ECSE 549
EXPERT SYSTEMS IN ELECTRICAL ENGINEERING DESGIN

SUPERVISED BY
PROF. DAVID LOWTHER

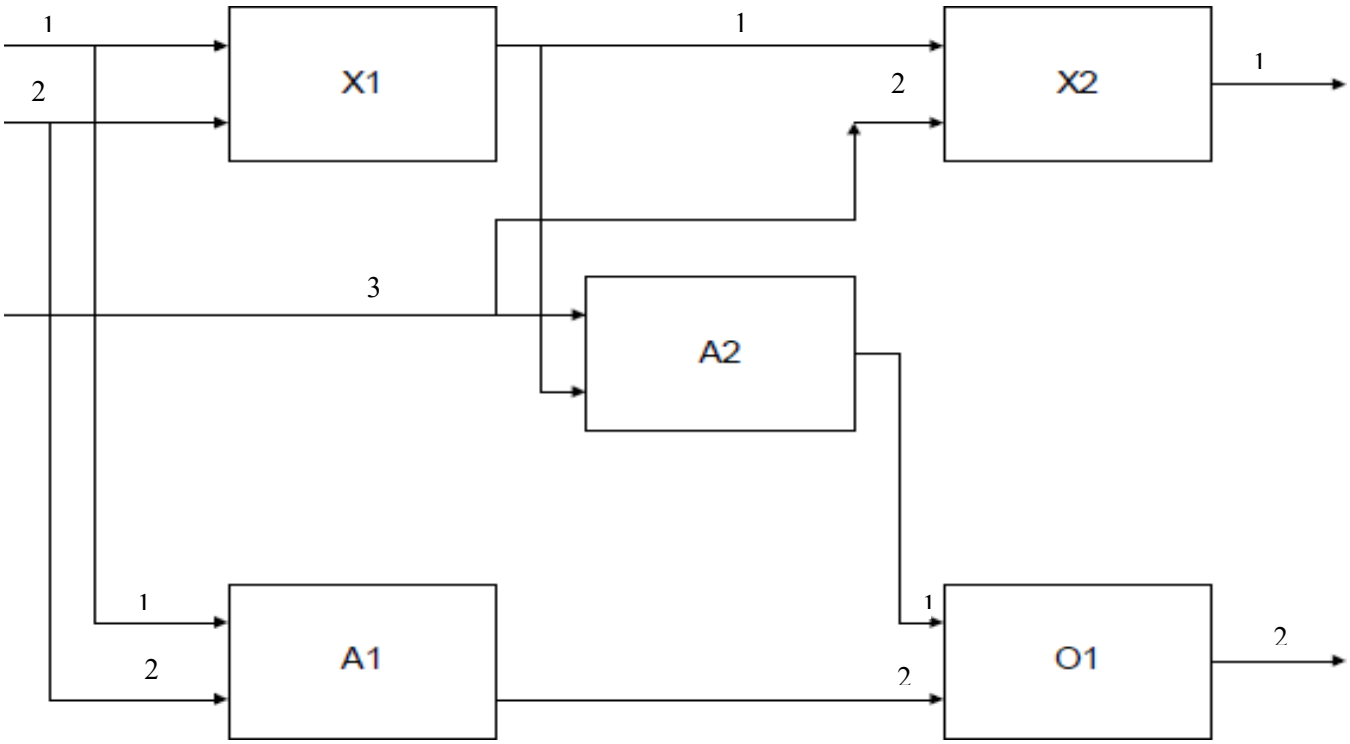
ASSIGNMENT 2

NAME: ***Student ID***
EBENEZER BAMIDELE AJAYI ***260888203***

Monday, March 09, 2020
WINTER 2020

MODELLING OF 1-BIT ADDER CIRCUIT

The program for modelling of a 1-bit adder circuit using JESS has been written and made available in appendix A1 stored as *Adder_CCT_Modelling .clp* file. The device has 3 inputs namely, "*IN_A*", "*IN_B*" and "*Carry_in*" respectively and 2 outputs labeled as "*Sum bit*" and "*C-out*" respectively.



