Laboratorium 3
Sterownik windy

Łukasz Kwinta, Kacper Kozubowski, Ida Ciepiela

maj 2024

Spis treści

1 Cel zadania

Proszę zaproponować, zbudować i przetestować układ sterujący windą w przykładowym trzykondygnacyjnym budynku.

Winda posiada:

- wskaźnik ruchu windy
- wskaźnik kierunku ruchu windy
- trzy czujniki otwarcia drzwi, po jednym na każdej kondygnacji
- trzy przyciski przywołania windy, po jednym na każdej kondygnacji
- trzy przyciski wyboru piętra w kabinie windy.

Winda powinna posiadać stale aktualizowany wskaźnik aktualnego piętra.

Rzeczy niedopowiedziane w treści zadania, proszę ustalić, doprecyzować i opisać samodzielnie.

2 Rozwiązania

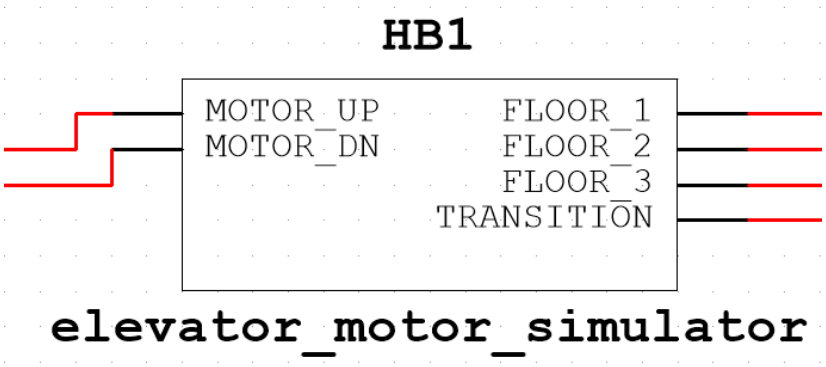
Postanowiliśmy rozbić problem na wiele mniejszych problemów i rozwiązać je osobno aby na końcu połączyć je w jeden działający system. Poniżej zamieszamy schemat blokowy przedstawiający poszczególne systemy.

2.1 Symulator silnika windy

Aby zasymulować ruch windy wraz z czasem przemieszczania się między piętrami, zdecydowaliśmy się zaimplementować układ bazujący na liczniku i demultiplexerze.

2.1.1 Black box

Poniżej zamieszczamy schemat wyjść i wejść oraz opis logiki układu.



Rysunek 2.1: Black box układu symulującego silnik windy

2.1.2 Wejścia

- `MOTOR_UP` - sygnał wejściowy nakazujący poruszać się windzie w górę
- `MOTOR_DN` - sygnał wejściowy nakazujący poruszać się windzie w dół

2.1.3 Wyjścia

- `FLOOR_1` - sygnał wyjściowy informujący o tym, że winda znajduje się na 1 piętrze, aktywny w stanie wysokim
- `FLOOR_2` - sygnał wyjściowy informujący o tym, że winda znajduje się na 2 piętrze, aktywny w stanie wysokim
- `FLOOR_3` - sygnał wyjściowy informujący o tym, że winda znajduje się na 3 piętrze, aktywny w stanie wysokim
- `TRANSITION` - sygnał wyjściowy informujący o tym że winda jest obecnie w ruchu, aktywny w stanie wysokim

2.1.4 Realizacja układu

Do realizacji układu wykorzystaliśmy 4 bitowy licznik z biblioteki komponentów programu Multisim, układ `74LS193N`. Poniżej zamieszczamy tabelkę przedstawiającą działanie układu:

`CLR`	`~LOAD`	`Up`	`Down`	Mode
H	X	X	X	Reset(Async.)
L	L	X	X	Preset(Async.)
L	H	H	H	No Change
L	H	↑	H	Count Up
L	H	H	↑	Count Down

Tabela 2.1: Źródło: `https://www.multisim.com/help/components/binary-counters/`

Streszczenie

- H - stan wysoki na wejściu
- L - stan niski na wejściu
- X - dowolny stan na wejściu
- ↑ - narastające zbocze sygnału

Stworzyliśmy układ kombinacyjny, który mapuje wyjście zegara, na adres demultiplexera, który z kolei przekazuje jedynkę logiczną na odpowiednie wyjście. Przyjęliśmy, że:

- 0000 - winda jest na 1 piętrze

- 1000 - winda jest na 2 piętrze
- 1111 - winda jest na 3 piętrze
- każdy inny - winda porusza się między piętrami

Jako demultiplexer wykorzystaliśmy układ `U7A 4555BD_5V`. Jest to demultiplexer 1: 4, z 2 bitami adresowymi.

