**ASSIGNMENT**

Electronic Systems Assignment Coin Operated Water Dispenser

Showing all stages in the process, design and show the implementation of a digital system using D-type flip-flops and the necessary combinatorial logic to meet the following specification:

A coin operated water dispenser for car wash requires 40p for each operation. It can take in coins of value 20p and 10p up to the exact amount. No change is available. Require indicators for ***ready*** and ***next coin***.



For simplicity, base the programme on a single item. Your report should include the following designs and implementations:

1. Determine what inputs and outputs are required

**Inputs**

1. **C10 (10p Coin Input)**:
   * A toggle button or sensor indicating a 10p coin has been inserted.
   * **Type**: Binary (1 = coin inserted, 0 = no coin).
2. **C20 (20p Coin Input)**:
   * A toggle button or sensor indicating a 20p coin has been inserted.
   * **Type**: Binary (1 = coin inserted, 0 = no coin).

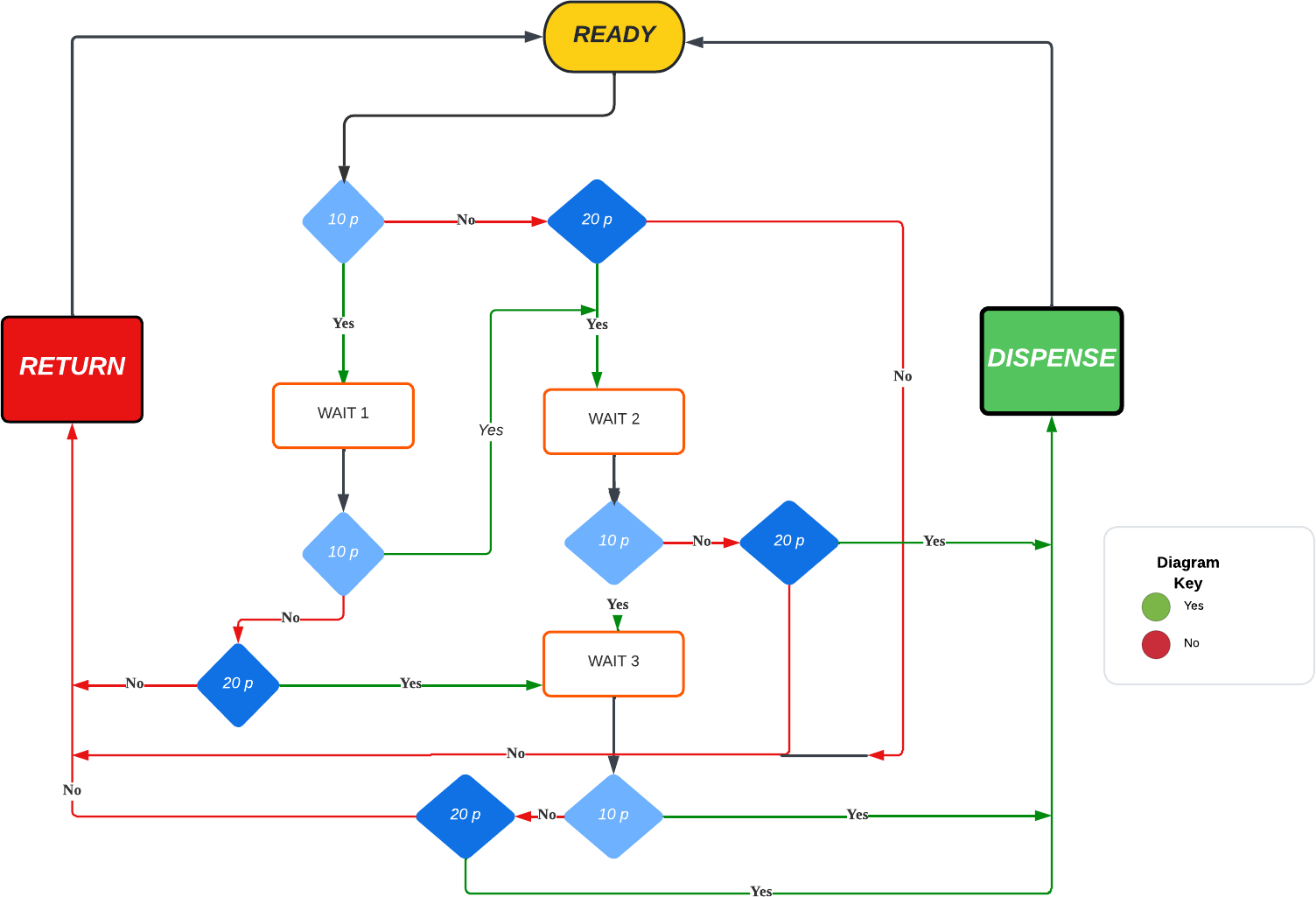
**Outputs**

1. **D (Dispense)**:
   * Indicates when the total amount reaches or exceeds 40p, and water should be dispensed.
   * **Type**: light.
2. **R (Return)**:
   * Indicates when coins should be returned (invalid input or timeout).
   * **Type**: light.
3. **Ready**:
   * Indicates the system is in the initial state (0p) and ready to accept coins.
   * **Type**: light.
4. **Next Coin**:
   * Indicates the system is waiting for the next coin (non-Ready state).
   * **Type**: light.