STEP 1 (CIRCUIT SET UP AND CONNECTIONS):

The three inputs for this system were defined namely *IN_A*, *IN_B* and *Carry_in*. The *initial-fact* is asserted by the reset command. It is used internally by Jess to keep track of its own operations.

```
;The three input for the system
(defrule inputs(initial-fact)
     =>
     (printout t crlf "INPUT VALUES")
     (printout t crlf "IN_A?(0,1):")
     (assert
          (IN 1 F 1 =(read)))
     (printout t crlf "IN_B?(0,1):")
     (assert
          (IN 2 F 1 =(read)))
     (printout t crlf "Carry_in(0,1):")
     (assert
          (IN 3 F 1 =(read))))
```

Then, the input connections were set up with the first input labelled as *IN 1* serving as an input to both the first XOR and AND gates labelled as X1 and A1 respectively, then followed by the second input as shown in the screenshot provided below. X, A and O represent XOR, AND and OR gates respectively.

```
;Connections INPUT
;;First Input for X1 and A1
(defrule INPUT1-F1-X1-A1
     (IN 1 F 1 ?a)
     =>
     (assert(IN 1 X 1 ?a))
     (assert(IN 1 A 1 ?a)))

;; Input2 for X1 and A1
(defrule INPUT2-F1-X1-A1
     (IN 2 F 1 ?a)
     =>
     (assert(IN 2 X 1 ?a))
     (assert(IN 2 A 1 ?a)))

;;Carry input for X2 and A2
(defrule INPUT3-F1-X2-A2
     (IN 3 F 1 ?a)
     =>
     (assert(IN 1 A 2 ?a))
     (assert(IN 2 X 2 ?a)))


;OUTPUT
;;Output of X1 for X2 and A2
(defrule OUTPUTX1-X2-A2
     (OUT X 1 ?a)
     =>
     (assert(IN 1 X 2 ?a))
     (assert(IN 2 A 2 ?a)))

;;Output of A1 for O1
(defrule OUTPUTA1-O1
     (OUT A 1 ?a)
    =>(assert(IN 2 O 1 ?a)))

;;Output of A2 for O1
(defrule OUTPUTA2-O1
     (OUT A 2 ?a)
     =>
     (assert(IN 1 O 1 ?a)))

;OVERALL OUTPUT
;;Output of X2 as device output-1
(defrule OUTPUTX2-device1
     (OUT X 2 ?a)
     =>
     (assert(OUT 1 F 1 ?a)))

;;Output of O1 as device output-2
(defrule OUTPUTO1-device2
     (OUT O 1 ?a)
     =>
     (assert(OUT 2 F 1 ?a)))
```

STEP 2 (Creation of rules for each gate)

I. The XOR gates X1 and X2 were defined based on their truth table as shown below:

IN_A	IN_B	OUTPUT
0	0	0
0	1	1
1	0	1
1	1	0

Logic used: If IN A and IN B have the same value, the output should be 0, otherwise 1. That is, if IN A = 0 and IN B = 0 or IN A = 1 and IN B = 1, the gate output is 0, else 1.

```
;;FOR XOR GATE
(defrule X
     (IN 1 X ?gate ?a1)
     (IN 2 X ?gate ?a2)
     =>
     (if
          (or
               (and
                    (= ?a1 0)  (= ?a2 0))
               (and
                    (=?a1 1)  (=?a2 1)))
          then
          (assert(OUT X ?gate 0))
          else
          (assert(OUT X ?gate 1))))
```

II. The AND gates A1 and A2 were defined based on their truth table as shown below:

IN_A	IN_B	OUTPUT
0	0	0
0	1	0
1	0	0
1	1	1

Logic used: An AND gate is an “all or none” gate, that is, output is TRUE only when both inputs are TRUE. If IN_A = 1 and IN_B = 1, the gate output is 1, else 0.

```
;;FOR AND GATE
(defrule A
     (IN 1 A ?gate ?a1)
     (IN 2 A ?gate ?a2)
     =>
     (if
          (and
               (=?a1 1)  (=?a2 1))
          then
          (assert(OUT A ?gate 1))
          else
          (assert(OUT A ?gate 0))))
```