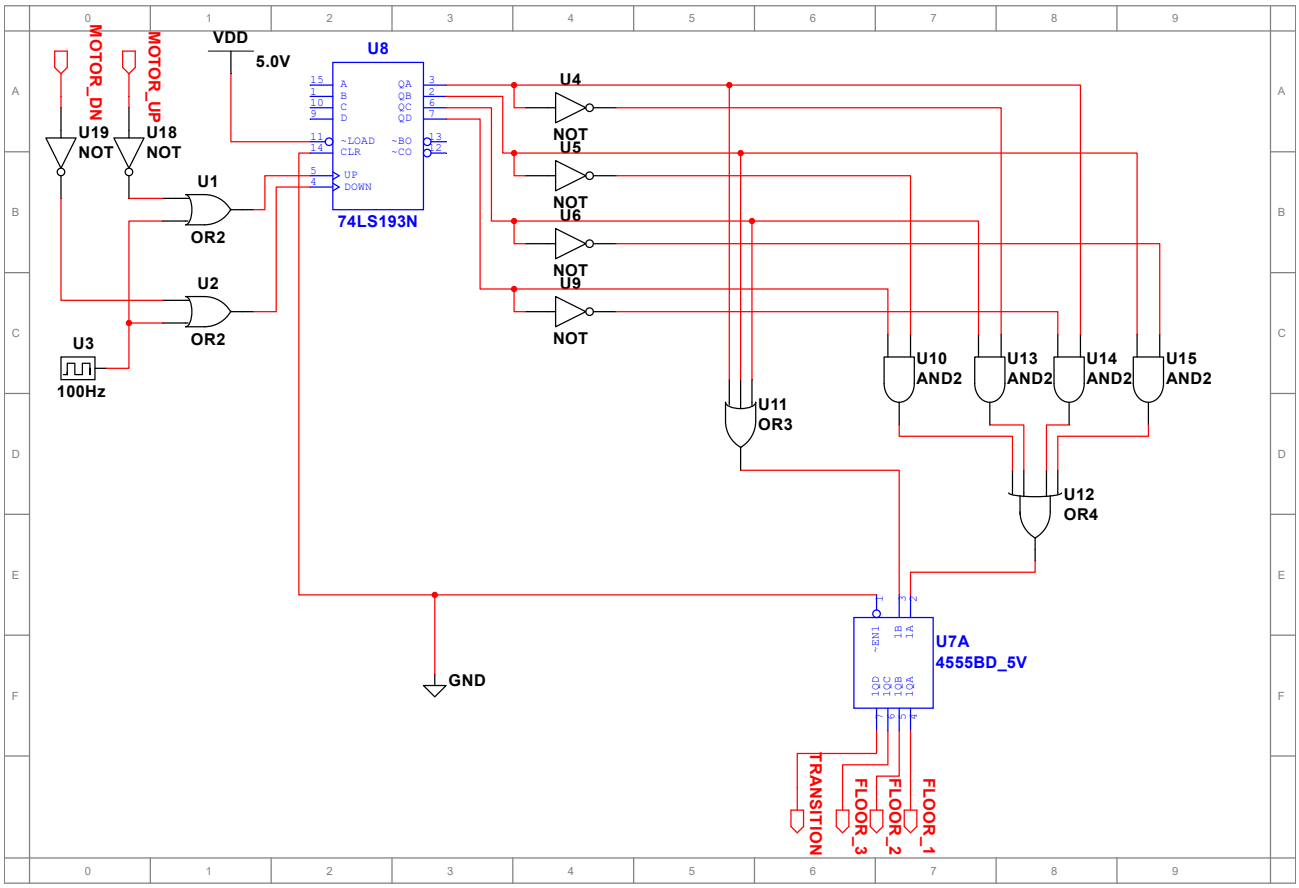
Do wyprowadzenia formuł wykorzystaliśmy skrypt w języku Python, który generuje tabelę prawdy oraz minimalizuje formuły logiczne. Poniżej zamieszczamy kod programu oraz wynik jego działania:

```python
import logicmin
elevator_motor_mux_tt = logicmin.TT(4, 2)

for i in range(16):
    permutation = bin(i).removeprefix("0b").rjust(4, '0')

    a1 = '1'
    b1 = '1'

    if i == 0:
        a1 = '0'
        b1 = '0'
    elif i == 8:
        a1 = '1'
        b1 = '0'
    elif i == 15:
        a1 = '0'
        b1 = '1'

    elevator_motor_mux_tt.add(permutation, [a1, b1])

print("--------------------------------elevator_motor_tt")
sols = elevator_motor_mux_tt.solve()
print(sols.printN(xnames=['QD', 'QC', 'QB', 'QA'],ynames=['1A', '1B']))
```