1. State-diagram / Flowchart

|  |
| --- |
|  |
| 01/S1 10/S2  11/S3 | |

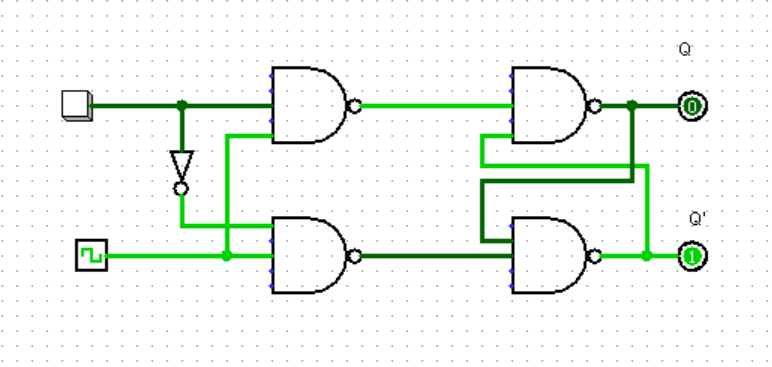
00/S00



1. D-type transition table

D flip-flop is a sequential logic circuit that stores a single bit of data with two inputs (D and CLK) and two outputs (Q and QG ). The D input of the flip-flop is connected to a digital signal source that can be either high or low, representing a binary 1 or 0, respectively. When the clock signal transitions from low to high, the state of the D input is captured and stored in the flip-flop. The stored value can then be read from the Q output. i.e., in the D flip-flops, the output can only be changed at the clock edge, and if the input changes at other times, the output will be unaffected.

Ref: [https://www.electronicsforu.com/technology-trends/learn-electronics/flip-flop-rs-jk-t-d](http://www.electronicsforu.com/technology-trends/learn-electronics/flip-flop-rs-jk-t-d)

****

|  |  |  |  |
| --- | --- | --- | --- |
| Current State (Q0 Q1) | Inputs (C10 C20) | Next State (Q0+ Q1+) | Outputs (D R) |
| 00 | 00 | 00 | 0 1 (Ready) |
| 00 | 01 | 01 | 0 0 |
| 00 | 10 | 10 | 0 0 |
| 00 | 11 | 00 | 0 1 (Return) |
| 01 | 00 | 00 | 0 1 (Return) |
| 01 | 01 | 10 | 0 0 |
| 01 | 10 | 11 | 0 0 |
| 01 | 11 | 00 | 0 1 (Return) |
| 10 | 00 | 00 | 0 1 (Return) |
| 10 | 01 | 11 | 0 0 |
| 10 | 10 | 00 | 1 0 (Dispense) |
| 10 | 11 | 00 | 0 1 (Return) |
| 11 | 00 | 00 | 0 1 (Return) |
| 11 | 01 | 00 | 1 0 (Dispense) |
| 11 | 10 | 00 | 1 0 (Dispense) |
| 11 | 11 | 00 | 1. 1 (Return) |