III. The OR gate O1 is defined based on its truth table as shown below:

IN_A	IN_B	OUTPUT
0	0	0
0	1	1
1	0	1
1	1	1

Logic used: An OR gate is an "any or all" gate, that is, output value is TRUE only when at least one of the inputs is TRUE. If IN A = 0 or IN B = 0, the gate output is 0, else 1.

```
;;FOR OR GATE
(defrule O
     (IN 1 O ?gate ?a1)
     (IN 2 O ?gate ?a2)
     =>
     (if
          (and
               (=?a1 0)  (=?a2 0))
          then
          (assert(OUT O ?gate 0))
          else
          (assert(OUT O ?gate 1))))
```

With three inputs, we are expected to have 8 different inputs combination. The result of the rule defined are:

(1)

```
;OUTPUT RESULT
(defrule output
     (OUT 1 F 1 ?a1)
     (OUT 2 F 1 ?a2)
     =>
     (printout t crlf "OUTPUT RESULT:")
```

Problems Javadoc Declaration Console

<terminated> Ass_2.clp [Jess Application] C:\Program Files\Java\jre1.8.0_241\bin\javaw.exe (2

```
INPUT VALUES
IN_A?(0,1):0

IN_B?(0,1):0

Carry_in(0,1):0

OUTPUT RESULT:
Sum bit = 0
C-Out = 0
```

(2)

```
;OUTPUT RESULT
(defrule output
     (OUT 1 F 1 ?a1)
     (OUT 2 F 1 ?a2)
     =>
     (printout t crlf "OUTPUT RESULT:")
```

Problems Javadoc Declaration Console

<terminated> Ass_2.clp [Jess Application] C:\Program Files\Java\jre1.8.0_241\bin\javaw.exe (2

```
INPUT VALUES
IN_A?(0,1):0

IN_B?(0,1):0

Carry_in(0,1):1

OUTPUT RESULT:
Sum bit = 1
C-Out = 0
```

(3)

```
;OUTPUT RESULT
(defrule output
     (OUT 1 F 1 ?a1)
     (OUT 2 F 1 ?a2)
     =>
     (printout t crlf "OUTPUT RESULT:")
```

Problems | Javadoc | Declaration | Console

<terminated> Ass_2.clp [Jess Application] C:\Program Files\Java\jre1.8.0_241\bin\javaw.exe (2

```
INPUT VALUES
IN_A?(0,1):0

IN_B?(0,1):1

Carry_in(0,1):0

OUTPUT RESULT:
Sum bit = 1
C-Out = 0
```

(4)

```
     (OUT 1 F 1 ?a1)
     (OUT 2 F 1 ?a2)
     =>
     (printout t crlf "OUTPUT RESULT:")
     (printout t crlf "Sum bit = "?a1"")
     (printout t crlf "C-Out = "?a2""))
(reset)
(run)
```

Problems | Javadoc | Declaration | Console

<terminated> Ass_2.clp [Jess Application] C:\Program Files\Java\jre1.8.0_241\bin\javaw.exe (2

```
INPUT VALUES
IN_A?(0,1):0

IN_B?(0,1):1

Carry_in(0,1):1

OUTPUT RESULT:
Sum bit = 0
C-Out = 1
```

(5)

```
(defrule output
     (OUT 1 F 1 ?a1)
     (OUT 2 F 1 ?a2)
     =>
     (printout t crlf "OUTPUT RESULT:")
     (printout t crlf "Sum bit = "?a1"")
     (printout t crlf "C-Out = "?a2""))
(reset)
(run)
```

Problems | Javadoc | Declaration | Console

<terminated> Ass_2.clp [Jess Application] C:\Program Files\Java\jre1.8.0_241\bin\javaw.exe (2

```
INPUT VALUES
IN_A?(0,1):1

IN_B?(0,1):0

Carry_in(0,1):0

OUTPUT RESULT:
Sum bit = 1
C-Out = 0
```