Wynikiem działania skryptu są zminimalizowane formuły logiczne:

```
--------------------------------elevator_motor_tt
1B <= QA + QB + QC
1A <= QD'.QA + QC'.QB + QC.QA' + QD.QB'
```

Po zaimplementowaniu układu w programie Multisim, uzyskaliśmy następujący schemat:



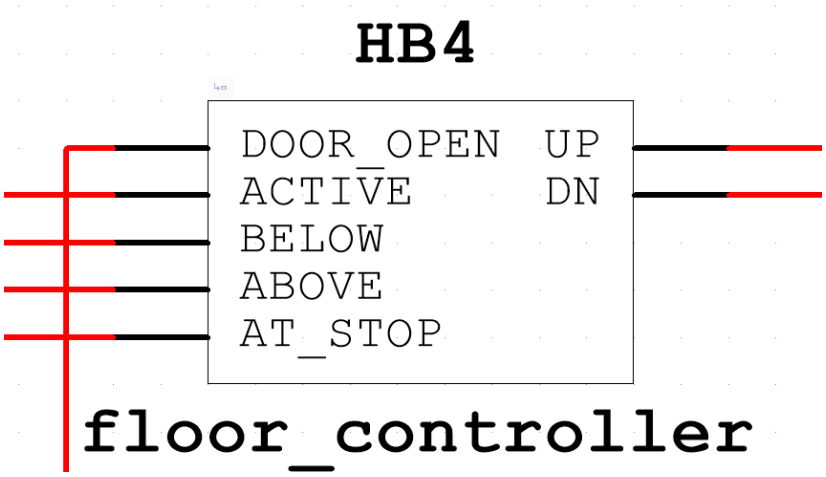
Rysunek 2.2: Schemat układu symulującego silnik windy

Bramki logiczne `NOT` i `OR` na wejściu układu służą do przekazywania stanu wysokiego lub zegara na wejście licznika. Zgodnie z tabelką działania układu zapewniają one, że w momencie gdy chcemy liczyć do góry, układ przekazuje stan wysoki na wejście licznika `DOWN` oraz sygnał zegara na wejście licznika `UP`.

2.2 Kontroler ruchu windy

Prosty układ kombinacyjny przekazujący silnikowi sygnał o ruchu w danym kierunku gdy wszystkie warunki na ruszenie windą zostaną spełnione.

2.2.1 Black box



Rysunek 2.3: Black box układu inicjującego ruch windy

2.2.2 Wejścia

- `DOOR_OPEN` - sygnał wejściowy informujący o stanie otwarcia drzwi
- `ACTIVE` - sygnał wejściowy informujący, że widna otrzymała wezwanie
- `BELOW` - sygnał wejściowy informujący, że piętro docelowe znajduje się poniżej obencej pozycji windy
- `ABOVE` - sygnał wejściowy informujący, że piętro docelowe znajduje się powyżej obencej pozycji windy
- `AT_STOP` - sygnał wejściowy informując, że widna znajduje się na piętrze na, które została wezwana

2.2.3 Wyjścia

- `UP` - sygnał wyjściowy informujący o gotowośći do ruchu w górę
- `DN` - sygnał wyjściowy informujący o gotowośći do ruchu w dół

2.2.4 Realizacja układu

W realizacji układu wykorzystaliśmy dwa przerzutniki SR, które są ustawiane w momencie gdy wszystkie warunki do ruchu w danym kierunku są spełnione i resetowane gdy spełnione są wszystkie warunki konieczne do zatrzymania windy.

