DD2452 – Homework 5
Mert Alp Taytak – taytak@kth.se

Report

Tables

Please note that I did not include redundant else statements where no change occurs.

CPU State

```
if cpu.state = 0 then cpu.state := 1
else if cpu.state = 1 then cpu.state := 2
else if cpu.state = 2 then cpu.state := 3
else if cpu.state = 3 then cpu.state := 4
else if cpu.state = 4 then cpu.state := 5
else if cpu.state = 5 then cpu.state := 0
```

CPU Memory

```
if cpu.state = 5 then
     cpu.memory[cpu.oldest] := cpu.temperature
```

CPU Oldest

```
if cpu.state = 5 then
     if cpu.oldest = 2 then cpu.oldest := 0
     else cpu.oldest := cpu.oldest + 1
```

CPU Temperature

```
if cpu.state = 2 then
     cpu.temperature := sensor.temperature
else if cpu.state = 4 then
     cpu.temperature := cpu.temperature + 273
```

Thermometer Sleep

```
if cpu.state = 1 then sensor.sleep := 0
else if cpu.state = 3 then sensor.sleep := 1
```

Thermometer Temperature

```
sensor.temperature := random from -100 to 100
```

Model

```
MODULE CPU(sensor)
VAR
    temperature: word[32];
    memory: array 0 .. 2 of word[32];
    oldest: {0, 1, 2};
    state: {0, 1, 2, 3, 4, 5};
ASSIGN
    init(state) := 0;
    next(state) :=
        case
            state = 0 : 1;
            state = 1 : 2;
            state = 2 : 3;
            state = 3 : 4;
            state = 4 : 5;
            state = 5 : 0;
        esac;
    next(sensor.sleep) :=
        case
            state = 1 : 0b1_0;
            state = 3 : 0b1_1;
            TRUE : sensor.sleep;
        esac;
    next(temperature) :=
        case
            state = 2 : sensor.temperature;
            state = 4 : temperature + uwconst(273, 32);
            TRUE : temperature;
        esac;
    next(oldest) :=
        case
            state = 5 & oldest = 2 : 0;
            state = 5 & oldest != 2 : oldest + 1;
            TRUE: oldest;
        esac;
    next(memory[0]) :=
        case
            state = 5 & oldest = 0 : temperature;
            TRUE : memory[0];
        esac;
    next(memory[1]) :=
        case
            state = 5 & oldest = 1 : temperature;
            TRUE : memory[1];
        esac;
    next(memory[2]) :=
        case
            state = 5 & oldest = 2 : temperature;
            TRUE : memory[2];
        esac;
```

```
MODULE THERMOMETER
VAR
    sleep: word[1];
    temperature: word[32];
ASSIGN
    -- I could not find a way to assign a random value to a word
    next(temperature) := uwconst(0, 32);
```

Results

Specifications Used

```
-- (1) Just after the ISR has terminated, the sensor is sleeping
CTLSPEC AX(cpu.state = 5) -> sensor.sleep = 0b1_1;
-- (2) When the sensor is not sleeping, then the ISR is executed
CTLSPEC AG(sensor.sleep != 0b1_1 -> cpu.state != 0);
-- (3) If the ISR is not executed, then the sensor is sleeping
CTLSPEC AG(cpu.state != 0 -> sensor.sleep = 0b1_1);
-- (4) When the ISR is not executed, then the value of memory
--      at the previous value of oldest is equal to cpu.temperature
CTLSPEC AG(cpu.state != 0 -> cpu.memory[cpu.oldest] = cpu.temperature);
-- (5) Just after the ISR has terminated, then the value of memory
--      at the previous value of oldest is equal to cpu.temperature
CTLSPEC AX(cpu.state = 5) -> cpu.memory[cpu.oldest] = cpu.temperature;
-- (6) When the ISR is not executed, then cpu.temperature >= 173
CTLSPEC AG(cpu.state = 0 -> cpu.temperature >= uwconst(173, 32));
-- (7) When ISR has terminated, then cpu.temperature >= 173
CTLSPEC AX(cpu.state = 5) -> cpu.temperature >= uwconst(173, 32);
-- (8) Oldest is always in the set {0,1,2}
CTLSPEC AG(cpu.oldest in 0 .. 2);
```

Specification Statuses

True: 1, 5, 7, 8

False: 2, 3, 4, 6

Timings

Specification **1** takes **37 milliseconds** to prove.

Specification **2** takes **59 milliseconds** to prove.

Specification **3** takes **70 milliseconds** to prove.

Specification **4** takes **740 milliseconds** to prove.

Specification **5** takes **30 milliseconds** to prove.

Specification **6** takes **67 milliseconds** to prove.

Specification **7** takes **34 milliseconds** to prove.

Specification **8** takes **29 milliseconds** to prove.