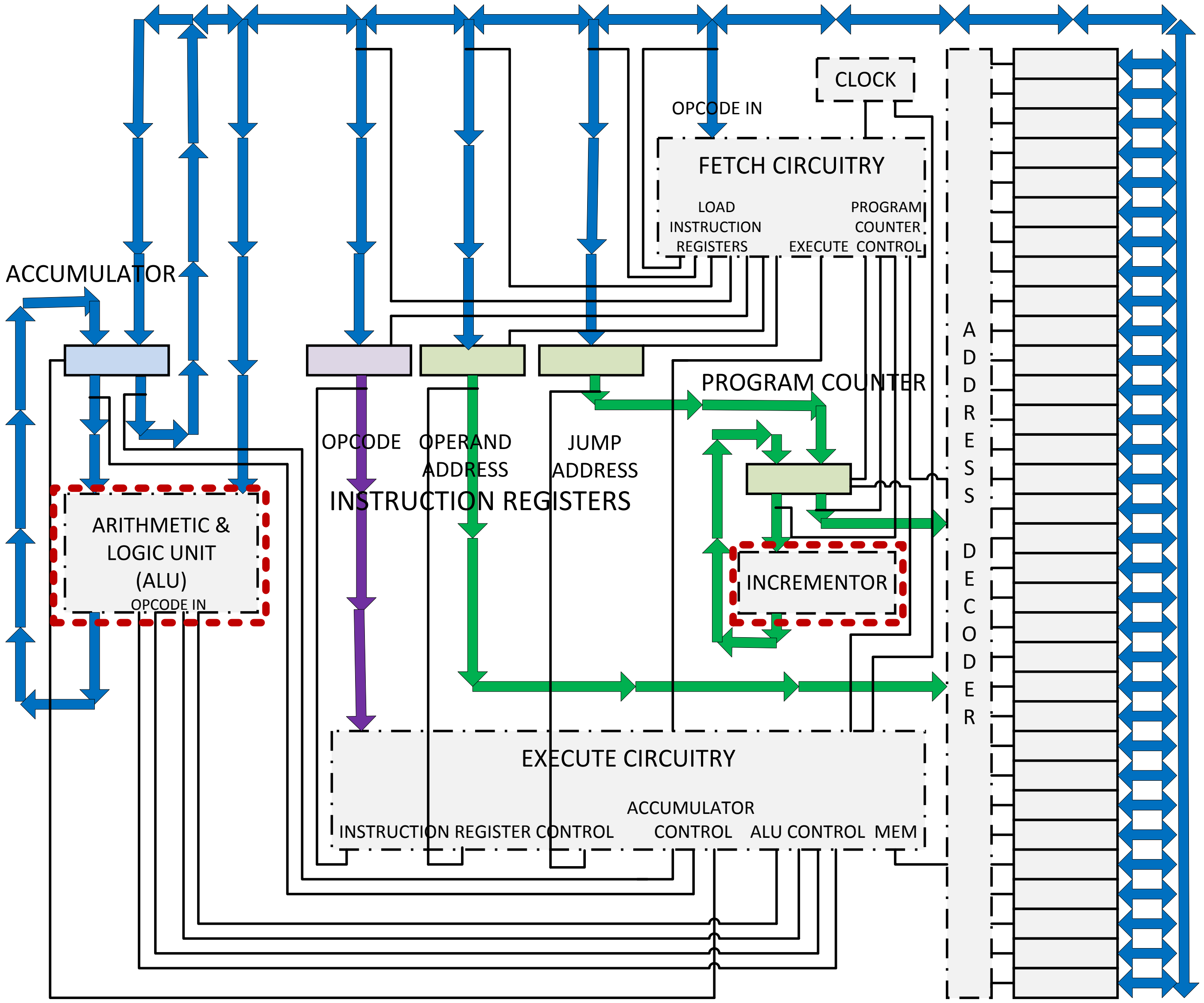




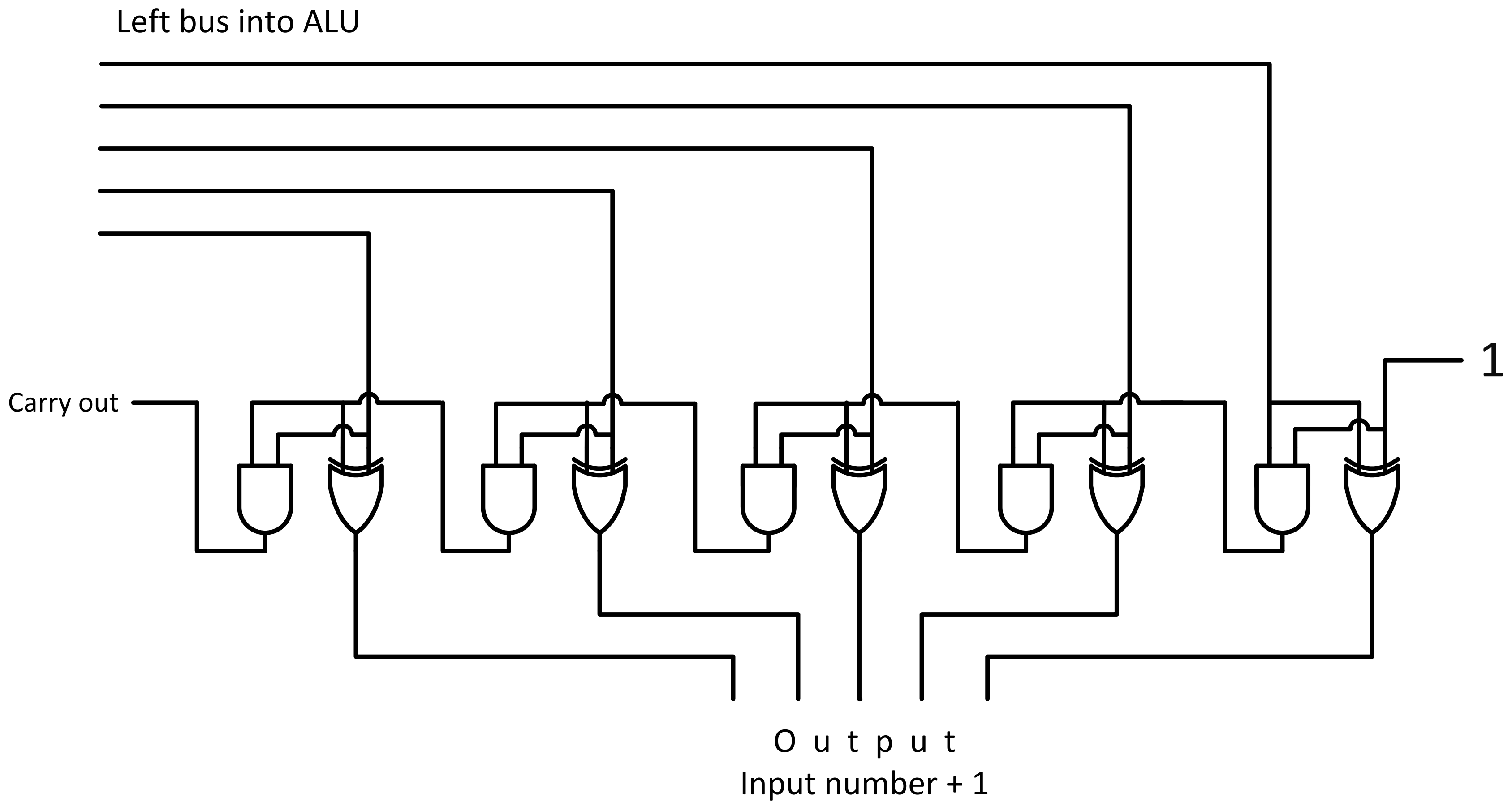
All output is 0 if Control Line is 0

Identical to Inputs From Bus if Control Line is 1

Incrementor


ACCUMULATOR
CLOCK
OPCODE IN
FETCH CIRCUITRY
LOAD INSTRUCTION REGISTERS
PROGRAM COUNTER
EXECUTE
CONTROL
PROGRAM COUNTER
OPCODE
OPERAND ADDRESS
JUMP ADDRESS
INSTRUCTION REGISTERS
ARITHMETIC & LOGIC UNIT (ALU)
OPCODE IN
INCREMENTOR
ADDRESS DECODER
EXECUTE CIRCUITRY
INSTRUCTION REGISTER CONTROL
ACCUMULATOR CONTROL
ALU CONTROL
MEM

Incrementor

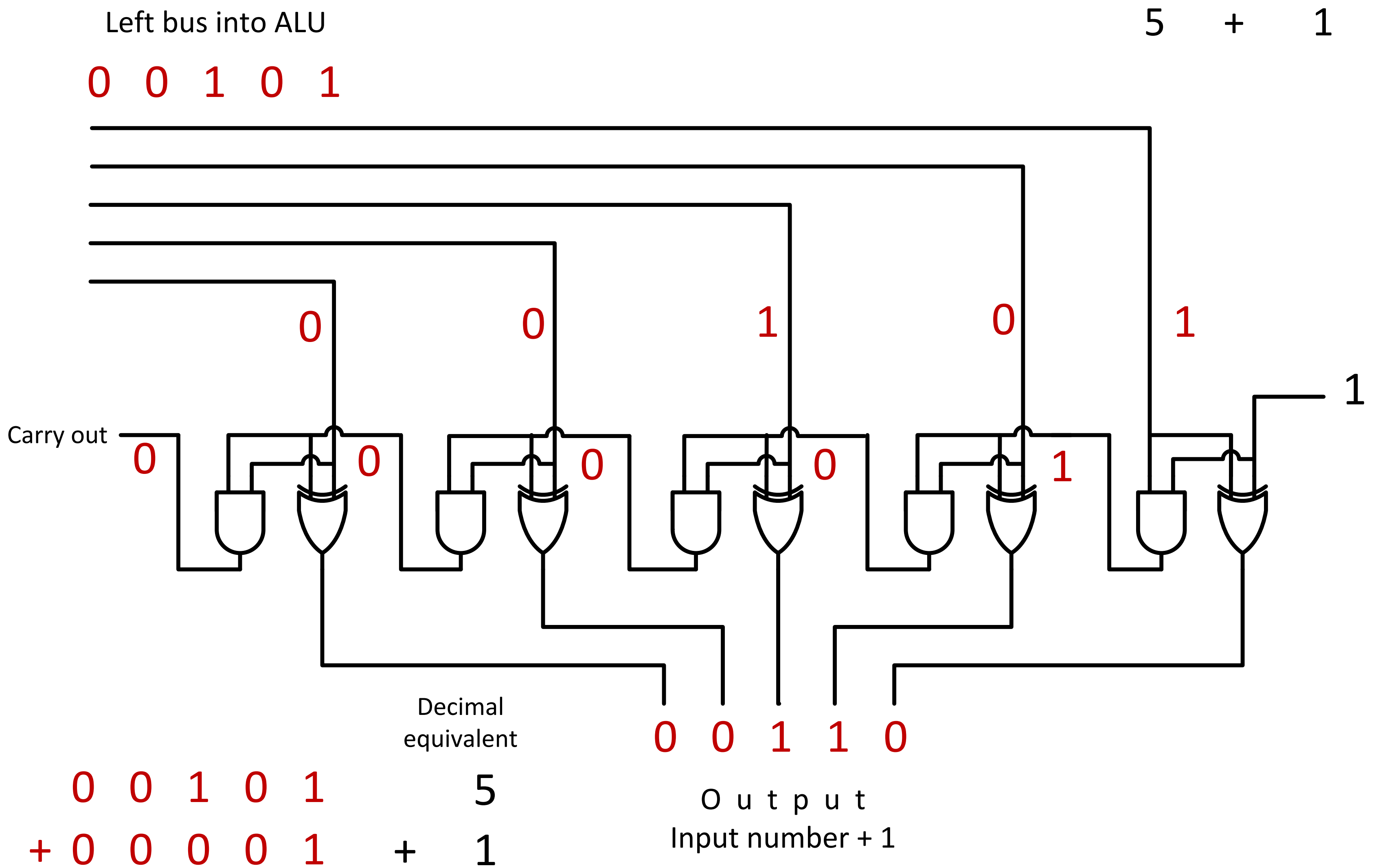


Incrementor

INC opcode

00101 + 00001

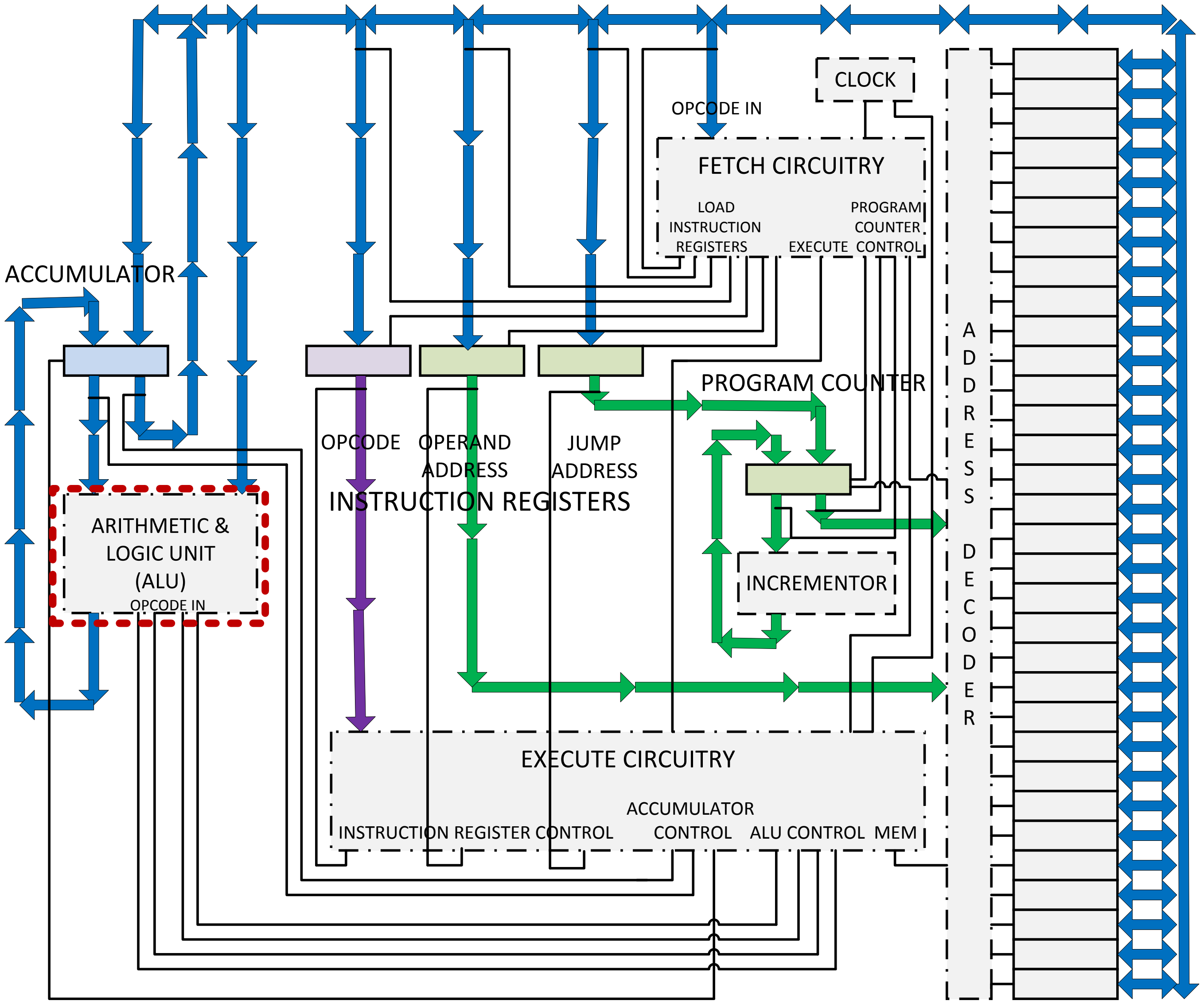
5 + 1



Decimal equivalent

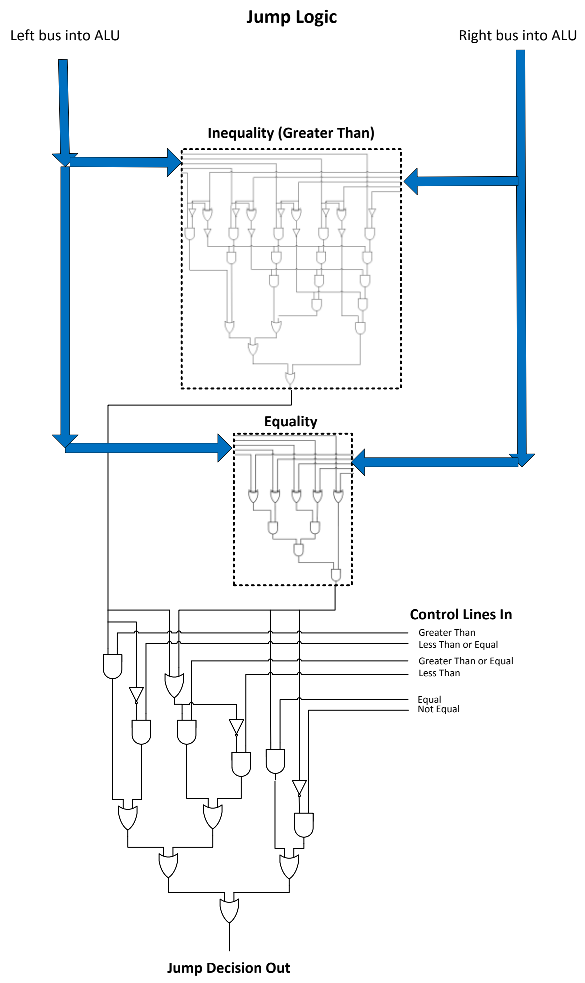
```
  0 0 1 0 1        5
+ 0 0 0 0 1     +  1
-----------     ----
  0 0 1 1 0        6
```