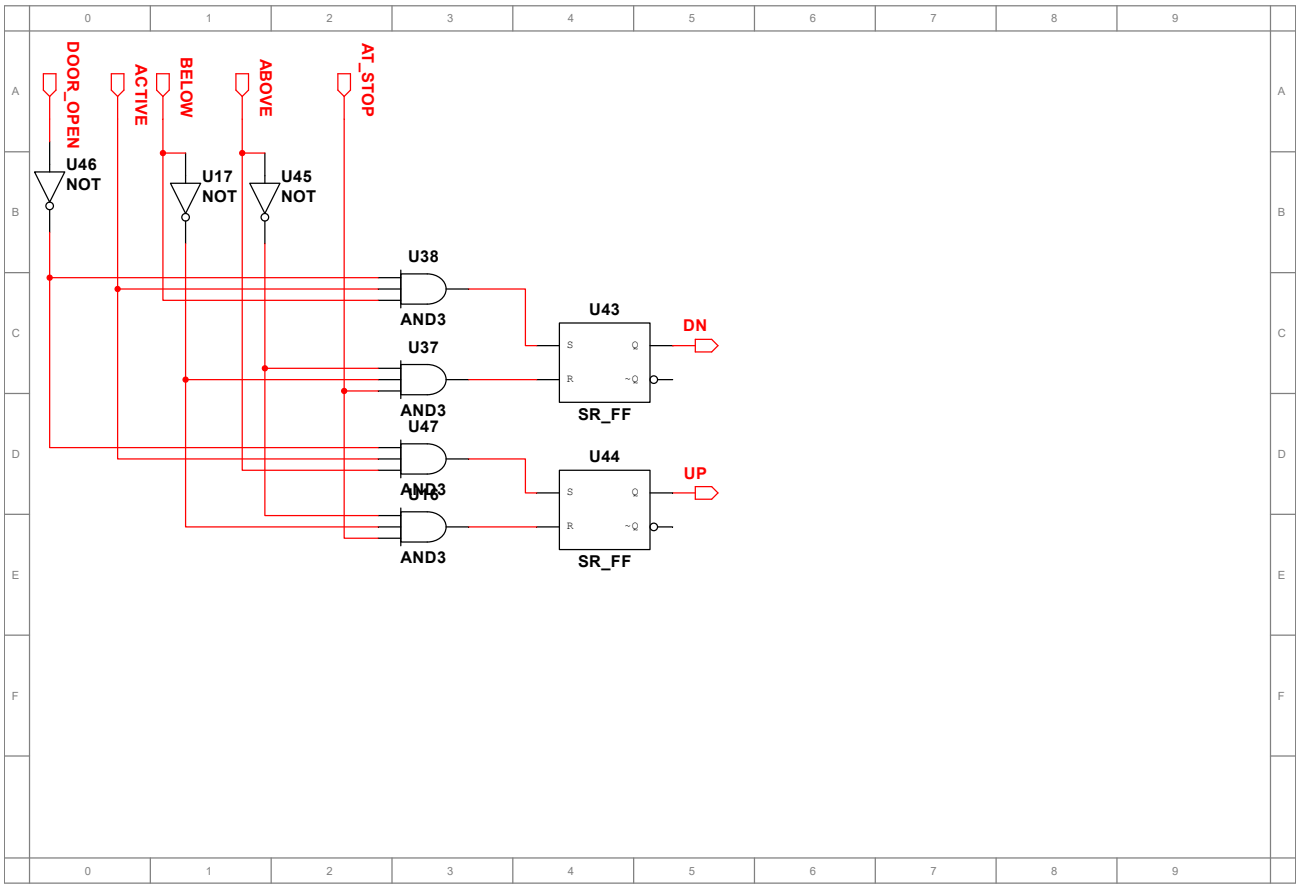
Poniżej przedstawiony został kod programu w języku Python, który wykorzystaliśmy do wyprowadzenia formuł logicznych.

```python
import logicmin
floor_controller_tt = logicmin.TT(5, 4)

for i in range(32):
    permutation = bin(i).removeprefix("0b").rjust(5, '0')

    variables = {
        'DOOR_OPEN': int(permutation[0]),
        'ACTIVE': int(permutation[1]),
        'BELOW': int(permutation[2]),
        'ABOVE': int(permutation[3]),
        'AT_STOP': int(permutation[4])
    }

    set_up = '0'
    reset_up = '0'
    set_dn = '0'
    reset_dn = '0'

    if not variables['ABOVE'] and not variables['BELOW'] and variables['AT_STOP']:
        set_up = '0'
        reset_up = '1'
        set_dn = '0'
        reset_dn = '1'

    elif variables['ACTIVE'] and variables['BELOW'] and not variables['DOOR_OPEN']:
        set_up = '0'
        reset_up = '0'
        set_dn = '1'
        reset_dn = '0'

    elif variables['ACTIVE'] and variables['ABOVE'] and not variables['DOOR_OPEN']:
        set_up = '1'
        reset_up = '0'
        set_dn = '0'
        reset_dn = '0'

    floor_controller_tt.add(permutation, [set_up, reset_up, set_dn, reset_dn])

print("------------------------------floor_controller_tt")
sols = floor_controller_tt.solve()
print(sols.printN(xnames=['DOOR_OPEN', 'ACTIVE', 'BELOW', 'ABOVE', 'AT_STOP'],ynames=['SET_UP', 'RESET_UP','SET_DN', 'RESET_DN']))
```

Wynikiem działania skryptu są zminimalizowane formuły logiczne:

```
--------------------------------floor_controller_tt
RESET_DN <= BELOW'.ABOVE'.AT_STOP
SET_DN <= DOOR_OPEN'.ACTIVE.BELOW
RESET_UP <= BELOW'.ABOVE'.AT_STOP
SET_UP <= DOOR_OPEN'.ACTIVE.BELOW'.ABOVE
```

Na podstawie otrzymanych formuł zaimplementowaliśmy w programie Multisim, niżej przedstawiony schemat.

