EE-334
Digital System Design

Discussion for Lab 4
Arbiter FSM in VHDL

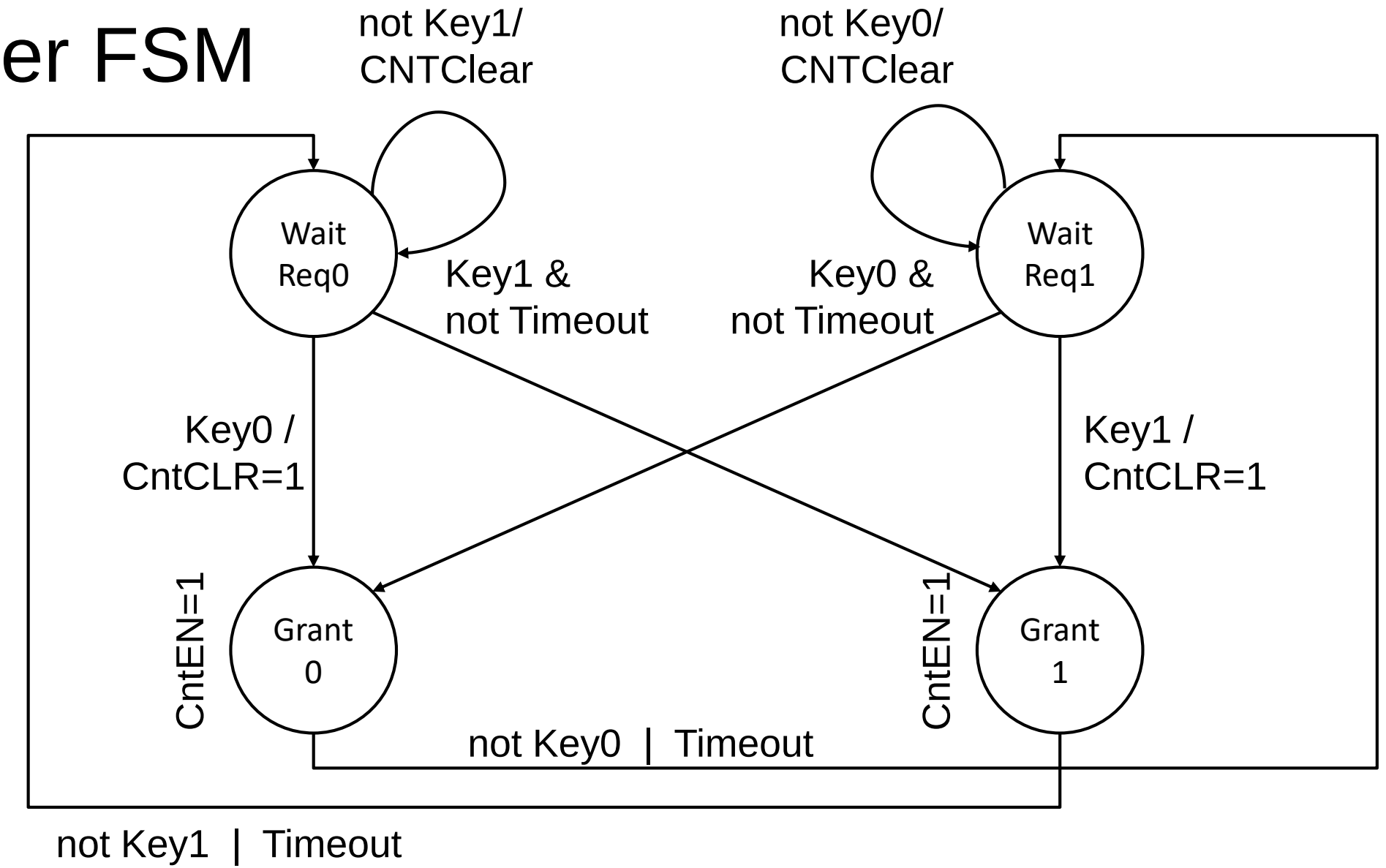
Andreas Burg, A. Kristensen

Arbiter Description and States

- Two nodes users can request access with `Key0,Key1 ∈ {‘0’, ‘1’}`
- Only one user is granted access at a time
- `GLED0,GLED1 ∈ {‘0’, ‘1’}` indicate access. `RLED0,RLED1 ∈ {‘0’, ‘1’}` indicate access is requested, but currently denied (other user has access)
- When a user is granted access it can keep the resource for at least 2 seconds
- After 2 seconds, the other user can take over (steal the access) with a request

- States:
  - WAIT_REQ1: User 0 has priority, but both users can request access
  - WAIT_REQ2: User 1 has priority, but both users can request access
  - GRANT_SS1: User 0 has the lock for less than 2 seconds and maintains access
  - GRANT_SS2: User 1 has the lock for less than 2 seconds and maintains access