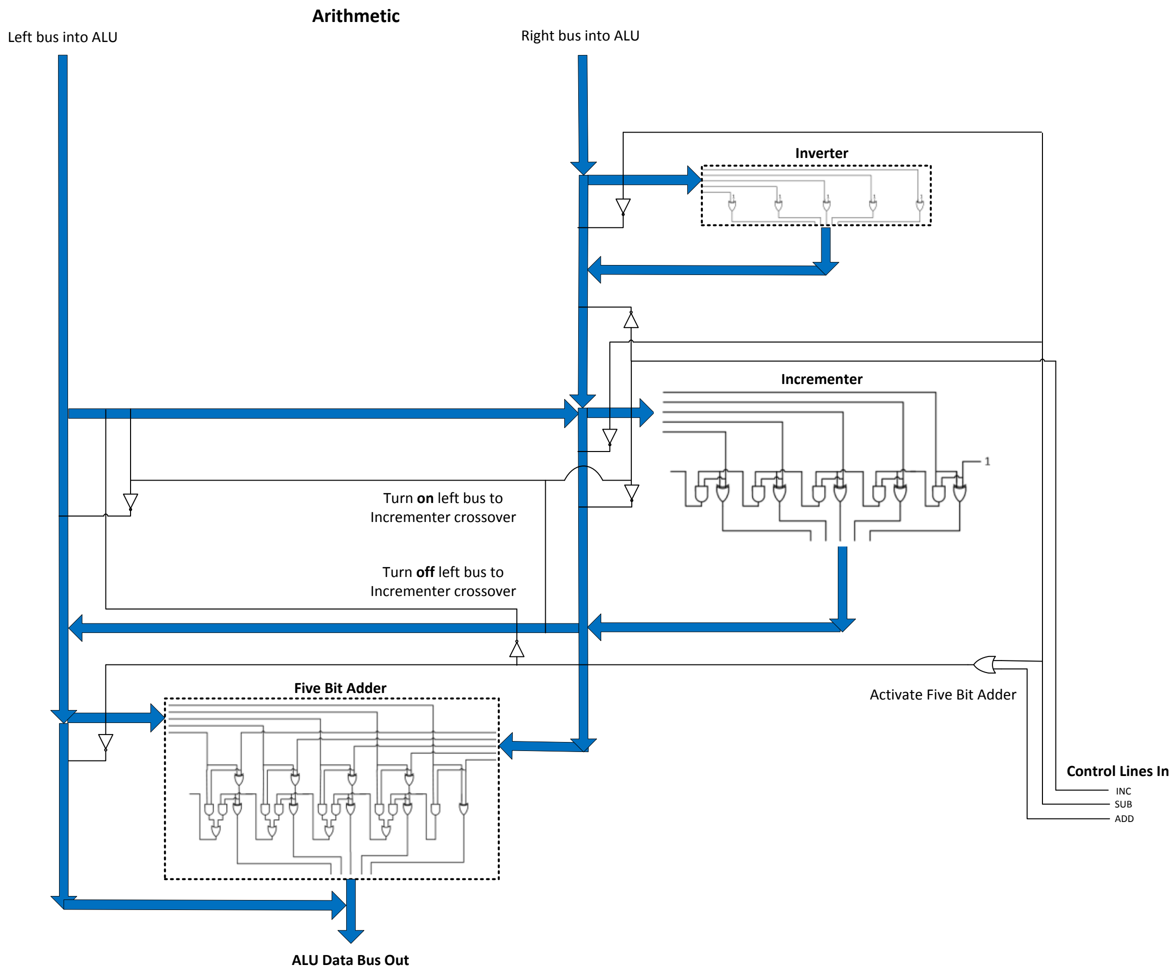
Arithmetic and Logic Unit (ALU)



CLOCK
OPCODE IN
FETCH CIRCUITRY
LOAD INSTRUCTION REGISTERS
PROGRAM COUNTER CONTROL
EXECUTE
ACCUMULATOR
PROGRAM COUNTER
OPCODE
OPERAND ADDRESS
JUMP ADDRESS
INSTRUCTION REGISTERS
ARITHMETIC & LOGIC UNIT (ALU)
OPCODE IN
INCREMENTOR
ADDRESS DECODER
EXECUTE CIRCUITRY
INSTRUCTION REGISTER CONTROL
ACCUMULATOR CONTROL
ALU CONTROL
MEM

A L U
(Arithmetic and Logic Unit)