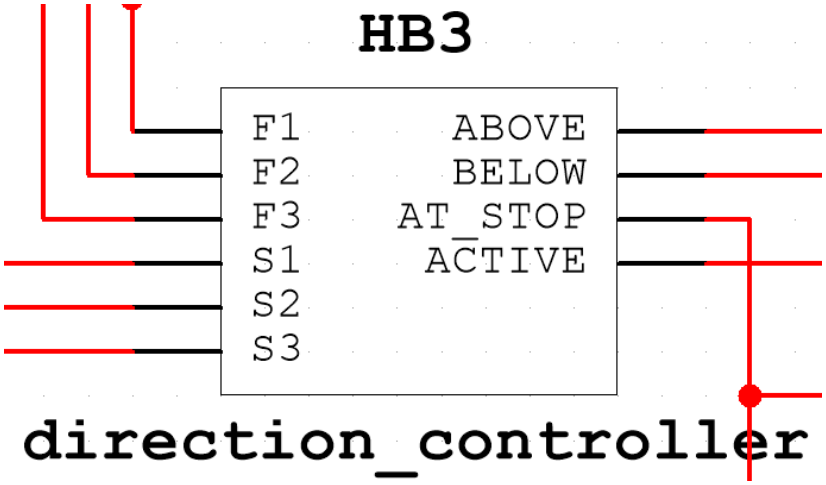


Rysunek 2.4: Schemat układu inicjującego ruch windy

2.3 Kontroler kierunku ruchu

Układ, który na podstawie otrzymanego wezwania rozpoznaje względną pozycję piętra docelowego w odniesieniu do obecnego położenia windy.

2.3.1 Black box



Rysunek 2.5: Black box układu przetwarzającego wezwanie

2.3.2 Wejścia

- `F1` - sygnał wejściowy informujący o tym, że winda znajduje się na 1. piętrze
- `F2` - sygnał wejściowy informujący o tym, że winda znajduje się na 2. piętrze
- `F3` - sygnał wejściowy informujący o tym, że winda znajduje się na 3. piętrze
- `S1` - sygnał wejściowy informujący o wezwaniu windy na 1. piętro
- `S2` - sygnał wejściowy informujący o wezwaniu windy na 2. piętro
- `S3` - sygnał wejściowy informujący o wezwaniu windy na 3. piętro

2.3.3 Wyjścia

- `ABOVE` - sygnał wyjściowy informujący, że piętro docelowe znajduje się powyżej obencej pozycji windy
- `BELOW` - sygnał wyjściowy informujący, że piętro docelowe znajduje się poniżej obencej pozycji windy
- `AT_STOP` - sygnał wyjściowy informujący, że widna znajduje się na piętrze docelowym
- `ACTIVE` - sygnał wyjściowy przekazujący dalej informację o otrzymaniu wezwania

2.3.4 Realizacja układu

Poniżej przedstawiony został kod programu w języku Python, który wykorzystaliśmy do wyprowadzenia formuł logicznych.