1. Next State Table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Q0 Q1 | C10 C20 | Q0+ Q1+ | D | R | Ready | Next Coin |
| 00 | 00 | 00 | 0 | 1 | 1 | 0 |
| 00 | 01 | 01 | 0 | 0 | 0 | 1 |
| 00 | 10 | 10 | 0 | 0 | 0 | 1 |
| 00 | 11 | 00 | 0 | 1 | 1 | 0 |
| 01 | 00 | 00 | 0 | 1 | 1 | 0 |
| 01 | 01 | 10 | 0 | 0 | 0 | 1 |
| 01 | 10 | 11 | 0 | 0 | 0 | 1 |
| 01 | 11 | 00 | 0 | 1 | 1 | 0 |
| 10 | 00 | 00 | 0 | 1 | 1 | 0 |
| 10 | 01 | 11 | 0 | 0 | 0 | 1 |
| 10 | 10 | 00 | 1 | 0 | 0 | 0 |
| 10 | 11 | 00 | 0 | 1 | 1 | 0 |
| 11 | 00 | 00 | 0 | 1 | 1 | 0 |
| 11 | 01 | 00 | 1 | 0 | 0 | 0 |
| 11 | 10 | 00 | 1 | 0 | 0 | 0 |
| 11 | 11 | 00 | 0 | 1 | 1 | 0 |

1. Karnaugh-Maps

##### **5.1 K-map for**Q0+

| **Q0 Q1\C10 C20** | **00** | **01** | **11** | **10** |
| --- | --- | --- | --- | --- |
| **00** | 0 | 0 | 0 | 0 |
| **01** | 0 | 1 | 0 | 1 |
| **11** | 0 | 0 | 0 | 0 |
| **10** | 0 | 1 | 0 | 1 |

**Simplified**:

*Q*0+=*Q*0′⋅*C*10′⋅*C*20+*Q*0′⋅*C*10⋅*C*20′+*Q*0⋅*Q*1′⋅*C*10′⋅*C*20

##### **5.2 K-map for**Q1+

| **Q0 Q1\C10 C20** | **00** | **01** | **11** | **10** |
| --- | --- | --- | --- | --- |
| **00** | 0 | 1 | 0 | 1 |
| **01** | 0 | 0 | 0 | 0 |
| **11** | 0 | 0 | 0 | 0 |
| **10** | 0 | 0 | 0 | 0 |

**Simplified**:

*Q*1+=*Q*1′⋅*C*10⋅*C*20′+*Q*0′⋅*Q*1⋅*C*10′⋅*C*20

##### **5.3 K-map for**D**(Dispense)**

| **Q0 Q1\C10 C20** | **00** | **01** | **11** | **10** |
| --- | --- | --- | --- | --- |
| **00** | 0 | 0 | 0 | 0 |
| **01** | 0 | 0 | 0 | 0 |
| **11** | 0 | 1 | 0 | 1 |
| **10** | 0 | 0 | 0 | 1 |

**Simplified**:

*D*=*Q*0⋅*C*10′⋅*C*20+*Q*0⋅*Q*1⋅*C*10⋅*C*20′

##### **5.4 K-map for***R***(Return)**

| **Q0 Q1\C10 C20** | **00** | **01** | **11** | **10** |
| --- | --- | --- | --- | --- |
| **00** | 1 | 0 | 1 | 0 |
| **01** | 1 | 0 | 1 | 0 |
| **11** | 1 | 0 | 1 | 0 |
| **10** | 1 | 0 | 1 | 0 |

**Simplified**:

*R*=*C*10⋅*C*20+*C*10′⋅*C*20′⋅(*Q*0+*Q*1)