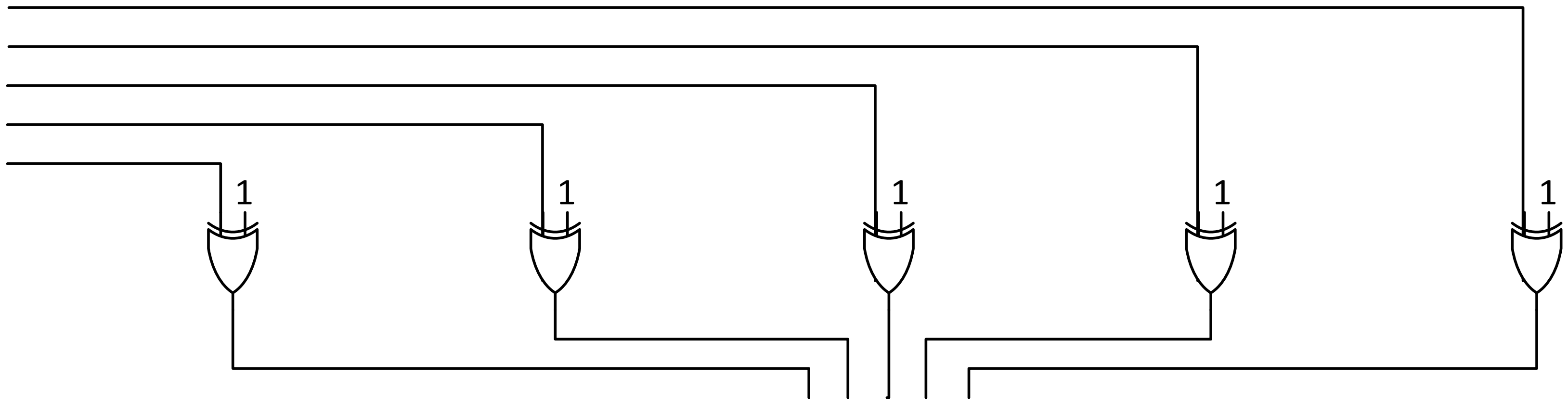




ALU - Invertor

Invertor

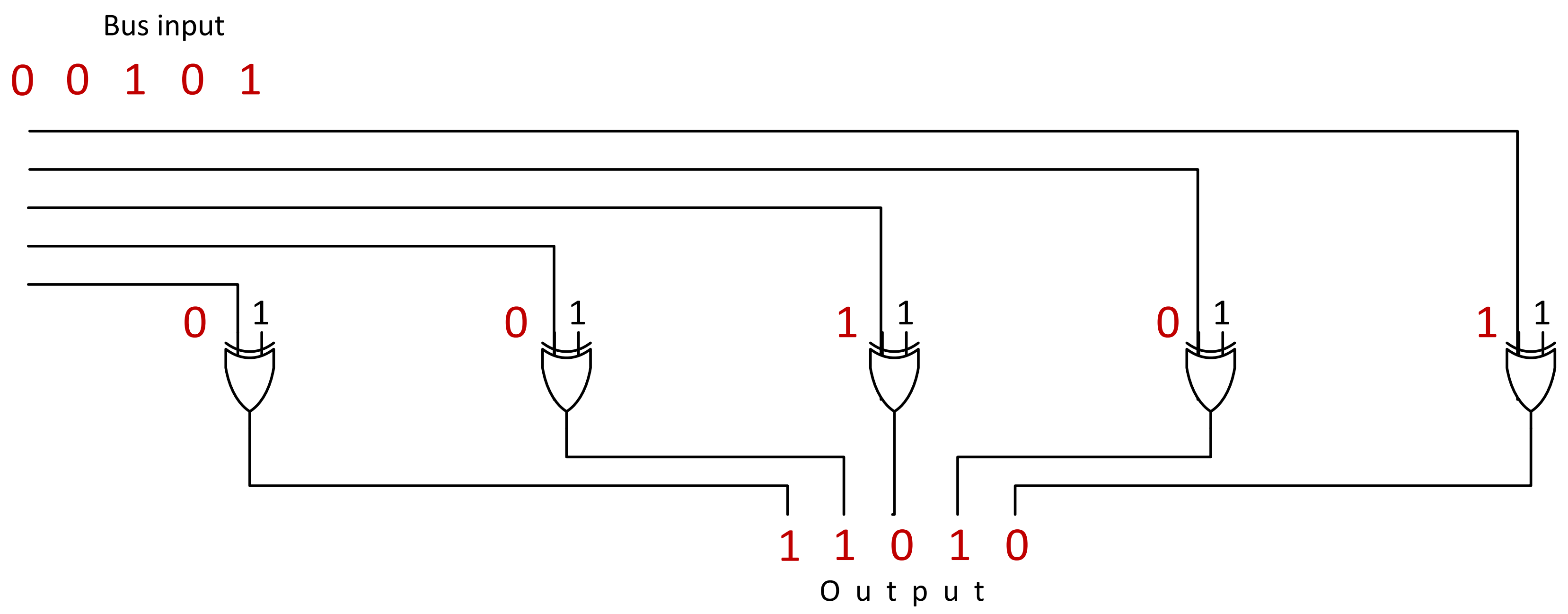
Bus input



O u t p u t

twos complement of the input bus

Invertor

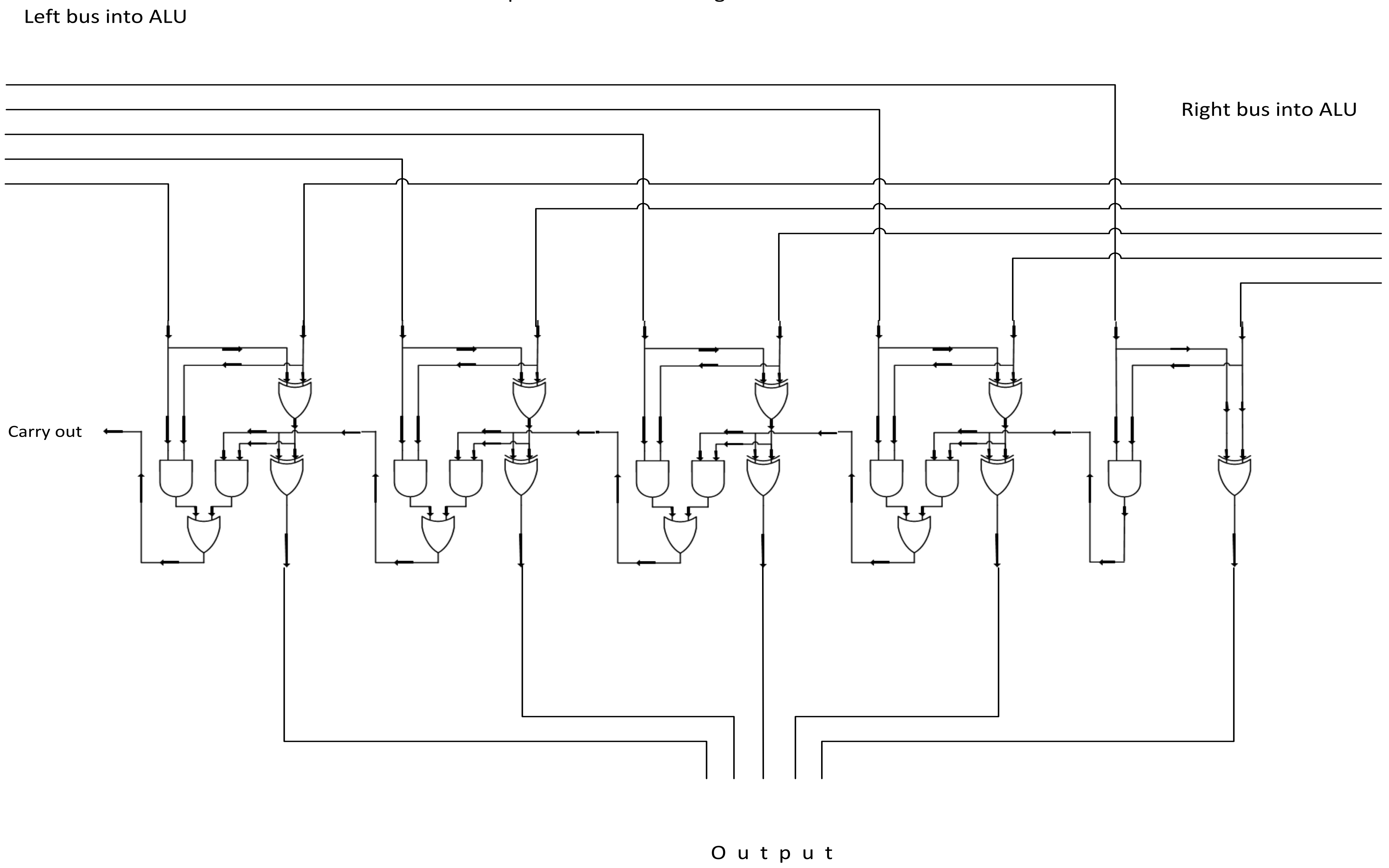


twos complement of the input bus

ALU - Five Bit Adder

Five Bit Adder

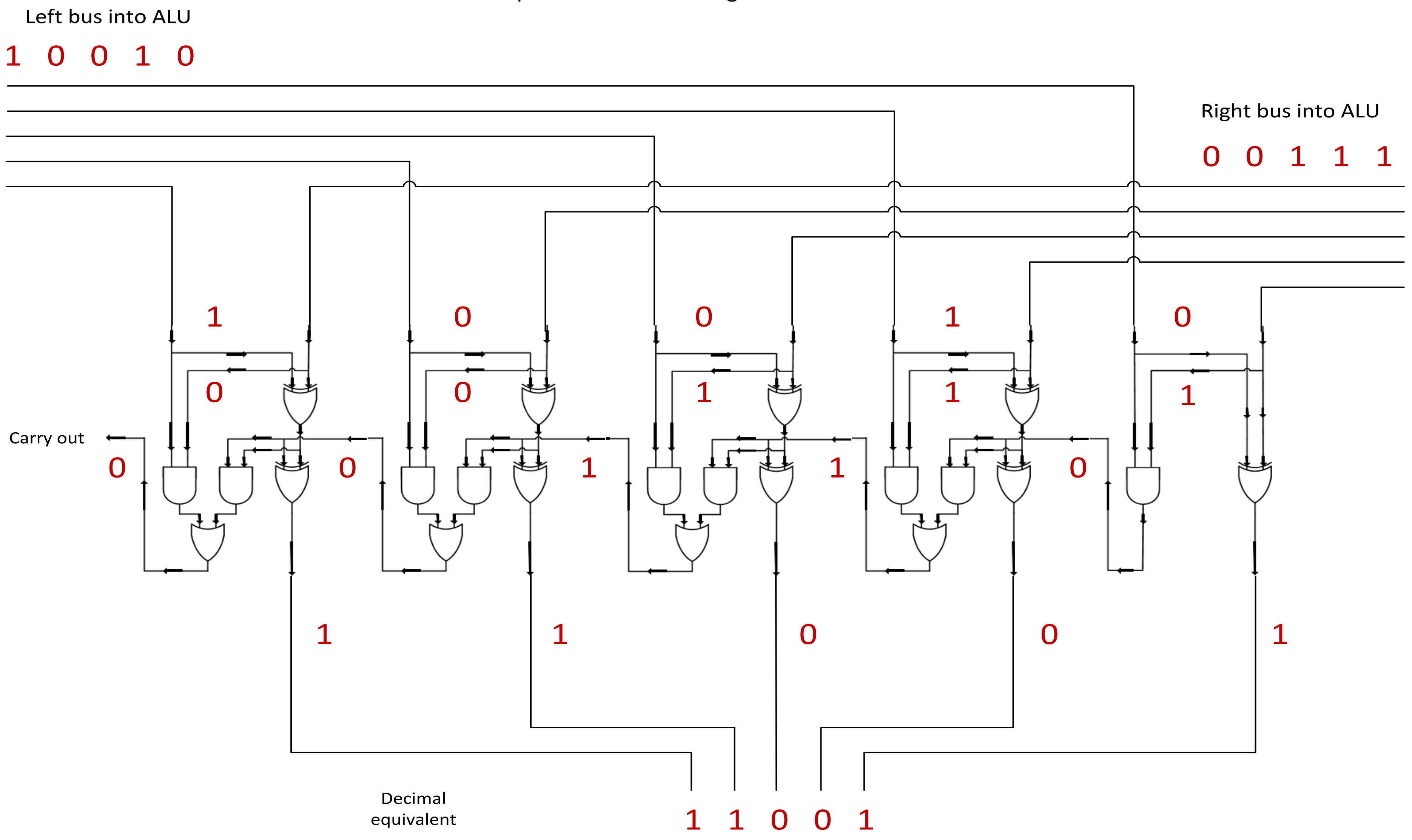
Inputs from Left and Right Busses into the ALU



Accurate addition of two 5-bit binary numbers.

Five Bit Adder

Inputs from Left and Right Busses into the ALU



Output

	1	0	0	1	0
+	0	0	1	1	1
	1	1	0	0	1

Decimal equivalent

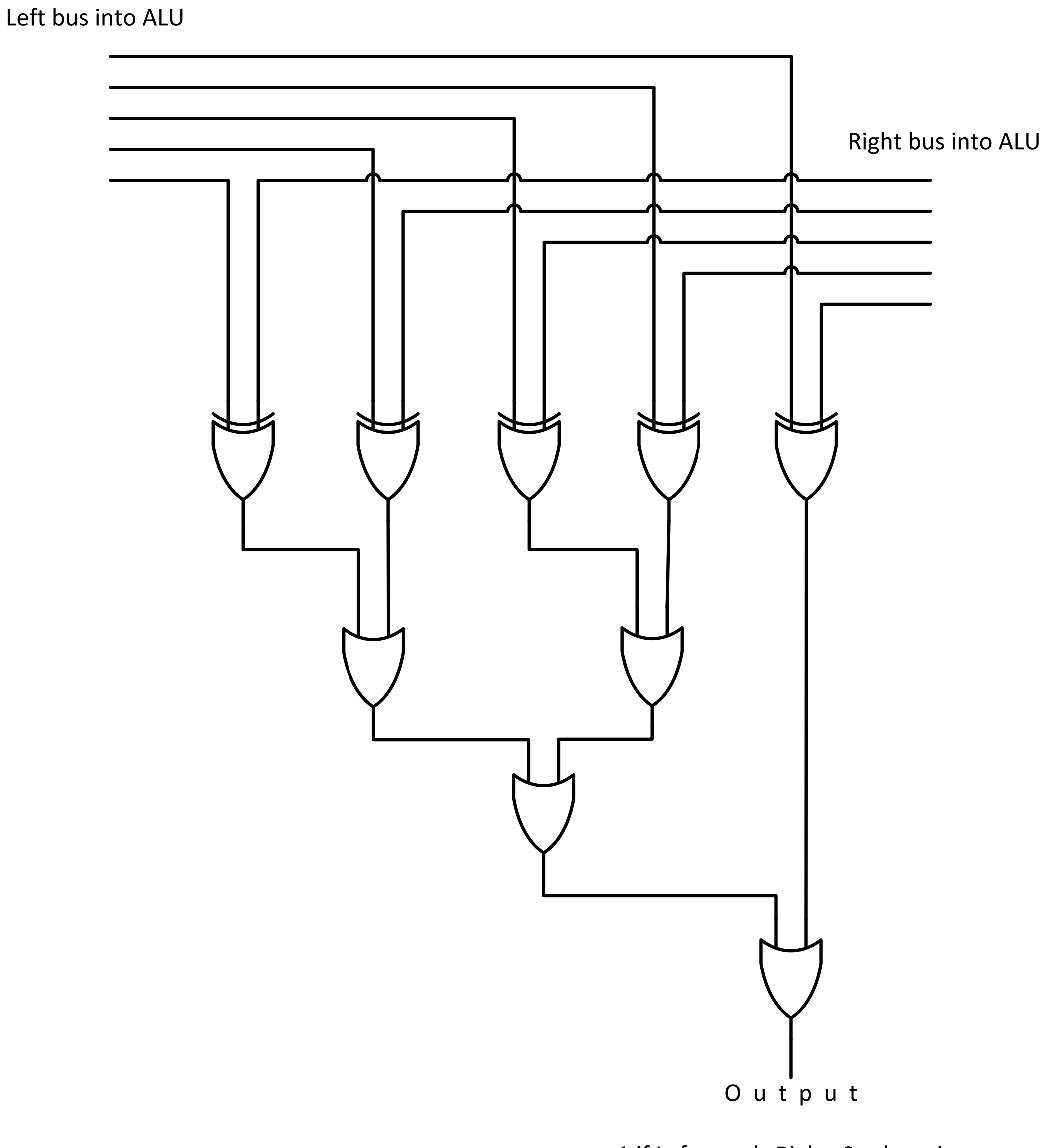
$$\begin{array}{r} 18 \\ +\ 7 \\ \hline 25 \end{array}$$

Accurate addition of two 5-bit binary numbers.

ALU – Equality

Equality

Inputs from Left and Right Busses into the ALU

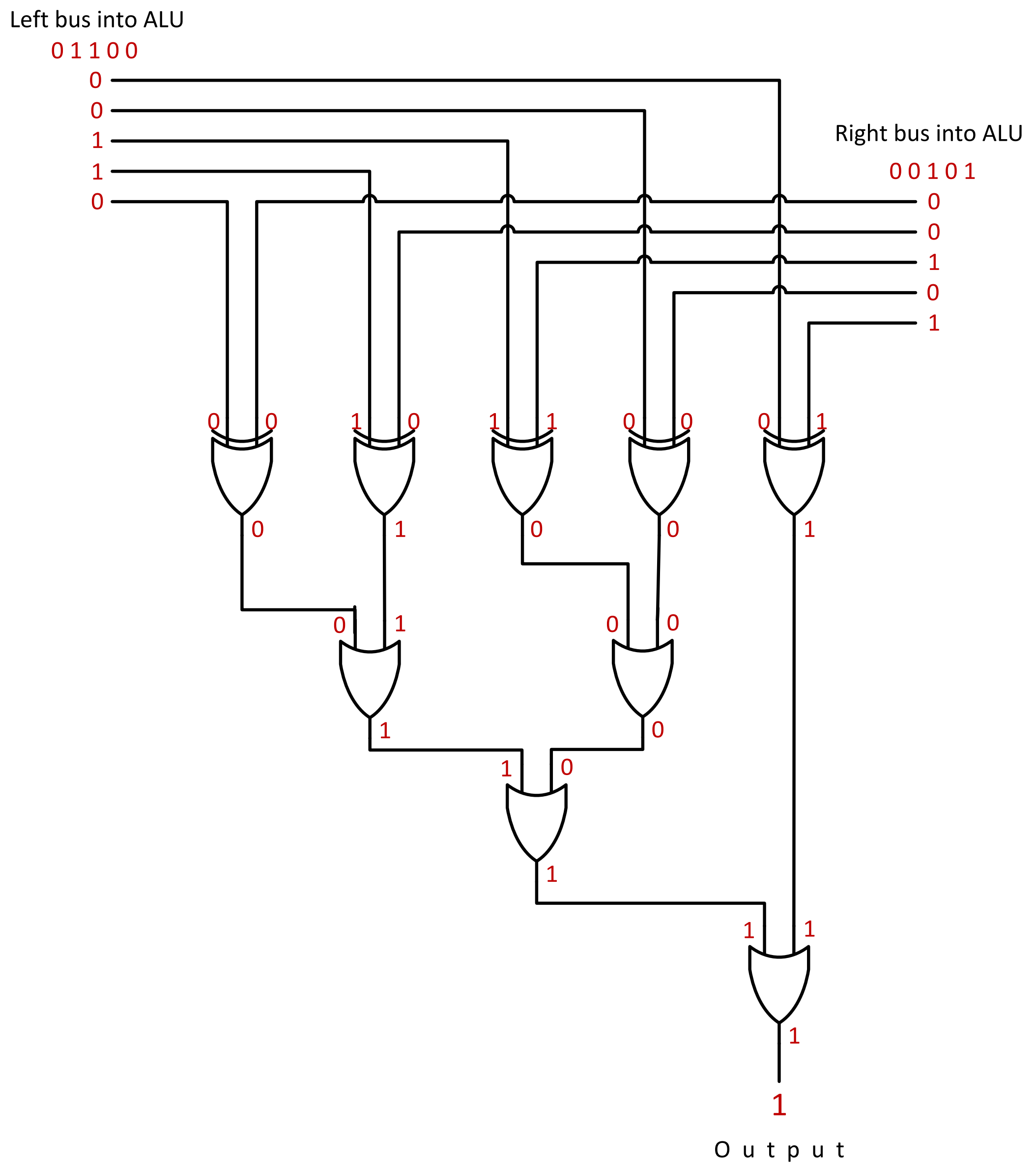


1 if Left equals Right, 0 otherwise

Equality

01100 = 00101
12 = 5

Inputs from Left and Right Busses into the ALU

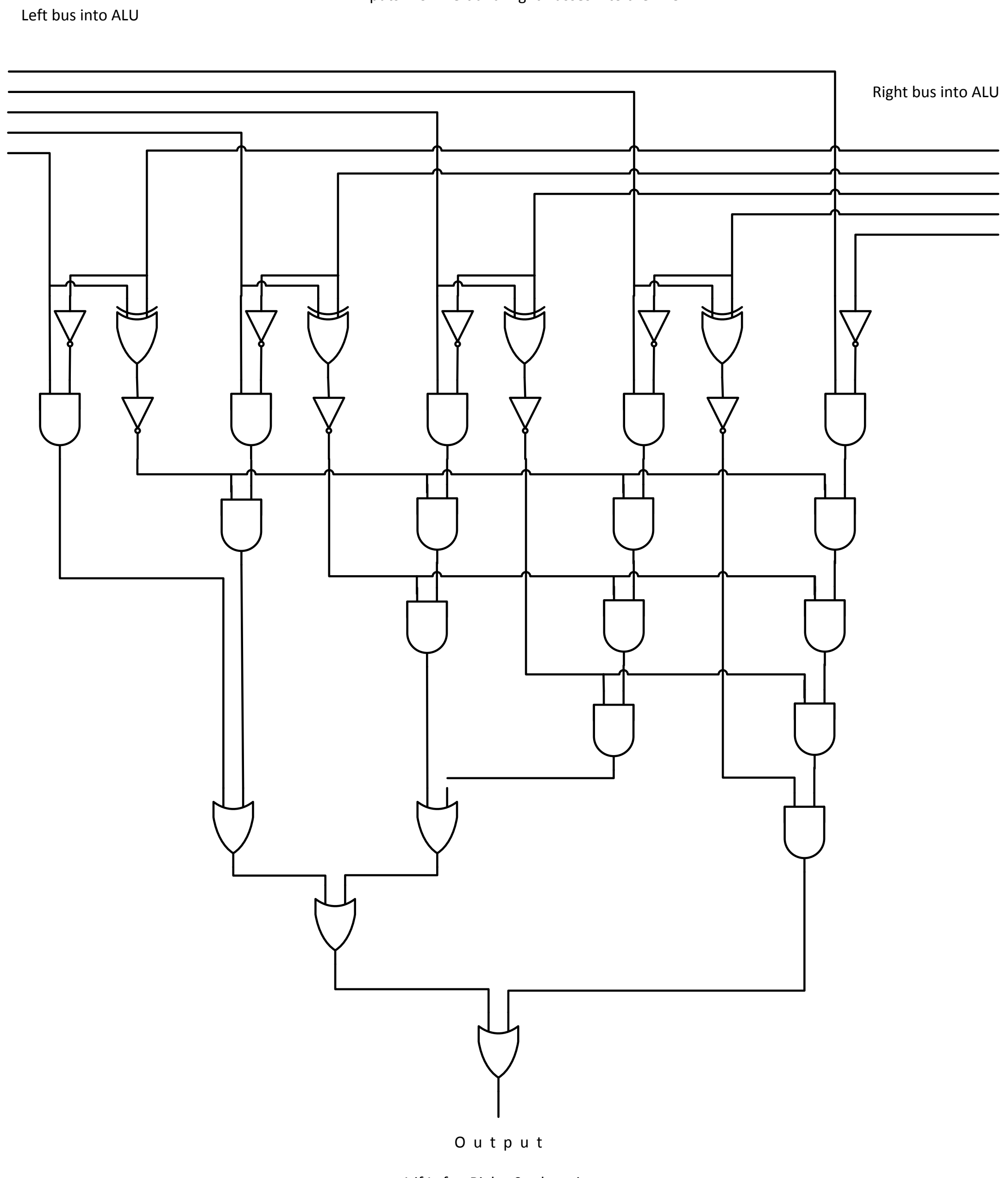


1 if Left equals Right, 0 otherwise

ALU – Greater Than

Inequality – Left > Right

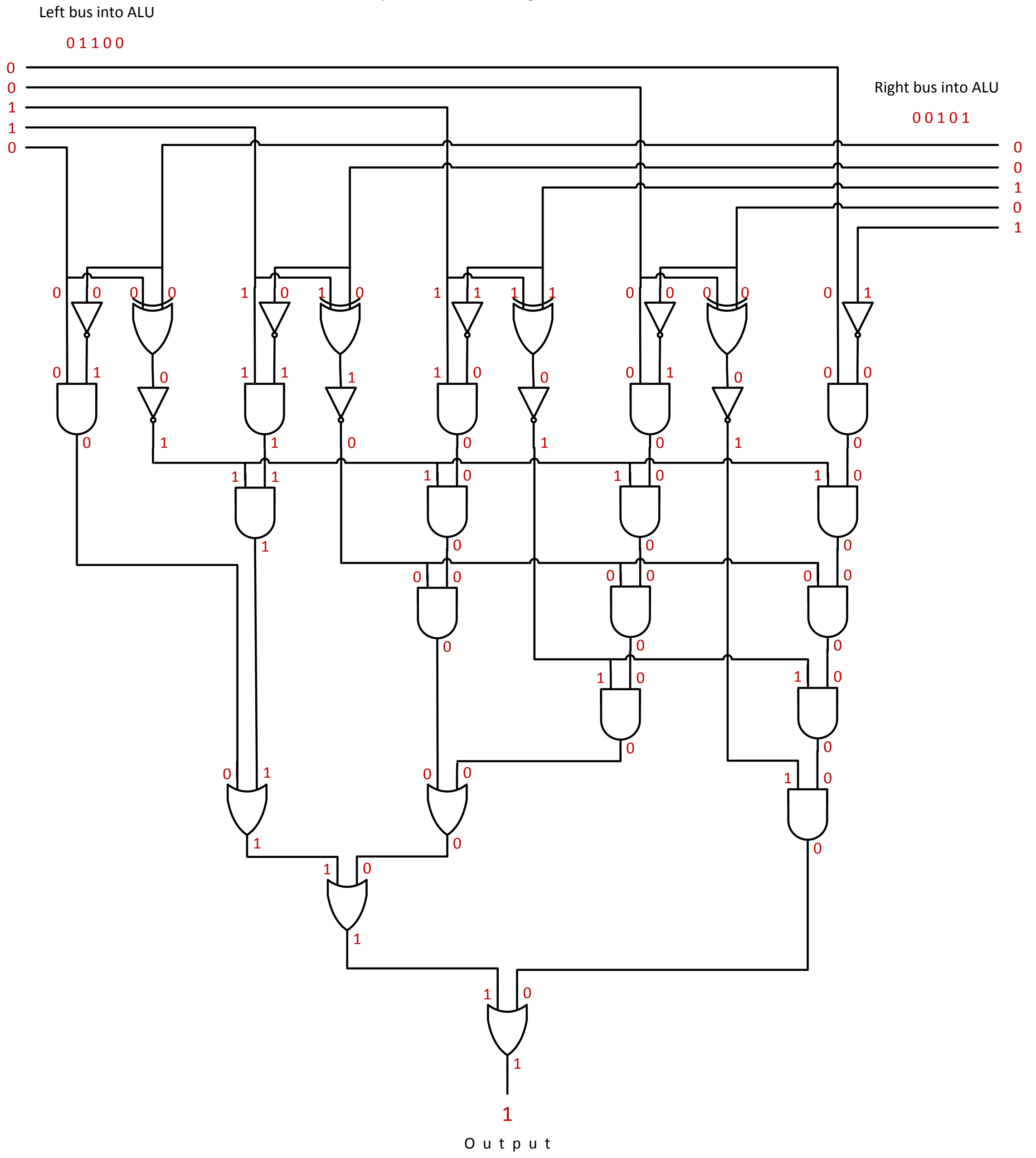
Inputs from Left and Right Busses into the ALU