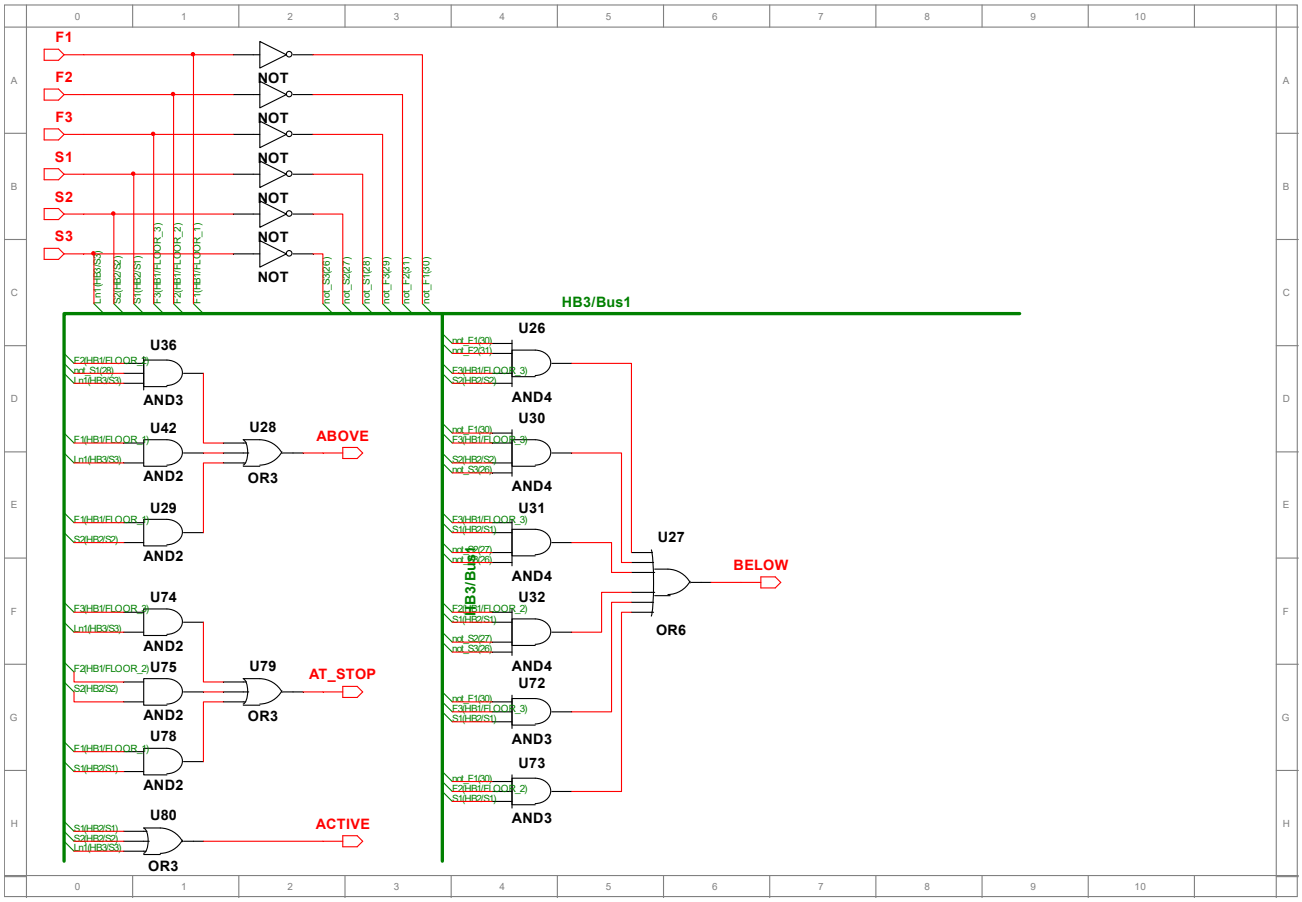
```
import logicmin
direction_controller_tt = logicmin.TT(6, 4)

for i in range(64):
    permutation = bin(i).removeprefix("0b").rjust(6, '0')

    variables = {
        'F1': int(permutation[0]),
        'F2': int(permutation[1]),
        'F3': int(permutation[2]),
        'S1': int(permutation[3]),
        'S2': int(permutation[4]),
        'S3': int(permutation[5])
    }

    above = '0'
    below = '0'
    at_stop = '0'
    active = '0'

    if variables['F1'] and (variables['S2'] or variables['S3']):
        above = '1'

    elif variables['F2'] and variables['S1']:
        below = '1'

    elif variables['F2'] and variables['S3']:
        above = '1'

    elif variables['F3']and (variables['S1'] or variables['S2']):
        below = '1'

    if (variables['F1'] and variables['S1']) or (variables['F2'] and variables['S2']) or (variables['F3'] and variables['S3']):
        at_stop = '1'

    if variables['S1'] or variables['S2'] or variables['S3']:
        active = '1'

    direction_controller_tt.add(permutation, [above, below, at_stop, active])

print("------------------------------direction_controller_tt")
sols = direction_controller_tt.solve()
print(sols.printN(xnames=['F1', 'F2', 'F3', 'S1', 'S2', 'S3'],ynames=['ABOVE','BELOW', 'AT_STOP', 'ACTIVE']))
```

Wynikiem działania skryptu są zminimalizowane formuły logiczne:

```
--------------------------------direction_controller_tt
ACTIVE <= S3 + S2 + S1
AT_STOP <= F3.S3 + F2.S2 + F1.S1
BELOW <= F1'.F2'.F3.S2 + F1'.F3.S2.S3' + F3.S1.S2'.S3' + F2.S1.S2'.S3' +
+ F1'.F3.S1 + F1'.F2.S1
```

```
ABOVE <= F2.S1'.S3 + F1.S3 + F1.S2
```

Na podstawie otrzymanych formuł zaimplementowaliśmy w programie Multisim, niżej przedstawiony schemat.