(6)

```
(defrule output
     (OUT 1 F 1 ?a1)
     (OUT 2 F 1 ?a2)
     =>
     (printout t crlf "OUTPUT RESULT:")
     (printout t crlf "Sum bit = "?a1"")
     (printout t crlf "C-Out = "?a2""))
(reset)
```

Problems | Javadoc | Declaration | Console

<terminated> Ass_2.clp [Jess Application] C:\Program Files\Java\jre1.8.0_241\bin\javaw.exe (20

```
INPUT VALUES
IN_A?(0,1):1

IN_B?(0,1):0

Carry_in(0,1):1

OUTPUT RESULT:
Sum bit = 0
C-Out = 1
```

(7)

```
;OUTPUT RESULT
(defrule output
     (OUT 1 F 1 ?a1)
     (OUT 2 F 1 ?a2)
     =>
     (printout t crlf "OUTPUT RESULT:")
     (printout t crlf "Sum bit = "?a1"")
     (printout t crlf "C-Out = "?a2""))
(reset)
```

Problems | Javadoc | Declaration | Console

<terminated> Ass_2.clp [Jess Application] C:\Program Files\Java\jre1.8.0_241\bin\javaw.exe (2

```
INPUT VALUES
IN_A?(0,1):1

IN_B?(0,1):1

Carry_in(0,1):0

OUTPUT RESULT:
Sum bit = 0
C-Out = 1
```

(8)

```
     =>
     (printout t crlf "OUTPUT RESULT:")
     (printout t crlf "Sum bit = "?a1"")
     (printout t crlf "C-Out = "?a2""))
(reset)
(run)
```

Problems | Javadoc | Declaration | Console

<terminated> Ass_2.clp [Jess Application] C:\Program Files\Java\jre1.8.0_241\bin\javaw.exe (202

```
INPUT VALUES
IN_A?(0,1):1

IN_B?(0,1):1

Carry_in(0,1):1

OUTPUT RESULT:
Sum bit = 1
C-Out = 1
```

The truth table for the 1-bit adder is shown below:

IN_A	IN_B	Carry_in	Sum_bit	C-OUT
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

FAULT DIAGNOSIS

This second code provided in the Appendix (*Fault_diagnosis.clp)* was written with the aim of detecting if there is(are) fault in the adder circuit when a set of inputs together with what the desired outputs should be are inputted into the system. For example, if the inputs to the adder are 0, 1 with a carry of 0, the output should be 1 with a carry out of 0. But if I interchanged the output as a result of an unlikely noticeable error, the system should be able to give a warning by telling the user that error exists in both the X2 gate which is the sum, and the O1 gate which is the C-out. To identify which gate(s) has error, the following steps were performed.

STEP 1:

Firstly, the inputs were provided. Now, the adder's outputs—sum and carryout will also serve as an input.

```
(deftemplate MAIN::problem_X2_gate
   "(Implied)"
   (multislot __data))

(deftemplate MAIN::problem_O1_gate
   "(Implied)"
   (multislot __data))

;Fault Diagnosis
(defrule fault_diagnosis
    (initial-fact)
    =>
    (printout t crlf "INPUT VALUES")
    (printout t crlf "IN_A?(0,1):")
    (assert(IN 1 F 1 =(read)))
    (printout t crlf "IN_B?(0,1):")
    (assert(IN 2 F 1 =(read)))
    (printout t crlf "Carry_in?(0,1):")
    (assert(IN 3 F 1 =(read)))

    (printout t crlf "DESIRED OUTPUTS")
    (printout t crlf "Sum_bit?(0,1):")
    (assert(OUT 1 F 1 =(read)))
    (printout t crlf "C-Out?(0,1):")
    (assert(OUT 2 F 1 =(read))))
```

STEP 2 (DEFINE RULES FOR FAULTS):

Rules were defined for the faults when there is a problem in sum bit and/or the carry out.