Inequality – Left > Right

01100 > 00101
12 > 5

Inputs from Left and Right Busses into the ALU

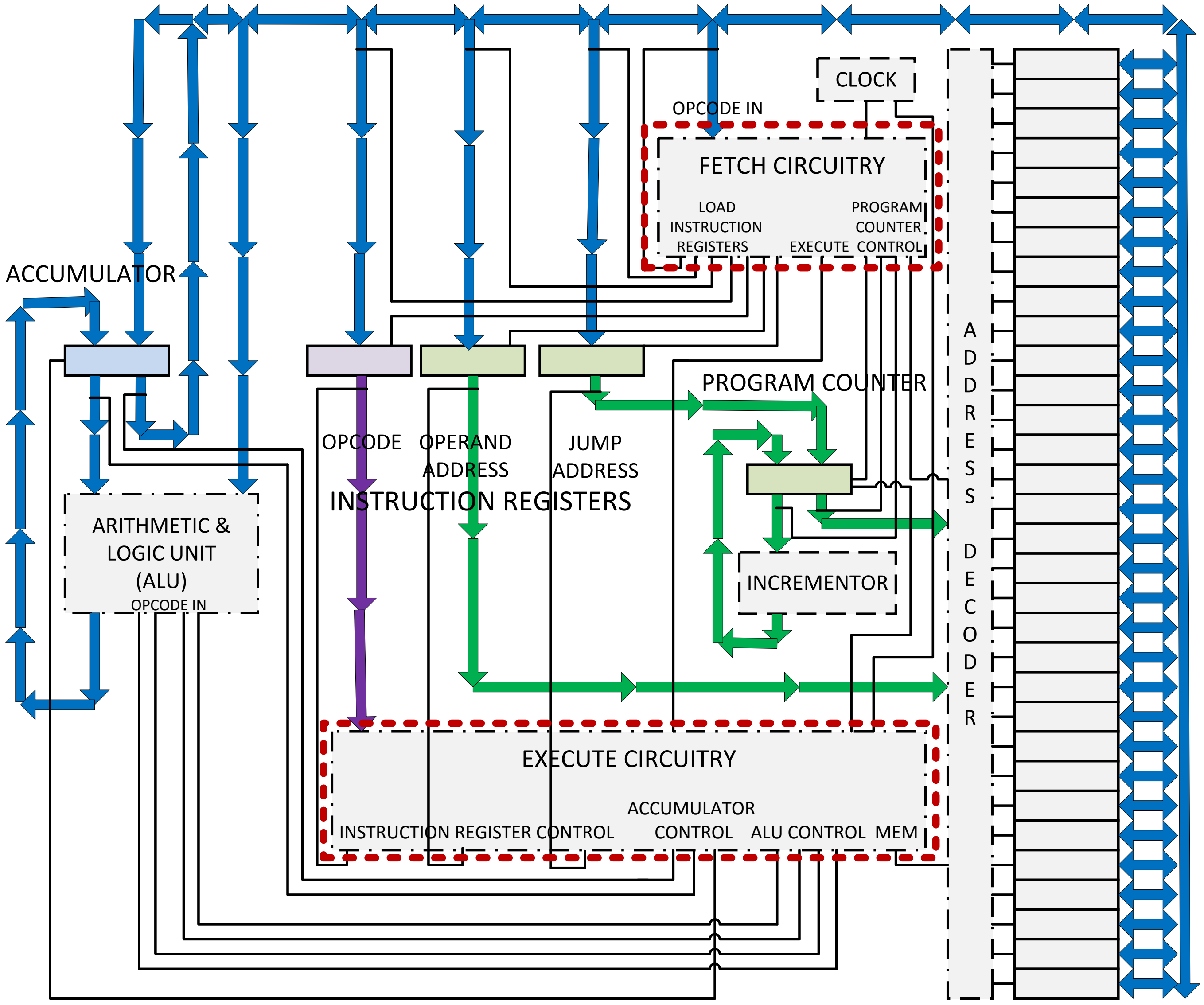


1 if Left > Right, 0 otherwise

End of ALU

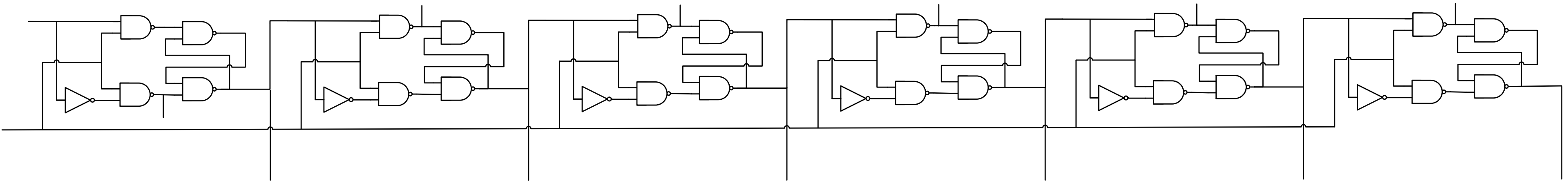
Control Circuitry

Ring Counter


CLOCK
OPCODE IN
FETCH CIRCUITRY
LOAD INSTRUCTION REGISTERS
PROGRAM COUNTER
EXECUTE CONTROL
ACCUMULATOR
PROGRAM COUNTER
OPCODE
OPERAND ADDRESS
JUMP ADDRESS
INSTRUCTION REGISTERS
ARITHMETIC & LOGIC UNIT (ALU)
OPCODE IN
INCREMENTOR
ADDRESS DECODER
EXECUTE CIRCUITRY
INSTRUCTION REGISTER CONTROL
ACCUMULATOR CONTROL
ALU CONTROL
MEM