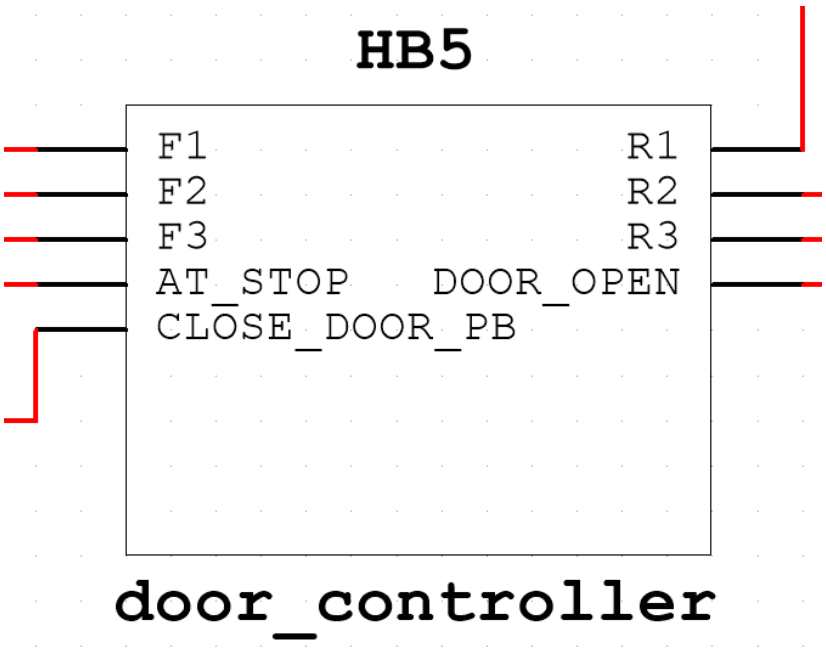


Rysunek 2.6: Schemat układu przetwarzającego wezwanie

2.4 Kontroler drzwi windy

Układ zajmujący się obsługą drzwi po znalezieniu się windy na piętrze docelowym. Przetwarza on również informację o wykonaniu obsługiwanego wezwania i wysyła sygnał do jego resetu.

2.4.1 Black box



Rysunek 2.7: Black box układu kontrolującego obsługę drzwi

2.4.2 Wejścia

- `F1` - sygnał wejściowy informujący o tym, że winda znajduje się na 1. piętrze
- `F2` - sygnał wejściowy informujący o tym, że winda znajduje się na 2. piętrze
- `F3` - sygnał wejściowy informujący o tym, że winda znajduje się na 3. piętrze
- `AT_STOP` - sygnał wejściowy informujący, że widna znajduje się na piętrze docelowym
- `CLOSE_DOOR_PB` - sygnał wejściowy nakazujący zamknięcie drzwi

2.4.3 Wyjścia

- `R1` - sygnał wyjściowy, informujący o wykonaniu wezwania na 1. piętro
- `R2` - sygnał wyjściowy, informujący o wykonaniu wezwania na 2. piętro
- `R3` - sygnał wyjściowy, informujący o wykonaniu wezwania na 3. piętro
- `DOOR_OPEN` - sygnał wyjściowy, informujący o otwarciu drzwi