1. If there is an error in the sum bit, the system should identify that the error is caused starting with X2 gate and/or the outputs of other gates which serve as an input to X2 gate, that is X1 gate.

   Logic used: If IN_A = IN_B and Carry_in = 1 and Sum_bit = 0, or
   If IN_A = IN_B and Carry_in = 0 and Sum_bit = 1, or
   If IN_A ≠ IN_B and Carry_in = 0 and Sum_bit = 0, or
   If IN_A ≠IN_B and Carry_in = 1 and Sum_bit = 1,
   Then, I should be notified of an error in the system, since the Sum_bit outputs are opposite of what it should be.

Code:

```
;;If there is any error in the sum bit,
;;the error should be detected starting with X2 and focus o the other outputs that were an input to X2
    (if
        (or
            (and
                (= ?IN_1 ?IN_2)
                (= ?Carry_in 1)
                (= ?Sum_bit 0))
            (and
                (= ?IN_1 ?IN_2)
                (= ?Carry_in 0)
                (= ?Sum_bit 1))
            (and
                (neq ?IN_1 ?IN_2)
                (= ?Carry_in 0)
                (= ?Sum_bit 0))
            (and
                (neq ?IN_1 ?IN_2)
                (= ?Carry_in 1)
                (= ?Sum_bit 1)))
        then
        (assert(problem_X2_gate)))
```

Test sample:

```
(defrule print_problem_overflow_bit
    (problem_O1_gate)
    =>
    (printout t crlf "There is a fault detection in O1 gate which could be as a result of the presence of fault in A1, A2 and/or O1"))

(defrule problem_with_X2_and_O1_gates
    (problem_X2_gate)
    (problem_O1_gate)
```

Problems | Javadoc | Declaration | Console

<terminated> u.clp [Jess Application] C:\Program Files\Java\jre1.8.0_241\bin\javaw.exe (2020-03-06 5:14:08 PM)

```
INPUT VALUES
IN_A?(0,1):1

IN_B?(0,1):0

Carry_in?(0,1):1

DESIRED OUTPUTS
Sum_bit?(0,1):1

C-Out?(0,1):1

There is a fault detection in X2 gate which could be as a result of the presence of fault in X1 and/or X2.
```

2. If there is an error in the Carry-Out bit, the system should identify that the error is caused starting with O1 gate and/or the outputs of other gates which served as an input to O1 gate, as well as the output gates that serve as an input to those gates too, that is A1, A2 and X1 gates.

   Logic used: If IN_A = IN_B = 1 and C_Out = 0, or
   If IN_A = IN_B = 0 and C_Out = 1, or
   If IN_A ≠ IN_B and C_Out ≠ Carry_in
   Then, I should be notified of an error in the system, since the Carry outputs are opposite of what it should be.

Code:

```
;;If there is any error in the Carry-out,
;;the error should be detetcted starting with O1 and focus on the other outputs that were an input to O1
    (if
        (or
            (and
                (= ?IN_1 ?IN_2 1)
                (= ?C-Out 0))
            (and
                (= ?IN_1 ?IN_2 0)
                (= ?C-Out 1))
            (and
                (neq ?IN_1 ?IN_2)
                (neq ?C-Out ?Carry_in)))
        then
        (assert(problem_O1_gate))))
```

Sample test:

```
(defrule print_problem_sum_bit
    (problem_X2_gate)
    =>
    (printout t crlf "There is a fault detection in X2 gate which could be as a result of the presence of fault in X1 and/or X2.

(defrule print_problem_Carry_out
    (problem_O1_gate)
    =>
    (printout t crlf "There is a fault detection in O1 gate which could be as a result of the presence of fault in A1, A2, X1 an
```

Problems | Javadoc | Declaration | Console

<terminated> u.clp [Jess Application] C:\Program Files\Java\jre1.8.0_241\bin\javaw.exe (2020-03-07 7:38:28 AM)

```
INPUT VALUES
IN_A?(0,1):1

IN_B?(0,1):0

Carry_in?(0,1):0

DESIRED OUTPUTS
Sum_bit?(0,1):1

C-Out?(0,1):1
There is a fault detection in O1 gate which could be as a result of the presence of fault in A1, A2, X1 and/or O1
```