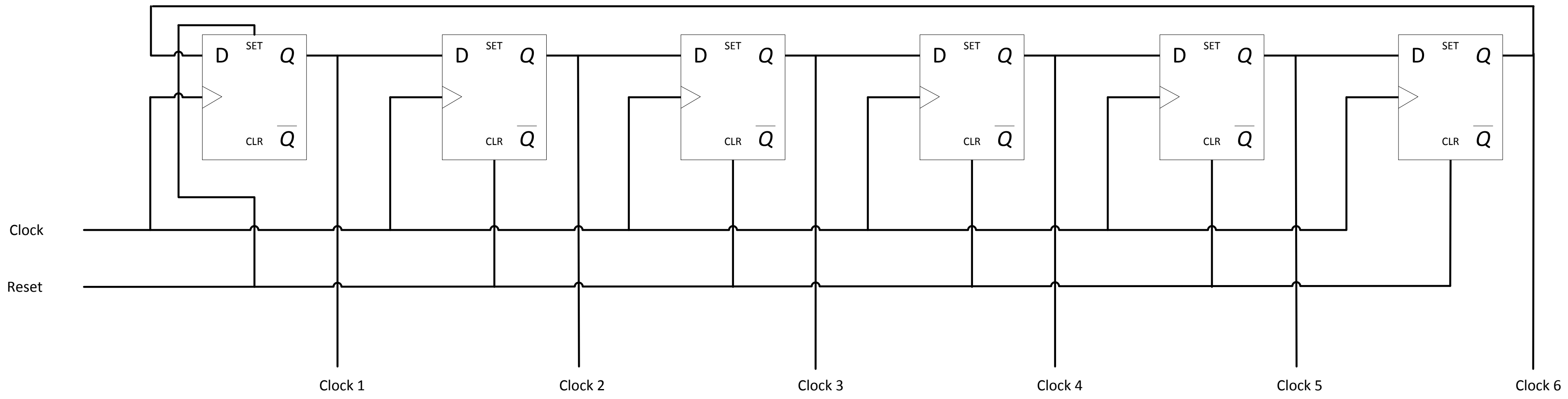
Ring Counter

based on a “D Flip-Flop” --- other designs are more efficient



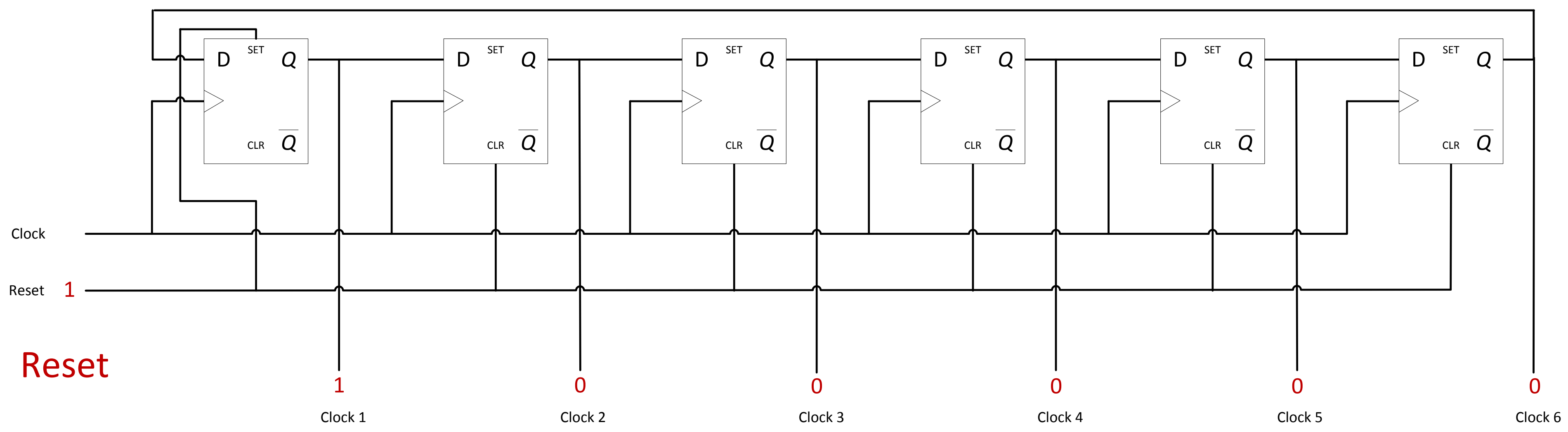
Ring Counter

based on a D Flip-Flops



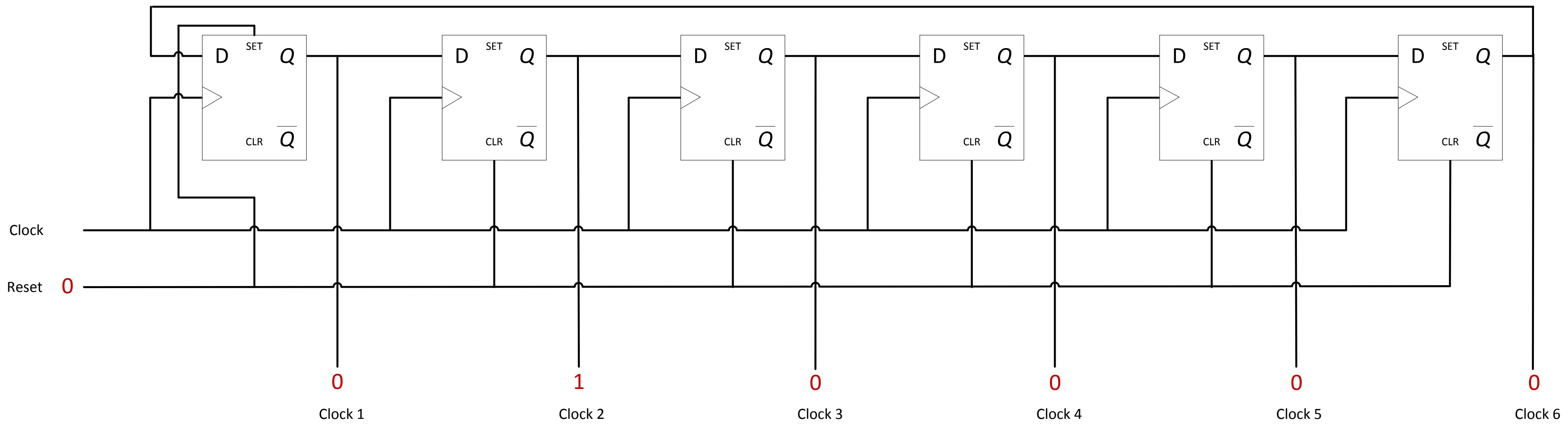
Ring Counter

based on a D Flip-Flops



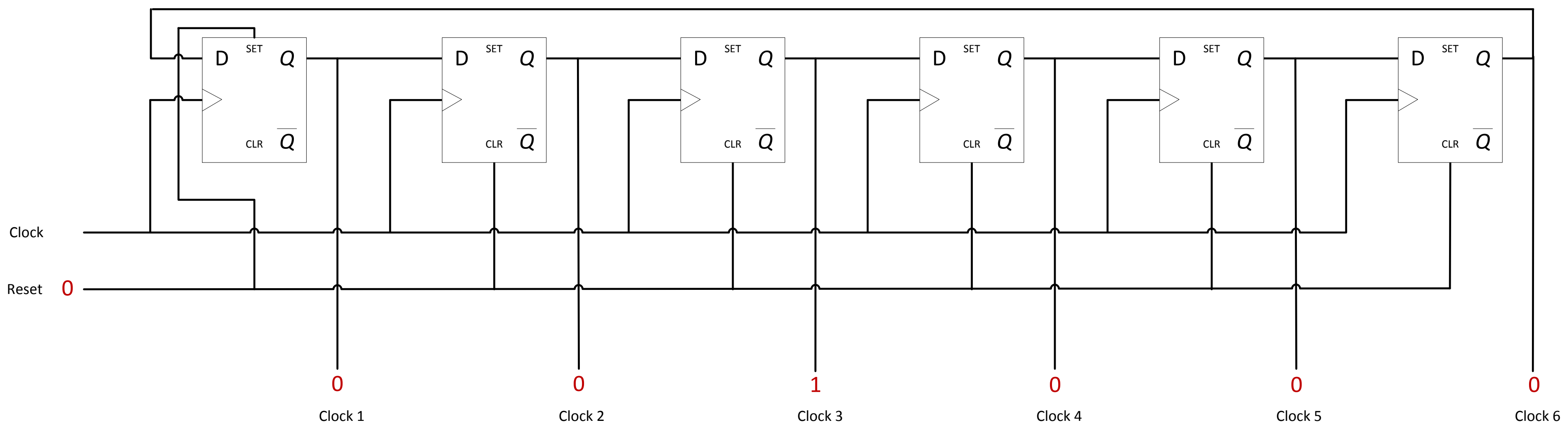
Ring Counter

based on a D Flip-Flops



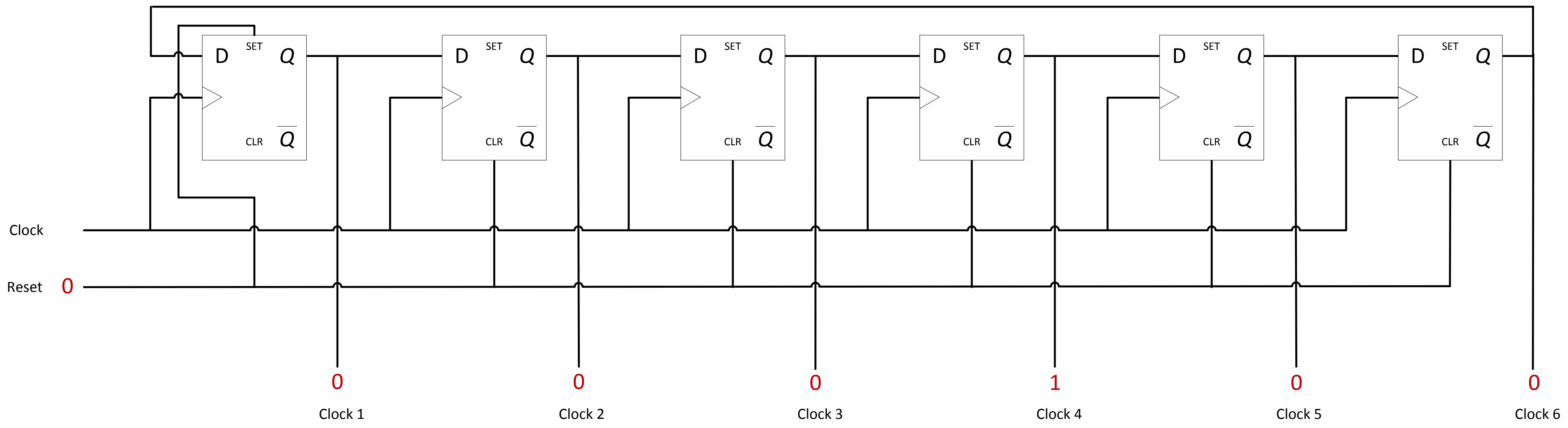
Ring Counter

based on a D Flip-Flops



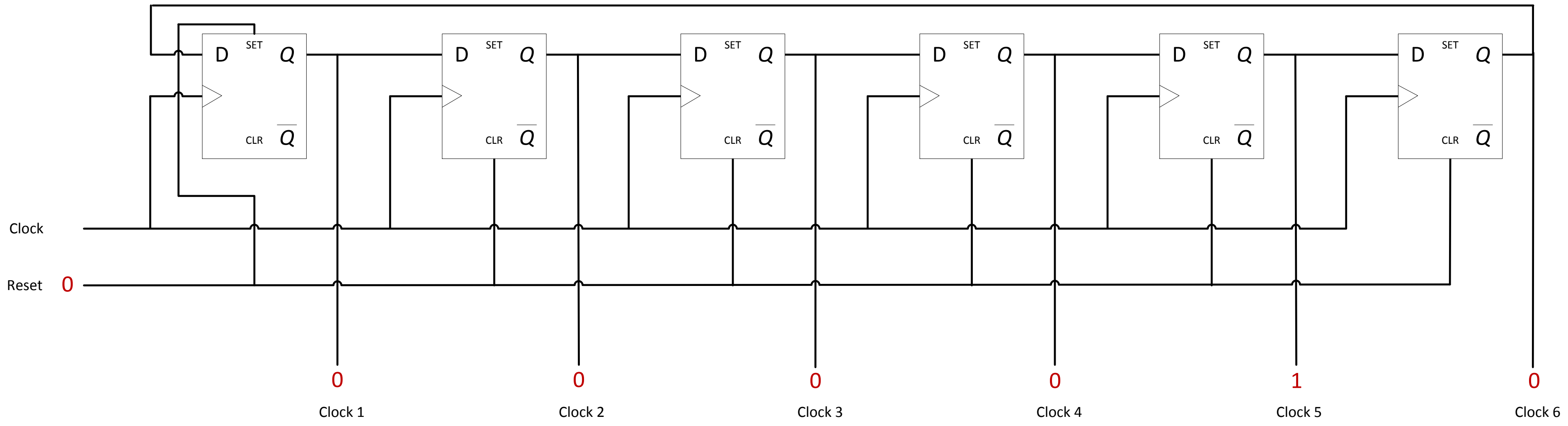
Ring Counter

based on a D Flip-Flops



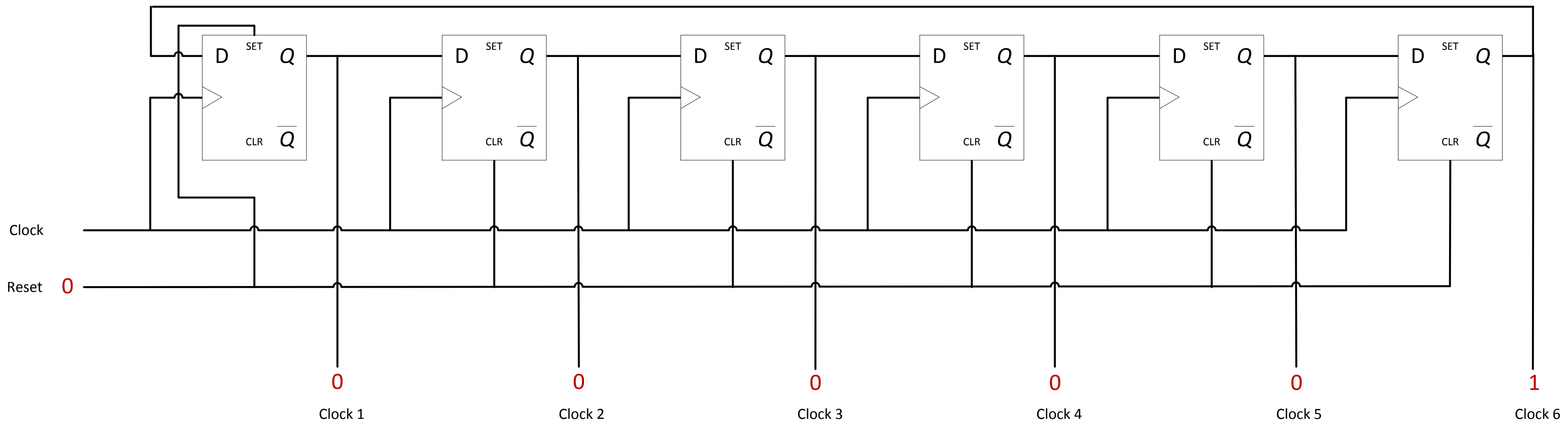
Ring Counter

based on a D Flip-Flops



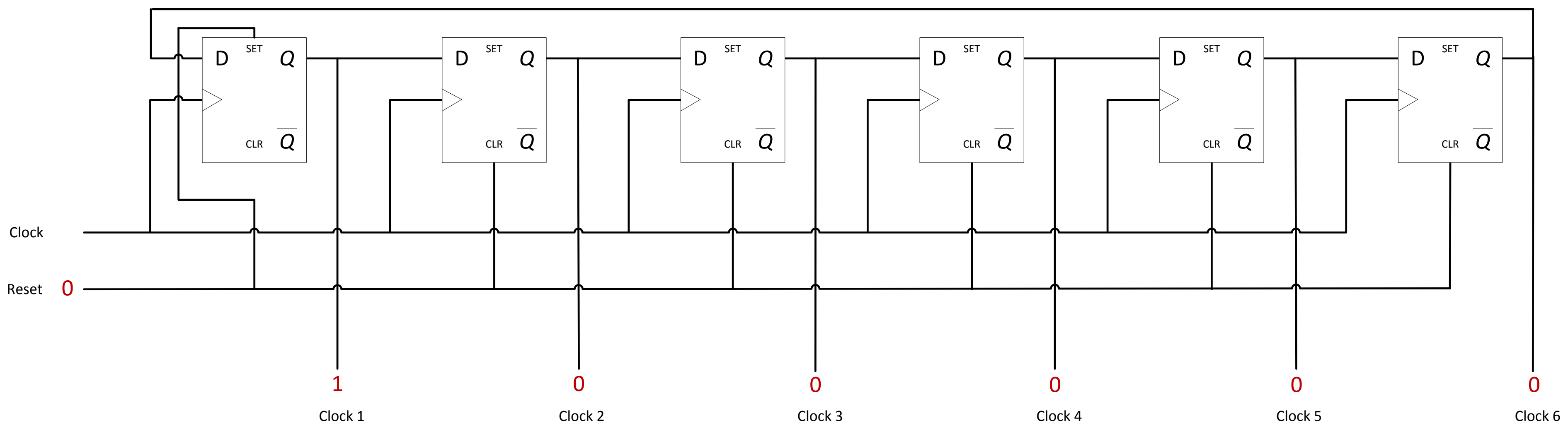
Ring Counter

based on a D Flip-Flops



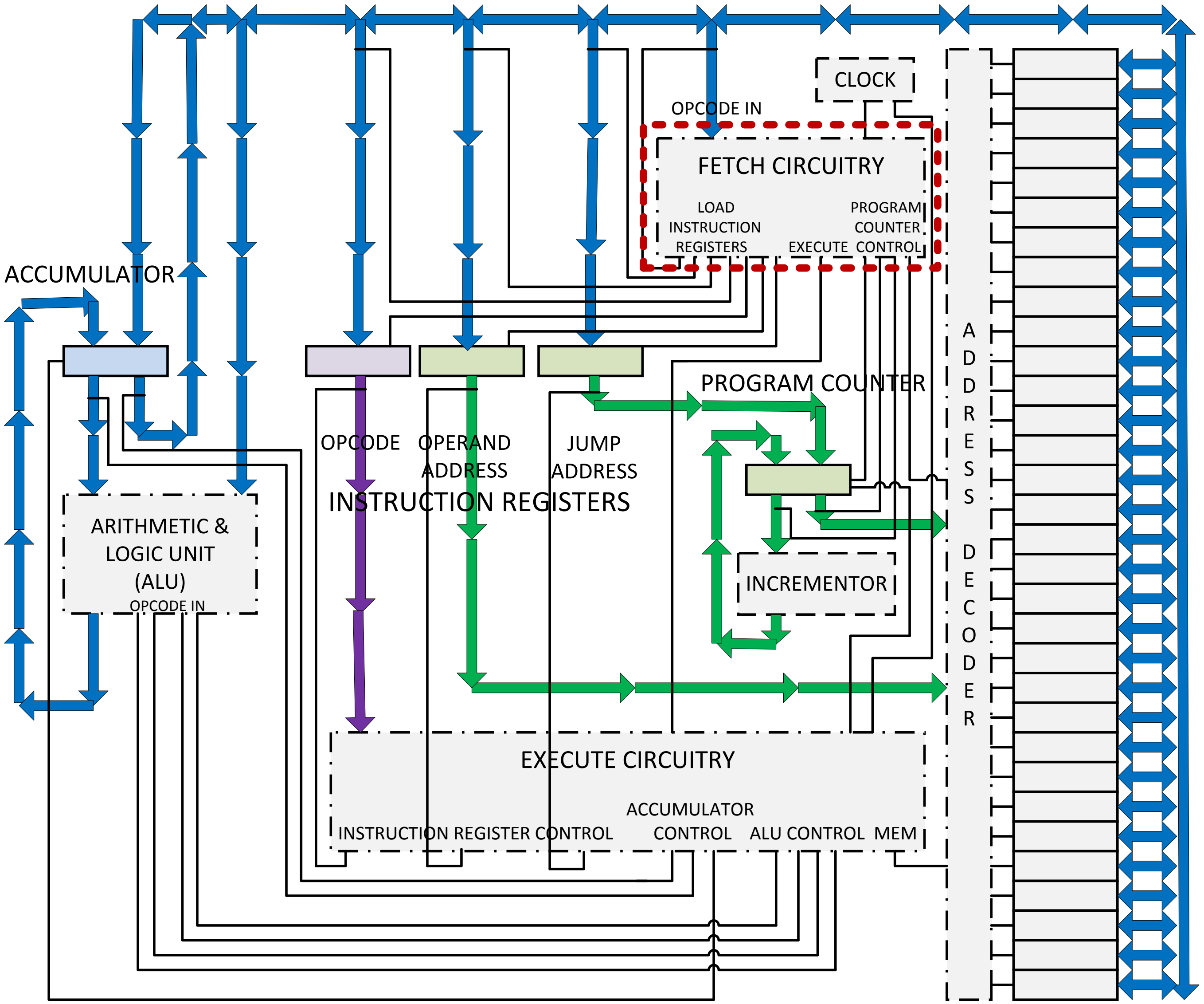
Ring Counter

based on a D Flip-Flops

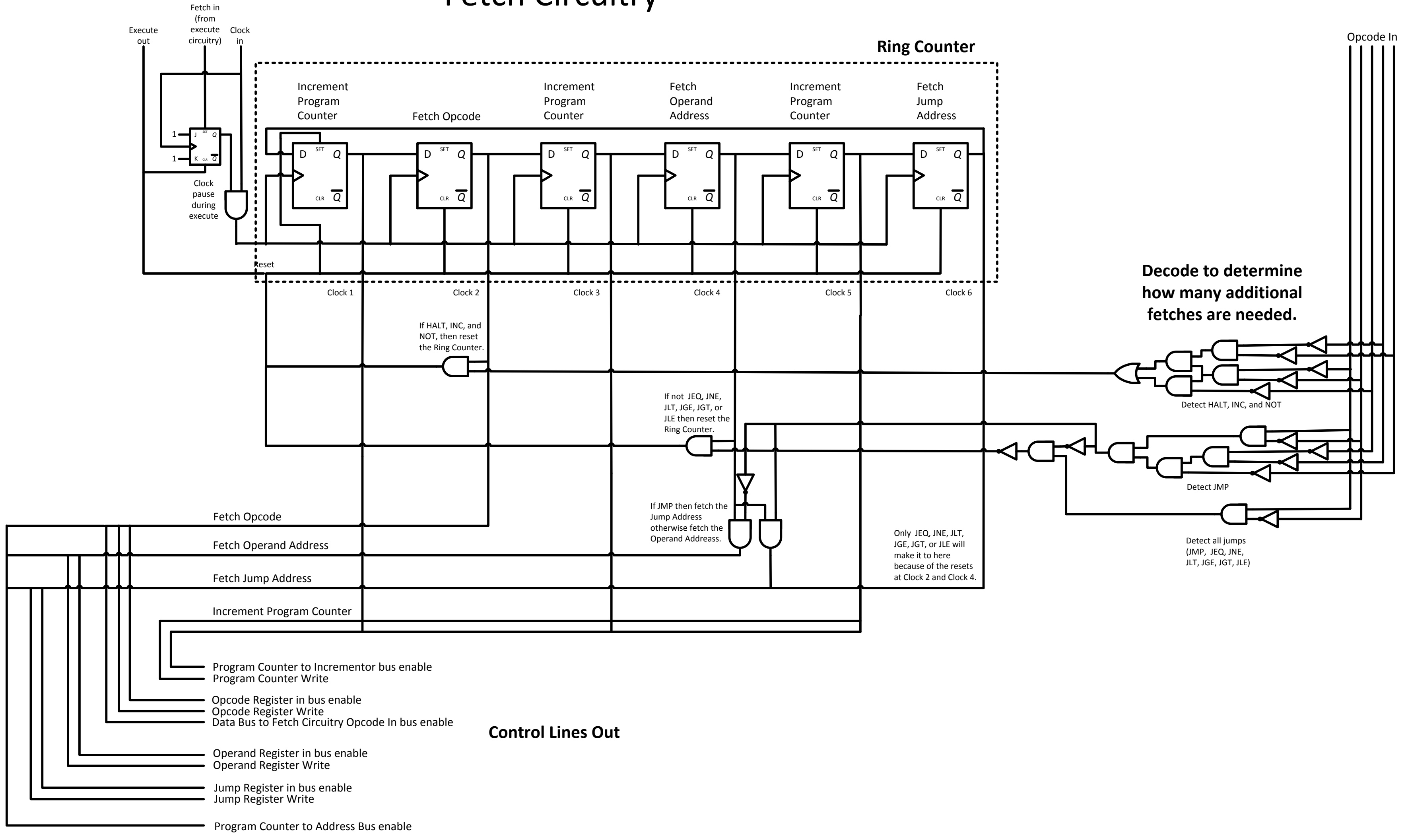


Control Circuitry

Fetch

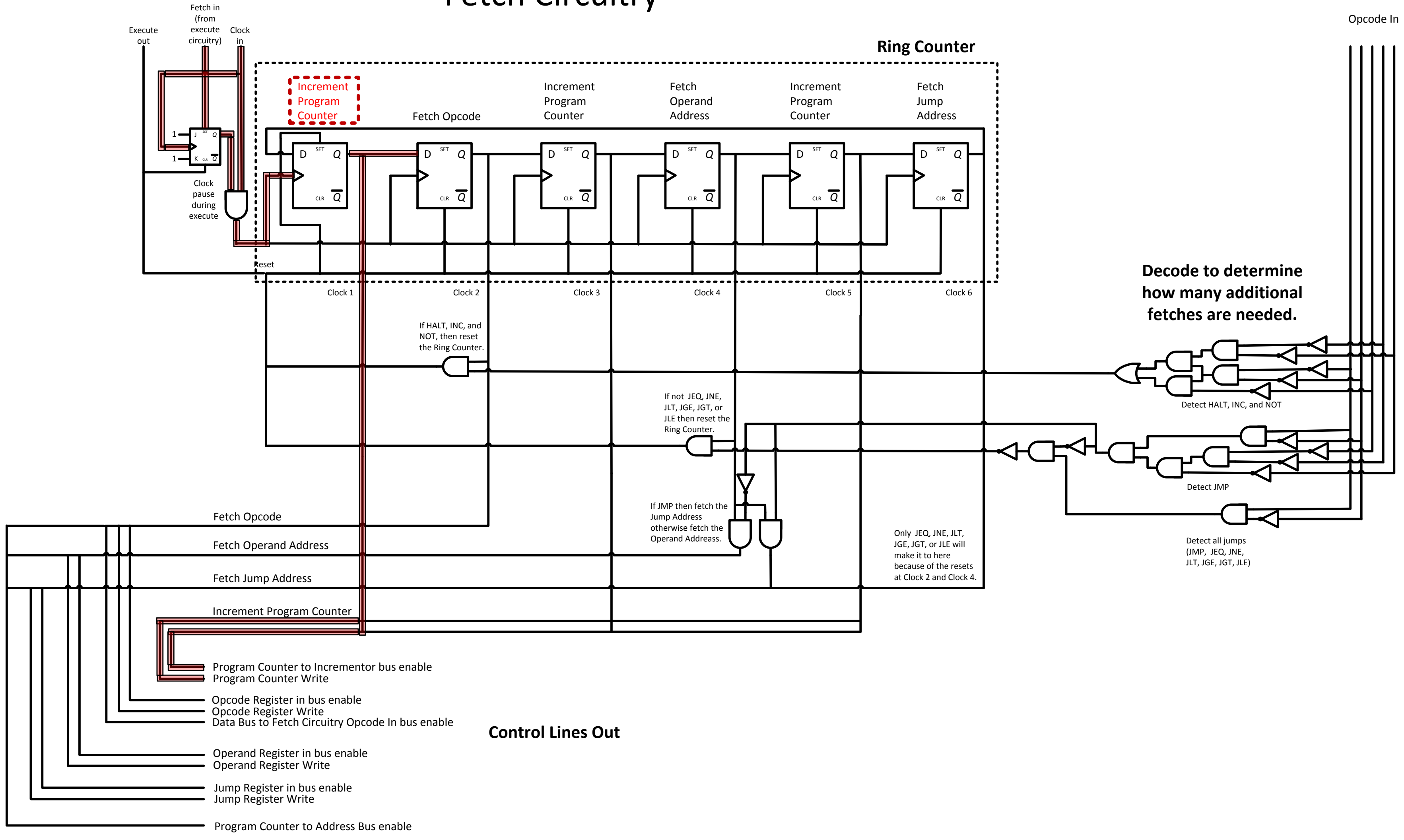

CLOCK
OPCODE IN
FETCH CIRCUITRY
LOAD INSTRUCTION REGISTERS
PROGRAM COUNTER
EXECUTE CONTROL
ACCUMULATOR
PROGRAM COUNTER
OPCODE
OPERAND ADDRESS
JUMP ADDRESS
INSTRUCTION REGISTERS
ARITHMETIC & LOGIC UNIT (ALU)
OPCODE IN
INCREMENTOR
ADDRESS DECODER
EXECUTE CIRCUITRY
INSTRUCTION REGISTER CONTROL
ACCUMULATOR CONTROL
ALU CONTROL
MEM

Fetch Circuitry

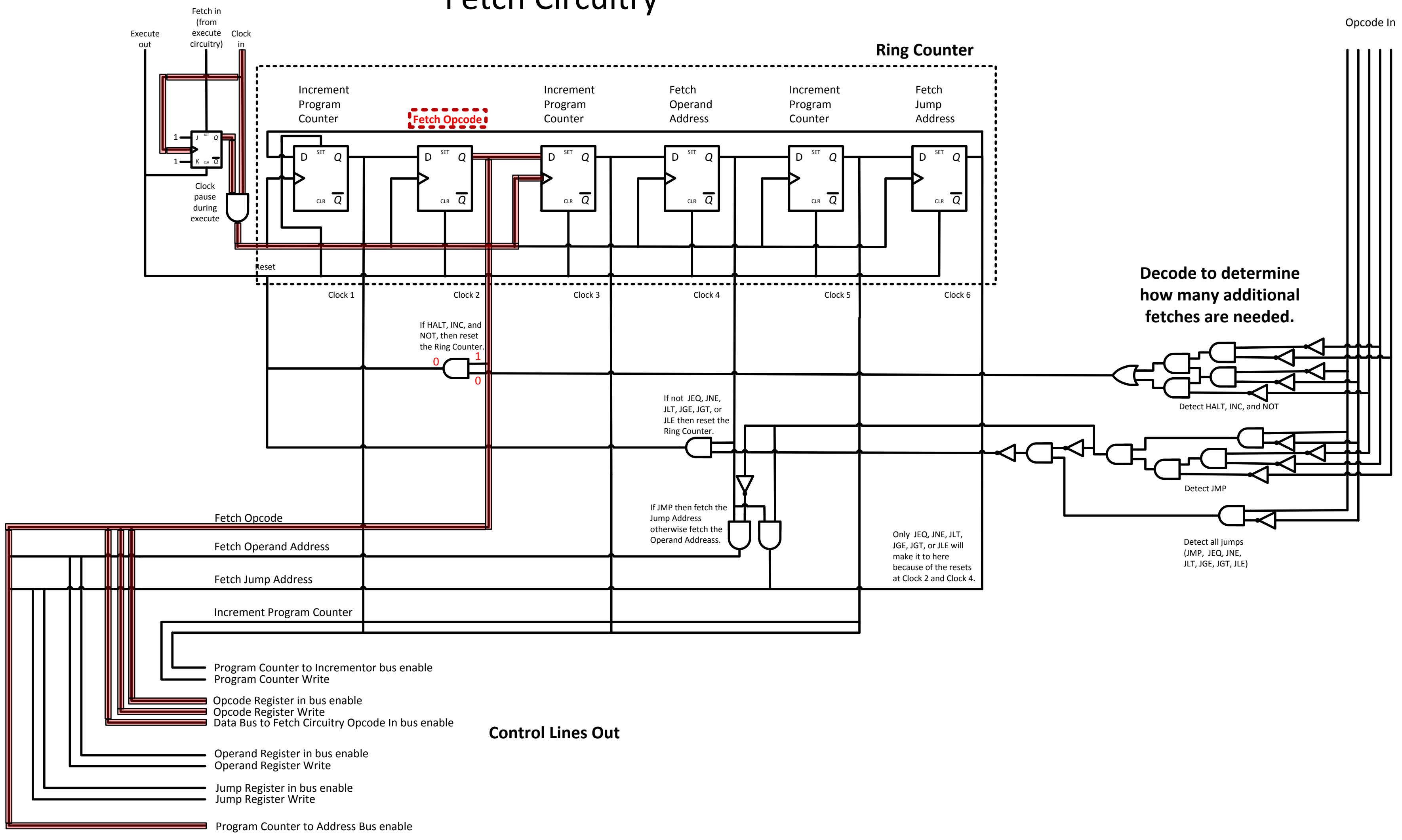


Execute out
Fetch in (from execute circuitry)
Clock in
1
1
J
K
SET
CLR
Q
Clock pause during execute
Reset
Ring Counter
Increment Program Counter
Fetch Opcode
Increment Program Counter
Fetch Operand Address
Increment Program Counter
Fetch Jump Address
D
Q
Clock 1
Clock 2
Clock 3
Clock 4
Clock 5
Clock 6
Opcode In
Decode to determine how many additional fetches are needed.
If HALT, INC, and NOT, then reset the Ring Counter.
Detect HALT, INC, and NOT
If not JEQ, JNE, JLT, JGE, JGT, or JLE then reset the Ring Counter.
Detect JMP