2.4.4 Realizacja układu

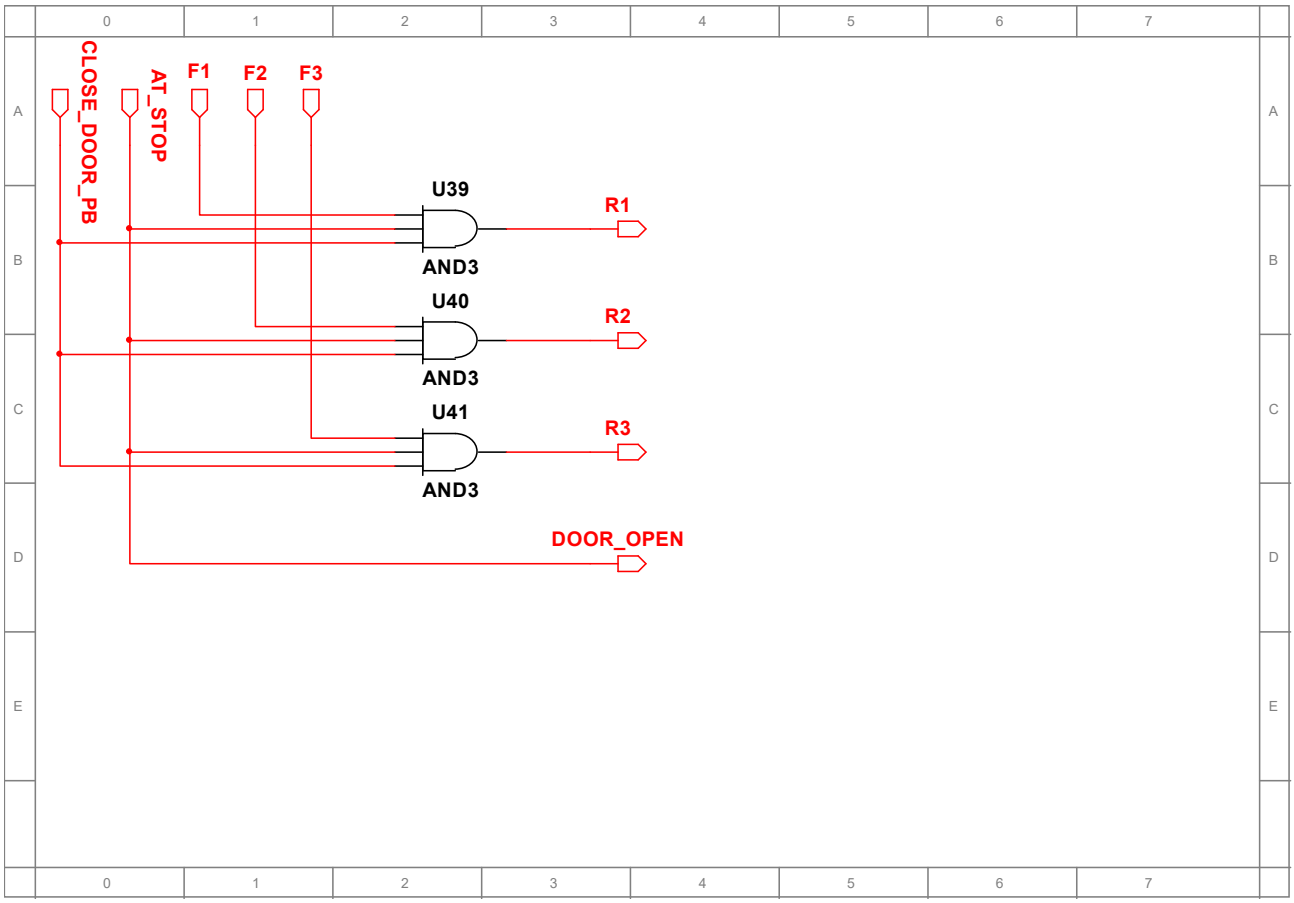
Poniżej przedstawiony został kod programu w języku Python, który wykorzystaliśmy do wyprowadzenia formuł logicznych.

```python
import logicmin
door_controller_tt = logicmin.TT(5, 4)

for i in range(32):
    permutation = bin(i).removeprefix("0b").rjust(5, '0')

    variables = {
        'CLOSE_DOOR_PB': int(permutation[0]),
        'AT_STOP': int(permutation[1]),
        'F1': int(permutation[2]),
        'F2': int(permutation[3]),
        'F3': int(permutation[4])
    }

    r1 = '0'
    r2 = '0'
    r3 = '0'
    door_open = '0'

    if variables['AT_STOP']: door_open = '1'
    if variables['AT_STOP'] and variables['CLOSE_DOOR_PB']:
        if variables['F1']: r1 = '1'
        if variables['F2']: r2 = '1'
        if variables['F3']: r3 = '1'

    door_controller_tt.add(permutation, [r1, r2, r3, door_open])

print("--------------------------------door_controller_tt")
sols = door_controller_tt.solve()
print(sols.printN(xnames=['CLOSE_DOOR_PB', 'AT_STOP', 'F1', 'F2', 'F3'],ynames=['R1','R2', 'R3', 'DOOR_OPEN']))
```

Wynikiem działania skryptu są zminimalizowane formuły logiczne:

```
--------------------------------door_controller_tt
DOOR_OPEN <= AT_STOP
R3 <= CLOSE_DOOR_PB.AT_STOP.F3
R2 <= CLOSE_DOOR_PB.AT_STOP.F2
R1 <= CLOSE_DOOR_PB.AT_STOP.F1
```

Na podstawie otrzymanych formuł zaimplementowaliśmy w programie Multisim, niżej przedstawiony schemat.



Rysunek 2.8: Schemat układu kontrolującego obsługę drzwi

3 Testy

4 Zastosowania

5 Wnioski