3. If there is an error in both the sum and C-Out bit, the system should identify that the error is caused starting with both the X2 and O1 gates, and subsequently with every other gate.

```
(defrule print_problem_sum_bit
     (problem_X2_gate)
     =>
     (printout t crlf "There is a fault detection in X2 gate which could be as a result of the presence of fault in X1 and/or X2." crlf)

(defrule print_problem_Carry_out
     (problem_O1_gate)
     =>
```

Problems | Javadoc | Declaration | Console

```
<terminated> u.clp [Jess Application] C:\Program Files\Java\jre1.8.0_241\bin\javaw.exe (2020-03-07 7:43:48 AM)
INPUT VALUES
IN_A?(0,1):0

IN_B?(0,1):1

Carry_in?(0,1):1

DESIRED OUTPUTS
Sum_bit?(0,1):1

C-Out?(0,1):0

There are errors in both the Sum bit and the Carry-Out. Check below for more information:
There is a fault detection in O1 gate which could be as a result of the presence of fault in A1, A2, X1 and/or O1
There is a fault detection in X2 gate which could be as a result of the presence of fault in X1 and/or X2.
```

APPENDIX

A1. *ADDER_CCT_MODELLING.CLP*

```
;The three input for the system
(defrule start(initial-fact)
    =>
    (printout t crlf "INPUT VALUES")
    (printout t crlf "IN_A?(0,1):")
    (assert(IN 1 F 1 =(read)))
    (printout t crlf "IN_B?(0,1):")
    (assert(IN 2 F 1 =(read)))
    (printout t crlf "Carry_in(0,1):")
    (assert(IN 3 F 1 =(read))))

;Connections INPUT
;;First Input for X1 and A1
(defrule INPUT1-F1-X1-A1(IN 1 F 1 ?a)
    =>
     (assert(IN 1 X 1 ?a))
     (assert(IN 1 A 1 ?a)))

;; Input2 for X1 and A1
(defrule INPUT2-F1-X1-A1
     (IN 2 F 1 ?a)
    =>
     (assert(IN 2 X 1 ?a))
     (assert(IN 2 A 1 ?a)))

;;Carry input for X2 and A2
(defrule INPUT3-F1-X2-A2
     (IN 3 F 1 ?a)
    =>
    (assert(IN 1 A 2 ?a))
    (assert(IN 2 X 2 ?a)))

;OUTPUT
;;Output of X1 for X2 and A2
(defrule OUTPUTX1-X2-A2
     (OUT X 1 ?a)
    =>
     (assert(IN 1 X 2 ?a))
     (assert(IN 2 A 2 ?a)))

;;Output of A1 for O1
(defrule OUTPUTA1-O1
     (OUT A 1 ?a)
    =>(assert(IN 2 O 1 ?a)))

;;Output of A2 for O1
(defrule OUTPUTA2-O1
     (OUT A 2 ?a)
    =>
     (assert(IN 1 O 1 ?a)))

;OVERALL OUTPUT
;;Output of X2 as device output-1
```

```
(defrule OUTPUTX2-device1
    (OUT X 2 ?a)
    =>
    (assert(OUT 1 F 1 ?a)))

;;Output of O1 as device output-2
(defrule outO1-device2
    (OUT O 1 ?a)
    =>
    (assert(OUT 2 F 1 ?a)))


;Creation of rules for each gate
;;FOR XOR GATE
(defrule X
    (IN 1 X ?gate ?a1)
    (IN 2 X ?gate ?a2)
    =>
    (if
        (or
            (and
                (= ?a1 0) (= ?a2 0))
            (and
                (=?a1 1) (=?a2 1)))
        then
        (assert(OUT X ?gate 0))
        else
        (assert(OUT X ?gate 1))))

;;FOR AND GATE
(defrule A
    (IN 1 A ?gate ?a1)
    (IN 2 A ?gate ?a2)
    =>
    (if
        (and
            (=?a1 1) (=?a2 1))
        then
        (assert(OUT A ?gate 1))
        else
        (assert
            (OUT A ?gate 0))))

;;FOR OR GATE
(defrule O
    (IN 1 O ?gate ?a1)
    (IN 2 O ?gate ?a2)
    =>
    (if
        (and
            (=?a1 0) (=?a2 0))
        then
        (assert(OUT O ?gate 0))
        else
        (assert(OUT O ?gate 1))))


;OUTPUT RESULT
```

```
(defrule output
    (OUT 1 F 1 ?a1)
    (OUT 2 F 1 ?a2)
    =>
    (printout t crlf "OUTPUT RESULT:")
    (printout t crlf "Sum bit = "?a1"")
    (printout t crlf "C-Out = "?a2""))
(reset)
(run)
```