If JMP then fetch the Jump Address otherwise fetch the Operand Addreass.
Only JEQ, JNE, JLT, JGE, JGT, or JLE will make it to here because of the resets at Clock 2 and Clock 4.
Detect all jumps (JMP, JEQ, JNE, JLT, JGE, JGT, JLE)
Fetch Opcode
Fetch Operand Address
Fetch Jump Address
Increment Program Counter
Program Counter to Incrementor bus enable
Program Counter Write
Opcode Register in bus enable
Opcode Register Write
Data Bus to Fetch Circuitry Opcode In bus enable
Control Lines Out
Operand Register in bus enable
Operand Register Write
Jump Register in bus enable
Jump Register Write
Program Counter to Address Bus enable

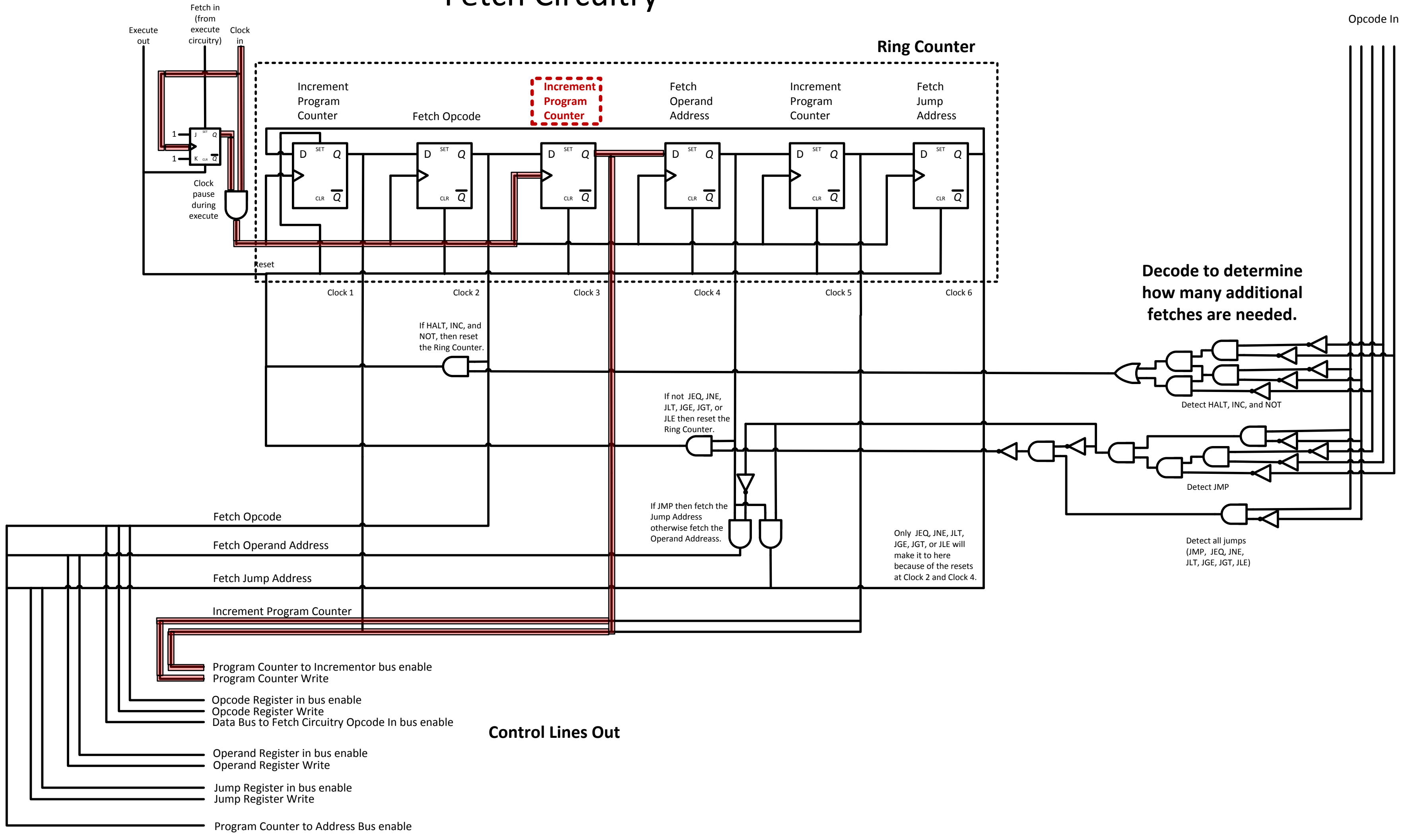
Fetch Circuitry


Execute out
Fetch in (from execute circuitry)
Clock in
Clock pause during execute
Opcode In
Ring Counter
Increment Program Counter
Fetch Opcode
Increment Program Counter
Fetch Operand Address
Increment Program Counter
Fetch Jump Address
Reset
Clock 1
Clock 2
Clock 3
Clock 4
Clock 5
Clock 6
If HALT, INC, and NOT, then reset the Ring Counter.
If not JEQ, JNE, JLT, JGE, JGT, or JLE then reset the Ring Counter.
If JMP then fetch the Jump Address otherwise fetch the Operand Addresss.
Only JEQ, JNE, JLT, JGE, JGT, or JLE will make it to here because of the resets at Clock 2 and Clock 4.
Decode to determine how many additional fetches are needed.
Detect HALT, INC, and NOT
Detect JMP
Detect all jumps (JMP, JEQ, JNE, JLT, JGE, JGT, JLE)
Fetch Opcode
Fetch Operand Address
Fetch Jump Address
Increment Program Counter
Program Counter to Incrementor bus enable
Program Counter Write
Opcode Register in bus enable
Opcode Register Write
Data Bus to Fetch Circuitry Opcode In bus enable
Operand Register in bus enable
Operand Register Write
Jump Register in bus enable
Jump Register Write
Program Counter to Address Bus enable
Control Lines Out