**K-map for Ready**

**Variables**: Q0,Q1

| **Q0\Q1** | **0** | **1** |
| --- | --- | --- |
| **0** | 1 | 0 |
| **1** | 0 | 0 |

**Simplified Equation**:

Ready=Q0′⋅Q1′

**K-map for Next Coin (Q0+Q1)**

**Variables**: Q0,Q1

| **Q0\Q1** | **0** | **1** |
| --- | --- | --- |
| **0** | 0 | 1 |
| **1** | 1 | 1 |

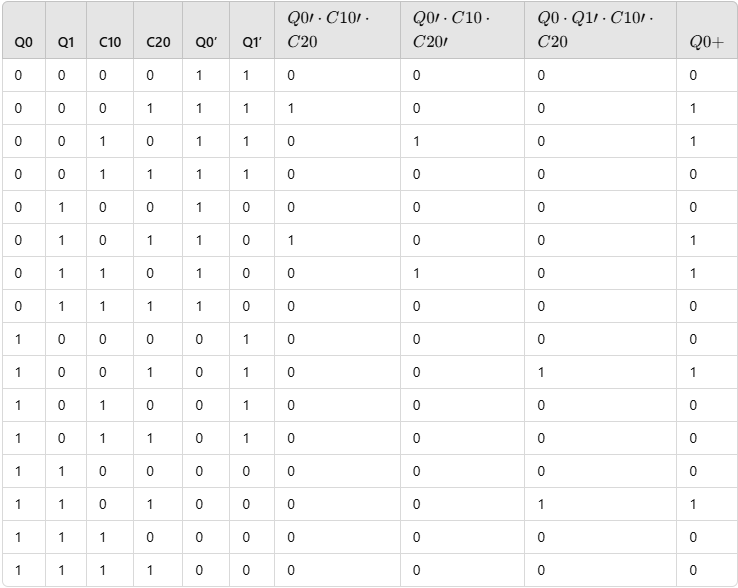
**Simplified Equation**:

Next Coin=Q0+Q1

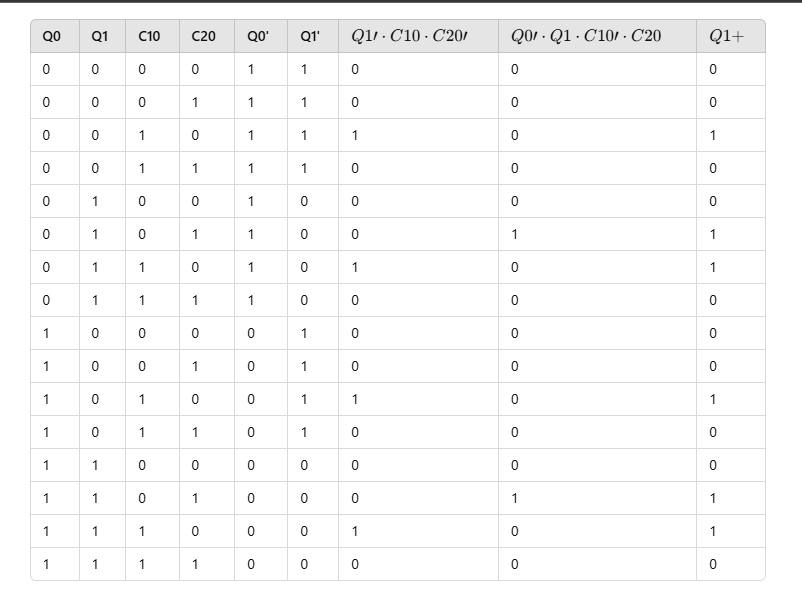
### Logical Expressions Verification

**Next State**:

* *Q0+=Q0′⋅C10′⋅C20+Q0′⋅C10⋅C20′+Q0⋅Q1′⋅C10′⋅C20*



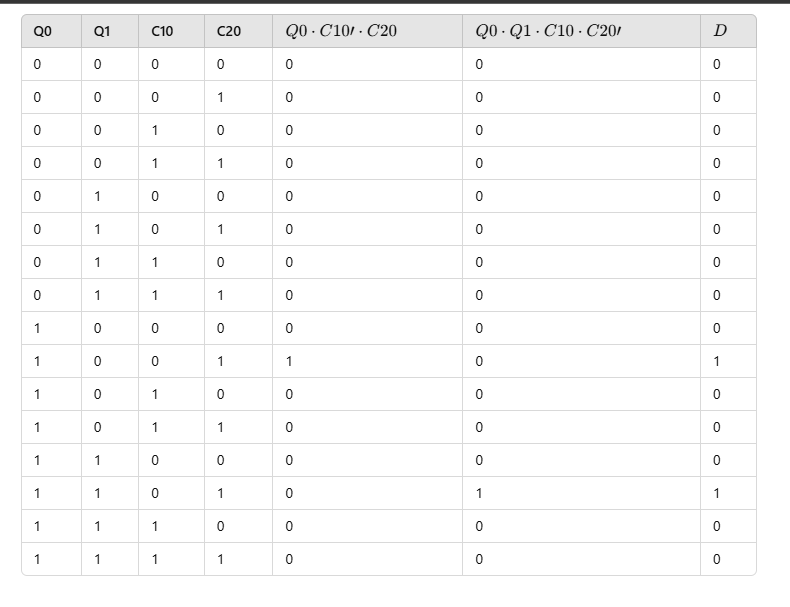
* Q1+=Q1′⋅C10⋅C20′+Q0′⋅Q1⋅C10′⋅C20



**Outputs**:

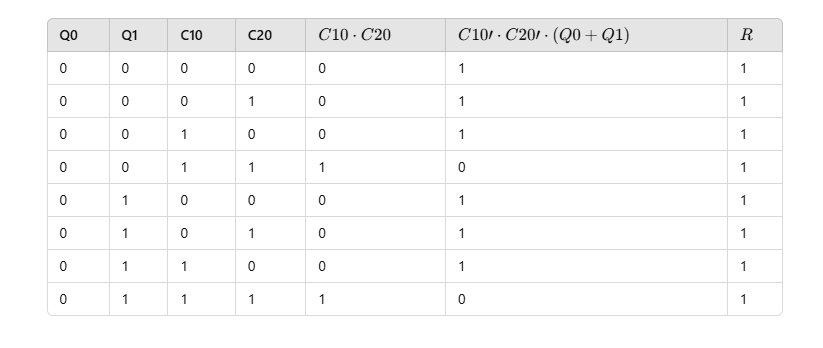
* + D=Q0⋅C10′⋅C20+Q0⋅Q1⋅C10⋅C20′

**Verification**:



**Simplification**: Correct.

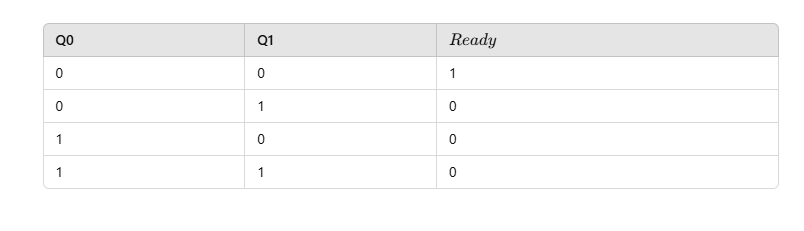
* + R=C10⋅C20+C10′⋅C20′⋅(Q0+Q1)



**Simplification**: Correct.

* + Ready=Q0′⋅Q1′

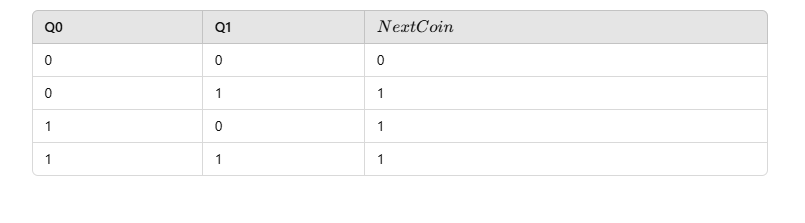
**Verification**:



**Simplification**: Correct.

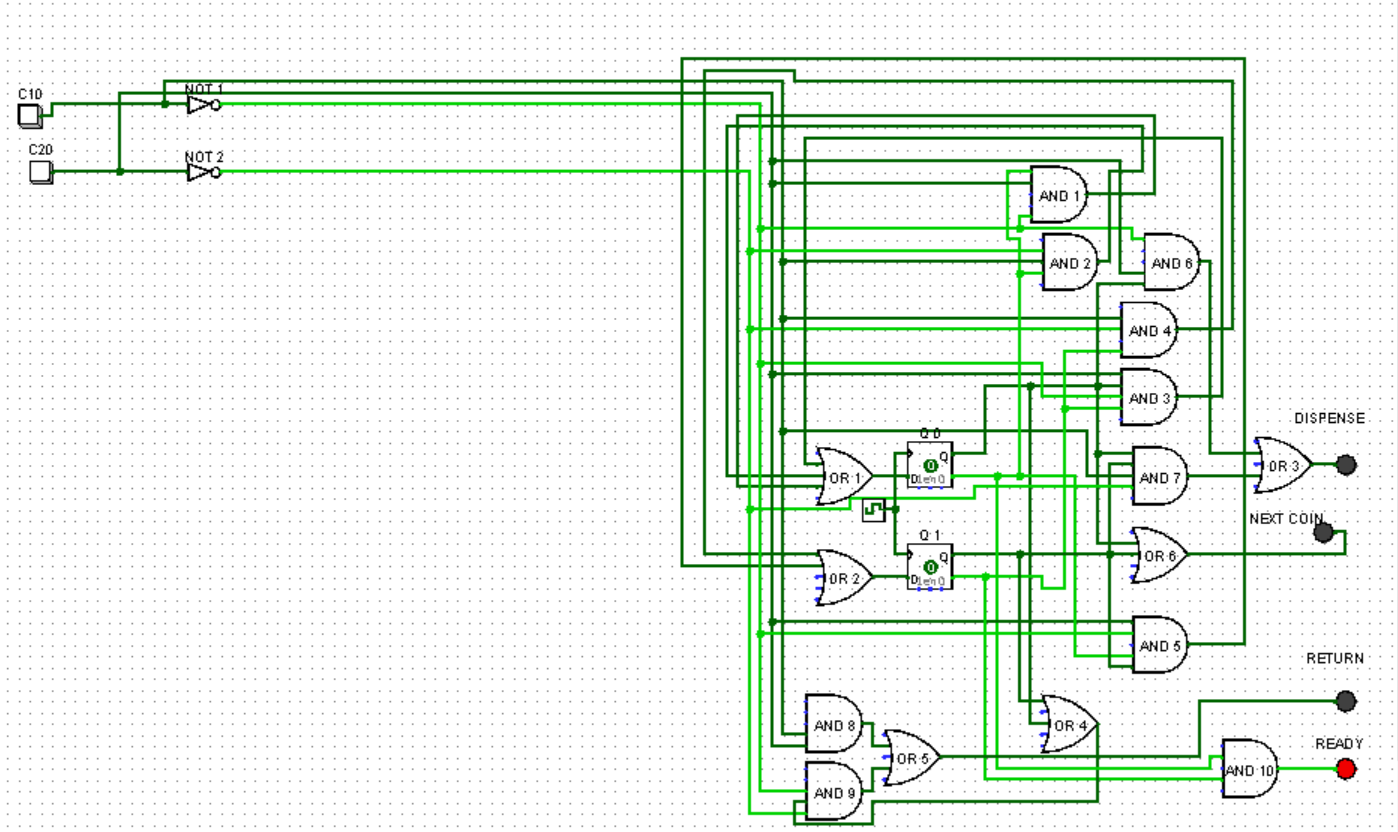
* + Next Coin=Q0+Q1

**Verification**:



**Simplification**: Correct.

1. Circuitry

****

1. Implement in Altera Hardware Description Language (AHDL)

CODE:

SUBDESIGN Coin\_Water\_Dispenser

(

C10, C20 : INPUT;

CLK, RESET : INPUT;

Dispense, Return, Ready, NextCoin : OUTPUT;

)

VARIABLE

Q0, Q1 : DFF;

BEGIN

-- State Transitions

CASE (Q0 & Q1) IS

"00": -- S0: Ready State

IF (C10 = 1) THEN Q0.d = 1; END IF;

IF (C20 = 1) THEN Q1.d = 1; END IF;

"01": -- S1: 10p Inserted

IF (C10 = 1) THEN Q0.d = 0; Q1.d = 1; END IF; -- Move to S2 (20p inserted)

IF (C20 = 1) THEN Q0.d = 1; END IF; -- Move to S3 (30p inserted)

"10": -- S2: 20p Inserted

IF (C10 = 1) THEN Q0.d = 1; END IF; -- Move to S3 (30p inserted)

IF (C20 = 1) THEN Q0.d = 0; Q1.d = 0; END IF; -- Reset after dispensing

"11": -- S3: 30p Inserted

IF (C10 = 1) THEN Q0.d = 0; Q1.d = 0; END IF; -- Dispense and reset

IF (C20 = 1) THEN Q0.d = 0; Q1.d = 0; END IF; -- Dispense and reset

END CASE;

-- Outputs

Dispense = (Q0 & ~Q1 & C10 & ~C20) OR (Q0 & Q1 & ~C10 & C20) OR (Q0 & Q1 & C10 & ~C20);

Return = (C10 & C20) OR (~C10 & ~C20 & (Q0 OR Q1));

Ready = ~Q0 & ~Q1;

NextCoin = Q0 OR Q1;

END;