A2. *FAULT_DETECTION.CLP*

```
(deftemplate MAIN::problem_X2_gate
   "(Implied)"
   (multislot __data))

(deftemplate MAIN::problem_O1_gate
   "(Implied)"
   (multislot __data))

;Fault Diagnosis
(defrule fault_diagnosis
    (initial-fact)
    =>
    (printout t crlf "INPUT VALUES")
    (printout t crlf "IN_A? (0,1):")
    (assert(IN 1 F 1 =(read)))
    (printout t crlf "IN_B? (0,1):")
    (assert(IN 2 F 1 =(read)))
    (printout t crlf "Carry_in?(0,1):")
    (assert(IN 3 F 1 =(read)))

    (printout t crlf "DESIRED OUTPUTS")
    (printout t crlf "Sum_bit?(0,1):")
    (assert(OUT 1 F 1 =(read)))
    (printout t crlf "C-Out?(0,1):")
    (assert(OUT 2 F 1 =(read))))

(defrule fault
    (IN 1 F 1 ?IN_1)
    (IN 2 F 1 ?IN_2)
    (IN 3 F 1 ?Carry_in)
    (OUT 1 F 1 ?Sum_bit)
    (OUT 2 F 1 ?C_Out)
    =>

;;If there is any error in the sum bit,
;;the error should be detected starting with X2 and focus o the other outputs that were an input to X2
    (if
        (or
            (and
                (= ?IN_1 ?IN_2)
                (= ?Carry_in 1)
                (= ?Sum_bit 0))
            (and
                (= ?IN_1 ?IN_2)
                (= ?Carry_in 0)
                (= ?Sum_bit 1))
            (and
                (neq ?IN_1 ?IN_2)
                (= ?Carry_in 0)
                (= ?Sum_bit 0))
            (and
                (neq ?IN_1 ?IN_2)
                (= ?Carry_in 1)
                (= ?Sum_bit 1)))
        then
        (assert(problem_X2_gate)))


;;If there is any error in the Carry-out,
;;the error should be detected starting with O1 and focus o the other outputs that were an input to O1
    (if
```

```
        (or
            (and
                (= ?IN_1 ?IN_2 1)
                (= ?C-Out 0))
            (and
                (= ?IN_1 ?IN_2 0)
                (= ?C-Out 1))
            (and
                (neq ?IN_1 ?IN_2)
                (neq ?C-Out ?Carry_in)))
        then
        (assert(problem_O1_gate))))

(defrule print_problem_sum_bit
    (problem_X2_gate)
    =>
    (printout t crlf "There is a fault detection in X2 gate which could be as a result of the presence
of fault in X1 and/or X2." crlf))

(defrule print_problem_C_Out
    (problem_O1_gate)
    =>
    (printout t crlf "There is a fault detection in O1 gate which could be as a result of presence of
fault in A1, A2 and/or O1"))

(defrule problem_with_both_gates
    (problem_O1_gate)
    (problem_X2_gate)
    =>
    (printout t crlf "There are errors in both the Sum bit and the Carry-Out. Check below for more
information:"))

(reset)
(run)
```