Fetch Circuitry

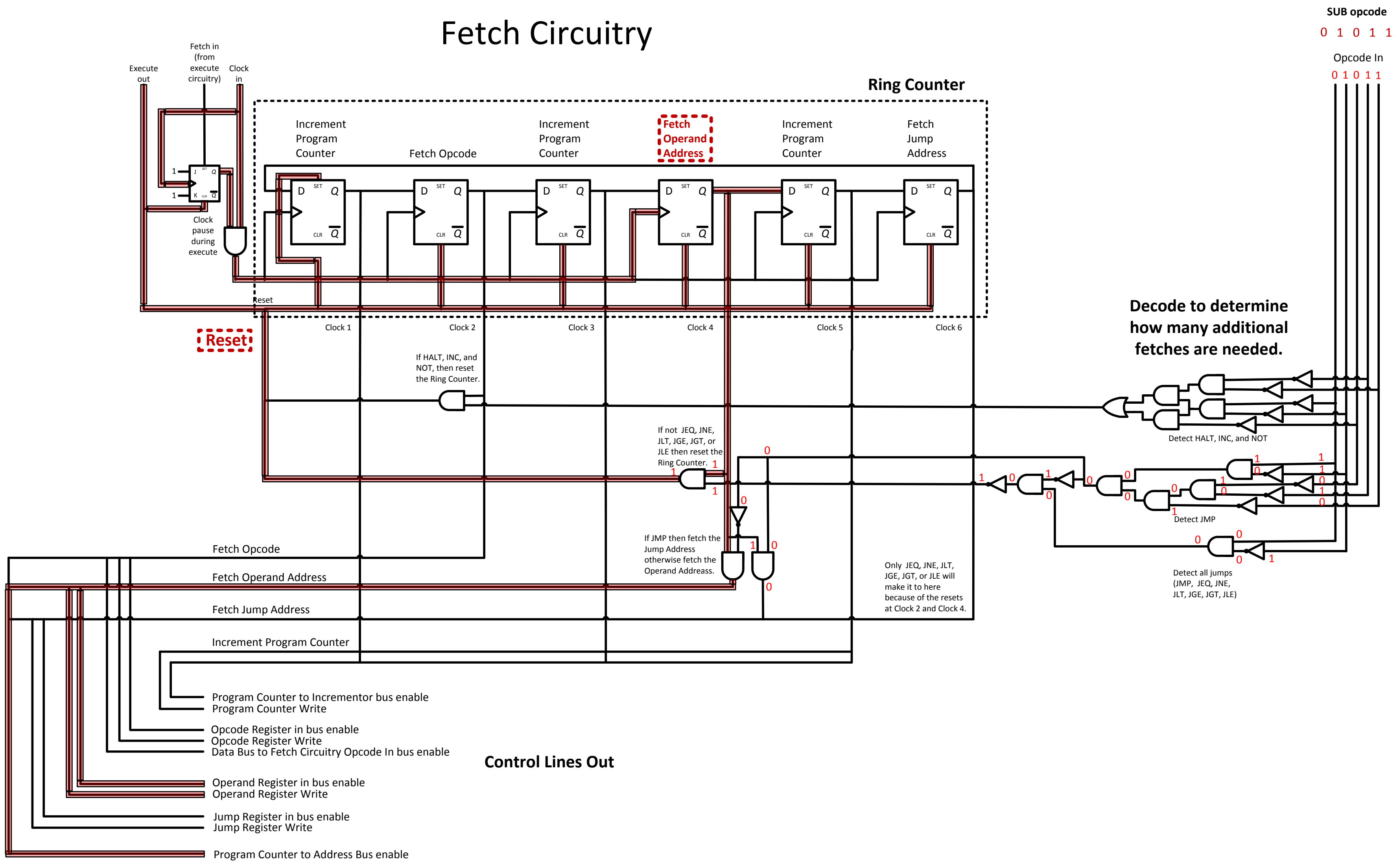

Execute out
Fetch in (from execute circuitry)
Clock in
1
1
Clock pause during execute
Ring Counter
Increment Program Counter
Fetch Opcode
Increment Program Counter
Fetch Operand Address
Increment Program Counter
Fetch Jump Address
D SET Q CLR Q
Reset
Clock 1
Clock 2
Clock 3
Clock 4
Clock 5
Clock 6
Opcode In
Decode to determine how many additional fetches are needed.
If HALT, INC, and NOT, then reset the Ring Counter.
0
1
0
Detect HALT, INC, and NOT
If not JEQ, JNE, JLT, JGE, JGT, or JLE then reset the Ring Counter.
Detect JMP
If JMP then fetch the Jump Address otherwise fetch the Operand Addreass.
Only JEQ, JNE, JLT, JGE, JGT, or JLE will make it to here because of the resets at Clock 2 and Clock 4.
Detect all jumps (JMP, JEQ, JNE, JLT, JGE, JGT, JLE)
Fetch Opcode
Fetch Operand Address
Fetch Jump Address
Increment Program Counter
Program Counter to Incrementor bus enable
Program Counter Write
Opcode Register in bus enable
Opcode Register Write
Data Bus to Fetch Circuitry Opcode In bus enable
Control Lines Out
Operand Register in bus enable
Operand Register Write
Jump Register in bus enable
Jump Register Write
Program Counter to Address Bus enable

Fetch Circuitry



Execute out
Fetch in (from execute circuitry)
Clock in
Ring Counter
Opcode In
Increment Program Counter
Fetch Opcode
Increment Program Counter
Fetch Operand Address
Increment Program Counter
Fetch Jump Address
1
1
J
K
Q
SET
CLR
Clock pause during execute
D
Q
Reset
Clock 1
Clock 2
Clock 3
Clock 4
Clock 5
Clock 6
Decode to determine how many additional fetches are needed.
If HALT, INC, and NOT, then reset the Ring Counter.
Detect HALT, INC, and NOT
If not JEQ, JNE, JLT, JGE, JGT, or JLE then reset the Ring Counter.
Detect JMP
If JMP then fetch the Jump Address otherwise fetch the Operand Address.
Only JEQ, JNE, JLT, JGE, JGT, or JLE will make it to here because of the resets at Clock 2 and Clock 4.
Detect all jumps (JMP, JEQ, JNE, JLT, JGE, JGT, JLE)
Fetch Opcode
Fetch Operand Address
Fetch Jump Address
Increment Program Counter
Program Counter to Incrementor bus enable
Program Counter Write
Opcode Register in bus enable
Opcode Register Write
Data Bus to Fetch Circuitry Opcode In bus enable
Control Lines Out
Operand Register in bus enable
Operand Register Write
Jump Register in bus enable
Jump Register Write
Program Counter to Address Bus enable

Fetch Circuitry


SUB opcode
0 1 0 1 1
Opcode In
0 1 0 1 1
Ring Counter
Execute out
Fetch in (from execute circuitry)
Clock in
Clock pause during execute
Increment Program Counter
Fetch Opcode
Increment Program Counter
Fetch Operand Address
Increment Program Counter
Fetch Jump Address
Reset
Reset
Clock 1
Clock 2
Clock 3
Clock 4
Clock 5
Clock 6
Decode to determine how many additional fetches are needed.
Detect HALT, INC, and NOT
Detect JMP
Detect all jumps (JMP, JEQ, JNE, JLT, JGE, JGT, JLE)
Fetch Opcode
Fetch Operand Address
Fetch Jump Address
Increment Program Counter
Control Lines Out

If HALT, INC, and NOT, then reset the Ring Counter.

If not JEQ, JNE, JLT, JGE, JGT, or JLE then reset the Ring Counter.

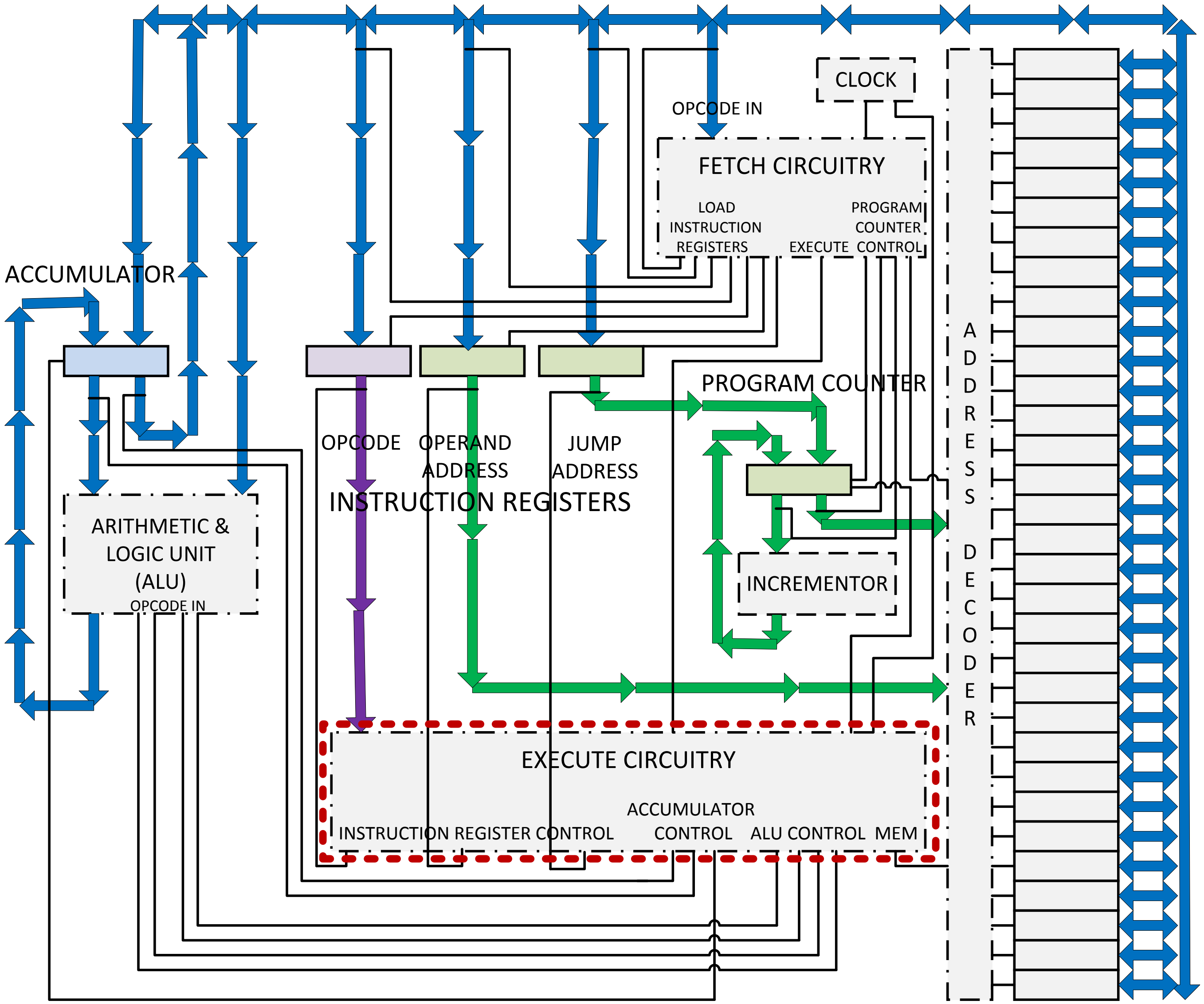
If JMP then fetch the Jump Address otherwise fetch the Operand Addreass.

Only JEQ, JNE, JLT, JGE, JGT, or JLE will make it to here because of the resets at Clock 2 and Clock 4.

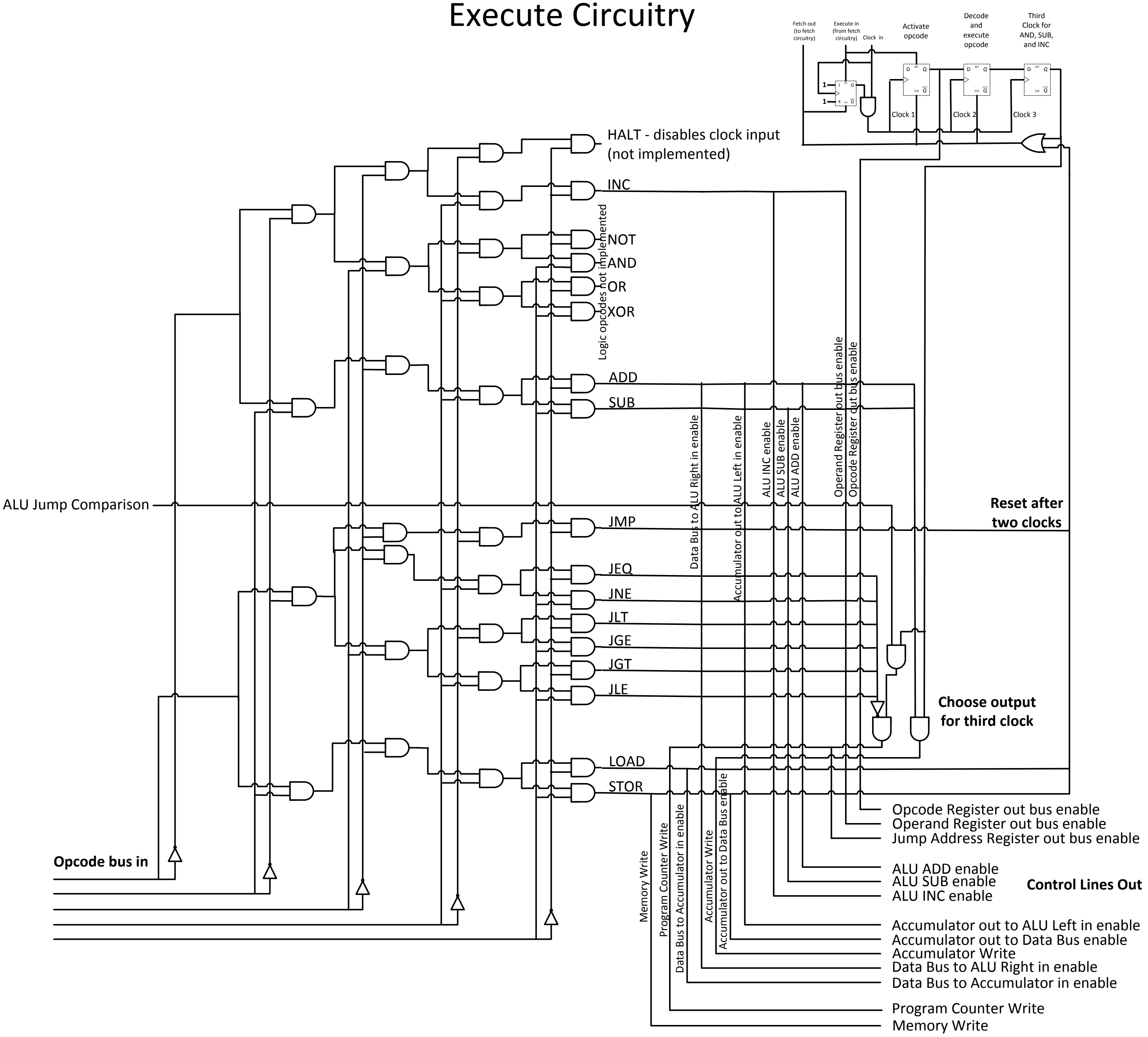
Program Counter to Incrementor bus enable
Program Counter Write
Opcode Register in bus enable
Opcode Register Write
Data Bus to Fetch Circuitry Opcode In bus enable
Operand Register in bus enable
Operand Register Write
Jump Register in bus enable
Jump Register Write
Program Counter to Address Bus enable

Control Circuitry

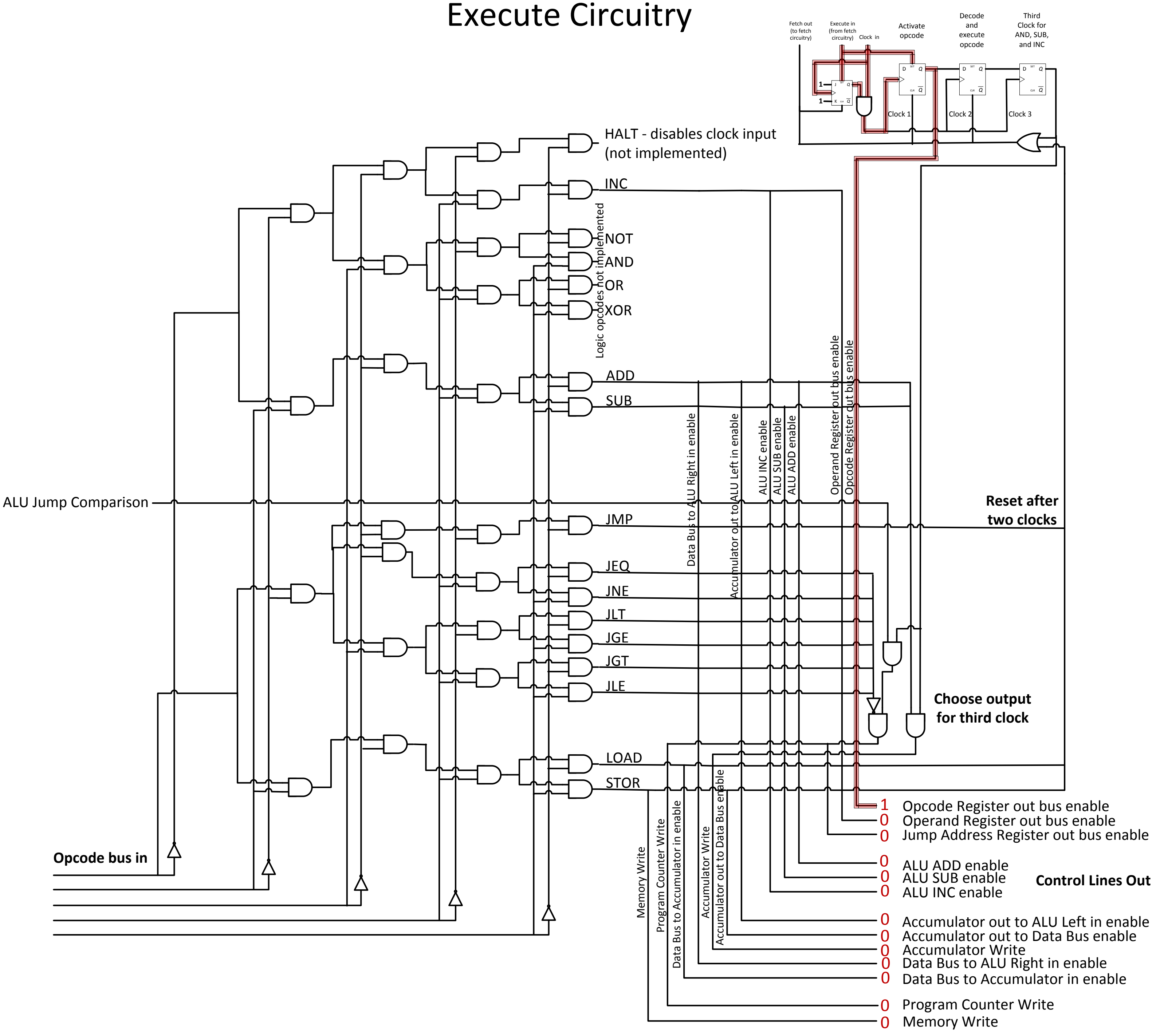
Execute


CLOCK
OPCODE IN
FETCH CIRCUITRY
LOAD INSTRUCTION REGISTERS
PROGRAM COUNTER CONTROL
EXECUTE
ACCUMULATOR
PROGRAM COUNTER
OPCODE
OPERAND ADDRESS
JUMP ADDRESS
INSTRUCTION REGISTERS
INCREMENTOR
ARITHMETIC & LOGIC UNIT (ALU)
OPCODE IN
ADDRESS DECODER
EXECUTE CIRCUITRY
INSTRUCTION REGISTER CONTROL
ACCUMULATOR CONTROL
ALU CONTROL
MEM

Execute Circuitry

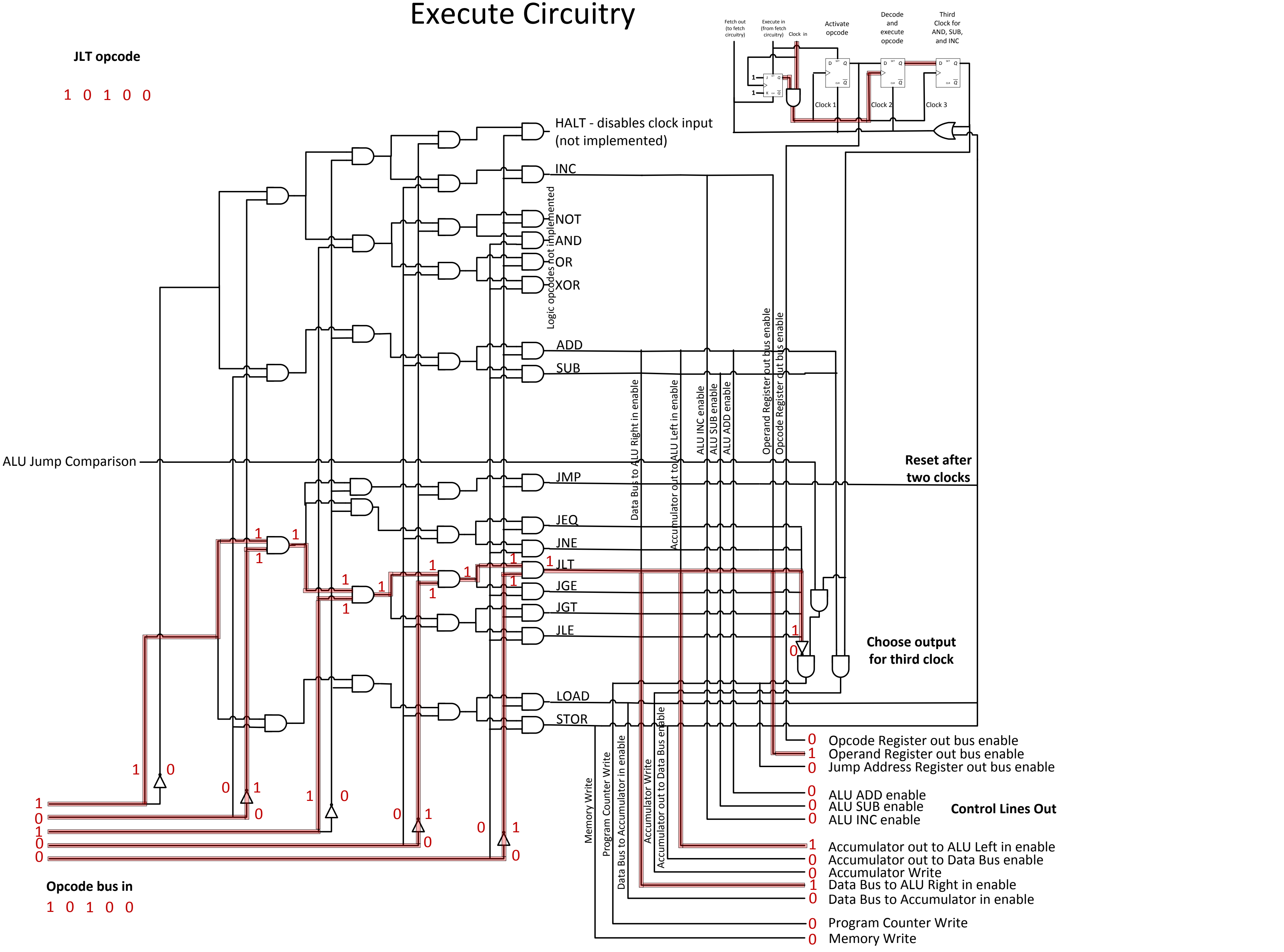

Fetch out (to fetch circuitry)
Execute in (from fetch circuitry)
Clock in
Activate opcode
Decode and execute opcode
Third Clock for AND, SUB, and INC
1
1
J
K
SET
CLR
Q
D
Clock 1
Clock 2
Clock 3
HALT - disables clock input (not implemented)
INC
Logic opcodes not implemented
NOT
AND
OR
XOR
ADD
SUB
Data Bus to ALU Right in enable
Accumulator out to ALU Left in enable
ALU INC enable
ALU SUB enable
ALU ADD enable
Operand Register out bus enable
Opcode Register out bus enable
ALU Jump Comparison
Reset after two clocks
JMP
JEQ
JNE
JLT
JGE
JGT
JLE
Choose output for third clock
LOAD
STOR
Opcode bus in
Memory Write
Program Counter Write
Data Bus to Accumulator in enable
Accumulator Write
Accumulator out to Data Bus enable
Opcode Register out bus enable
Operand Register out bus enable
Jump Address Register out bus enable
ALU ADD enable
ALU SUB enable
ALU INC enable
Control Lines Out
Accumulator out to ALU Left in enable
Accumulator out to Data Bus enable
Accumulator Write
Data Bus to ALU Right in enable
Data Bus to Accumulator in enable
Program Counter Write
Memory Write

Execute Circuitry

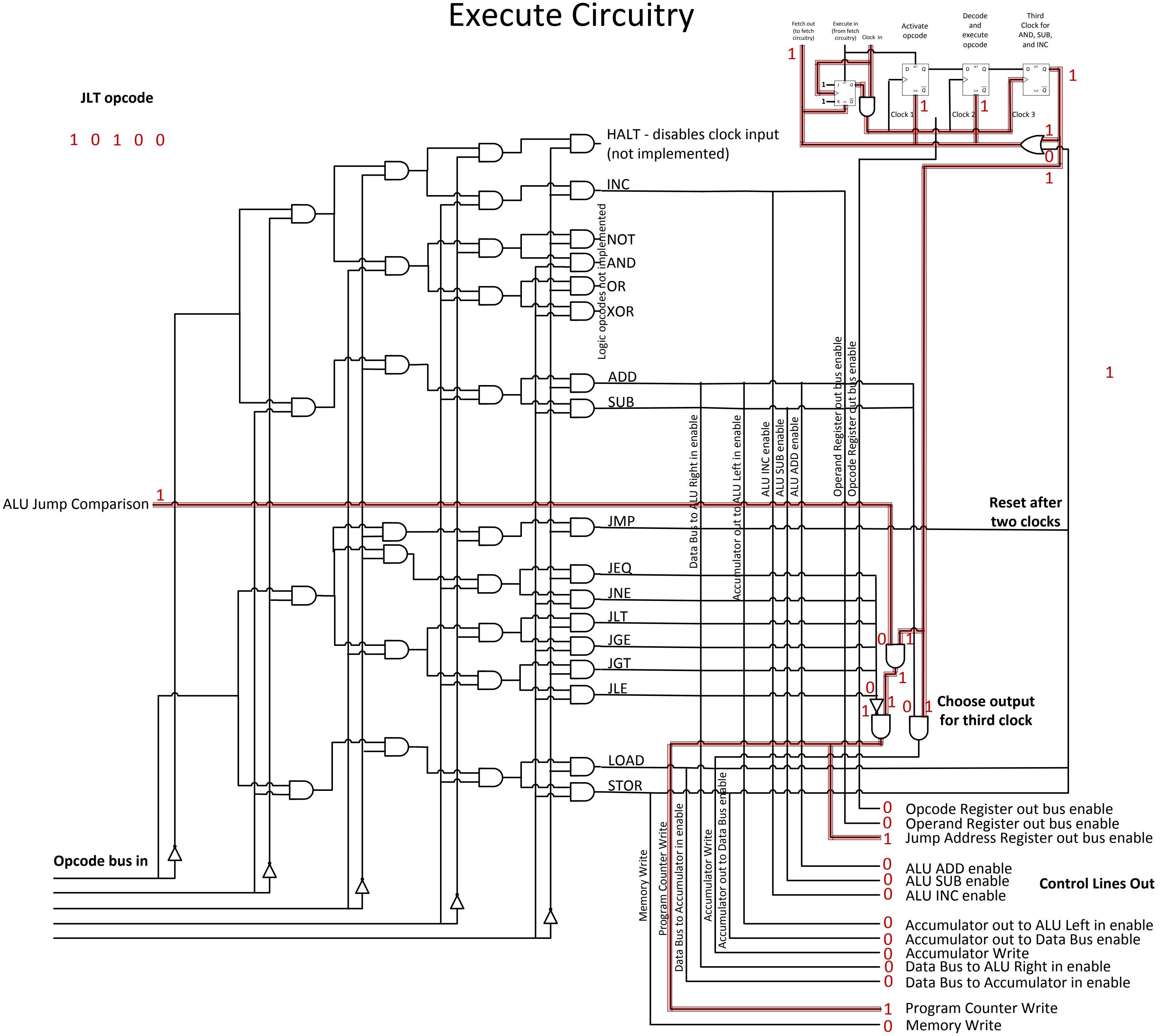


Fetch out (to fetch circuitry)
Execute in (from fetch circuitry)
Clock in
Activate opcode
Decode and execute opcode
Third Clock for AND, SUB, and INC
1
1
J
K
SET
CLR
Q
$\overline{Q}$
D
Clock 1
Clock 2
Clock 3
HALT - disables clock input (not implemented)
INC
Logic opcodes not implemented
NOT
AND
OR
XOR
ADD
SUB
Data Bus to ALU Right in enable
Accumulator out to ALU Left in enable
ALU INC enable
ALU SUB enable
ALU ADD enable
Operand Register out bus enable
Opcode Register out bus enable
ALU Jump Comparison
Reset after two clocks
JMP
JEQ
JNE
JLT
JGE
JGT
JLE
Choose output for third clock
LOAD
STOR
Opcode bus in
Memory Write
Program Counter Write
Data Bus to Accumulator in enable
Accumulator Write
Accumulator out to Data Bus enable
1 Opcode Register out bus enable
0 Operand Register out bus enable
0 Jump Address Register out bus enable
0 ALU ADD enable
0 ALU SUB enable
0 ALU INC enable
Control Lines Out
0 Accumulator out to ALU Left in enable
0 Accumulator out to Data Bus enable
0 Accumulator Write
0 Data Bus to ALU Right in enable
0 Data Bus to Accumulator in enable
0 Program Counter Write
0 Memory Write

Execute Circuitry


JLT opcode
1 0 1 0 0
Fetch out (to fetch circuitry)
Execute in (from fetch circuitry)
Clock in
Activate opcode
Decode and execute opcode
Third Clock for AND, SUB, and INC
Clock 1
Clock 2
Clock 3
HALT - disables clock input (not implemented)
INC
NOT
AND
OR
XOR
Logic opcodes not implemented
ADD
SUB
ALU Jump Comparison
JMP
JEQ
JNE
JLT
JGE
JGT
JLE
LOAD
STOR
Data Bus to ALU Right in enable
Accumulator out to ALU Left in enable
ALU INC enable
ALU SUB enable
ALU ADD enable
Operand Register out bus enable
Opcode Register out bus enable
Reset after two clocks
Choose output for third clock
Memory Write
Program Counter Write
Data Bus to Accumulator in enable
Accumulator Write
Accumulator out to Data Bus enable
0 Opcode Register out bus enable
1 Operand Register out bus enable
0 Jump Address Register out bus enable
0 ALU ADD enable
0 ALU SUB enable
0 ALU INC enable
Control Lines Out
1 Accumulator out to ALU Left in enable
0 Accumulator out to Data Bus enable
0 Accumulator Write
1 Data Bus to ALU Right in enable
0 Data Bus to Accumulator in enable
0 Program Counter Write
0 Memory Write
Opcode bus in
1 0 1 0 0

Execute Circuitry



JLT opcode
1 0 1 0 0
Fetch out (to fetch circuitry)
Execute in (from fetch circuitry)
Clock in
Activate opcode
Decode and execute opcode
Third Clock for AND, SUB, and INC
Clock 1
Clock 2
Clock 3
HALT - disables clock input (not implemented)
INC
NOT
AND
OR
XOR
Logic opcodes not implemented
ADD
SUB
Data Bus to ALU Right in enable
Accumulator out to ALU Left in enable
ALU INC enable
ALU SUB enable
ALU ADD enable
Operand Register out bus enable
Opcode Register out bus enable
ALU Jump Comparison
Reset after two clocks
JMP
JEQ
JNE
JLT
JGE
JGT
JLE
Choose output for third clock
LOAD
STOR
Opcode bus in
Memory Write
Program Counter Write
Data Bus to Accumulator in enable
Accumulator Write
Accumulator out to Data Bus enable
0 Opcode Register out bus enable
0 Operand Register out bus enable
1 Jump Address Register out bus enable
0 ALU ADD enable
0 ALU SUB enable
0 ALU INC enable
Control Lines Out
0 Accumulator out to ALU Left in enable
0 Accumulator out to Data Bus enable
0 Accumulator Write
0 Data Bus to ALU Right in enable
0 Data Bus to Accumulator in enable
1 Program Counter Write
0 Memory Write

Next Presentation:
Execution Sequence, individual clock cycles

End of Presentation