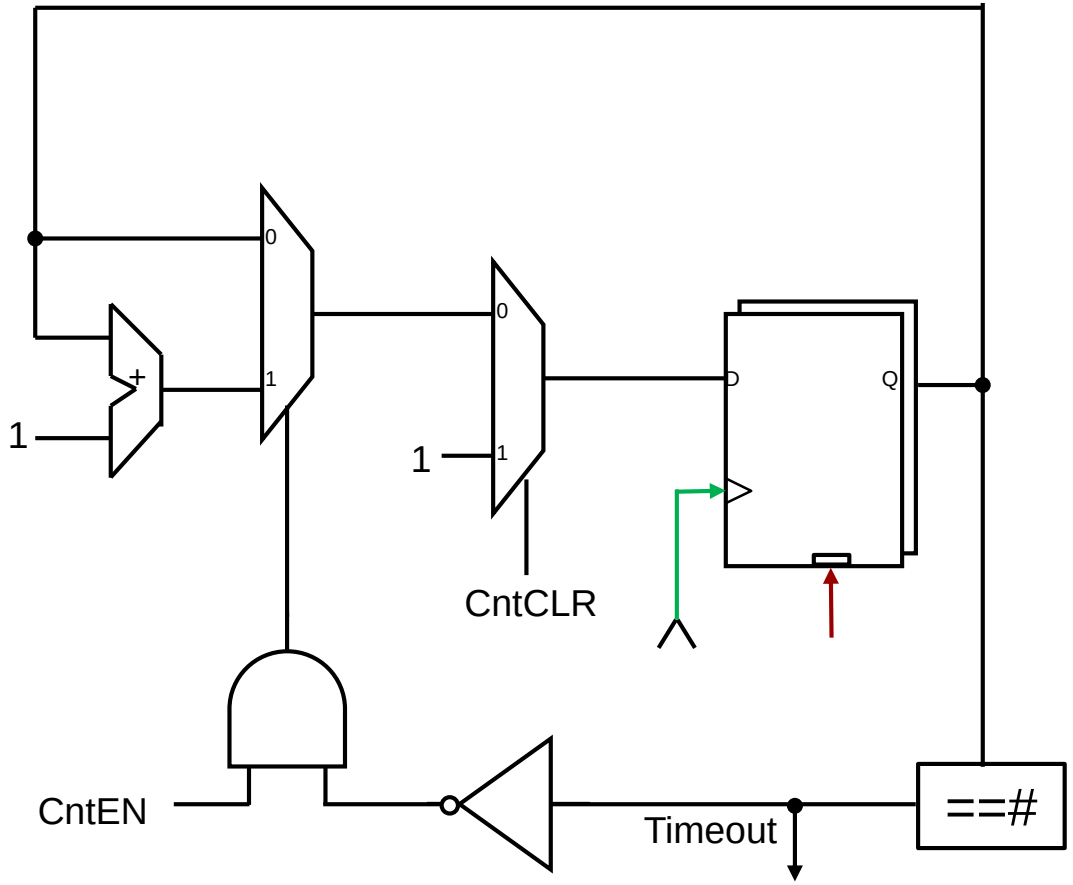
Arbiter FSM



DEFAULT: remain in same state
CntEN=0, CntCLR=0

Timeout Counter

- Start counting when cleared, but stop when reaching timeout



VHDL

- Registers: FSM and Counter in one process (two OK as well)
- Combinational logic for the counter

```
--============================================================================
--  STATE AND DATA REGISTERS
--! Processes for updating the state and data registers
--============================================================================
p_REG: PROCESS (CLKxCI, RSTxRI) IS
BEGIN
    IF RSTxRI = '1' THEN
        STATExDP <= WAIT_REQ1;
        CountxDP <= (OTHERS => '0');
    ELSIF CLKxCI'event AND CLKxCI = '1' THEN
        STATExDP <= STATExDN;
        CountxDP <= CountxDN;
    END IF;
END PROCESS p_REG;
```

```
--============================================================================
-- COUNTER LOGIC
--============================================================================'

-- Enable counter when access has been granted for a guaranteed 2 seconds
CountENxS      <= '1' WHEN (STATExDP = GRANT_SS0 OR STATExDP = GRANT_SS1) ELSE '0';
CountTimeoutxS <= '1' WHEN CountxDP = CountTimeout ELSE '0';

CountxDN <= (OTHERS => '0') WHEN CountCLRxS = '1' ELSE
        CountxDP + 1 WHEN CountENxS = '1' AND CountTimeoutxS = '0' ELSE
        CountxDP;
```

VHDL

- FSM combinational logic with (single process possible as well)
  - process for next state assignments and counter control
  - concurrent statements for controlling output LEDs (see next slide)

```
--=============================================================================
--  FSM
--! Process defining the FSM for the arbiter
--=============================================================================
p_FSMComb: PROCESS (KEY0xDI, KEY1xDI, STATExDP, CountTimeoutxS) IS
BEGIN

    -- Defaults registers
    STATExDN <= STATExDP;

    -- Defaults combinational
    CountCLRxS <= '0';

    -- FSM implementation of arbiter. Note that if a request is removed or we
    -- switch to serving another user, the counter is cleared to guarantee
    -- 2 seconds of access
    CASE STATExDP IS

        -- WAIT_REQ1: Prioritize requests from user 1, otherwise serve user 2
        WHEN WAIT_REQ0 =>

            CountCLRxS <= NOT Key1xDI OR Key0xDI;

            IF Key0xDI = '1' THEN
                STATExDN <= GRANT_SS0;
            ELSIF Key1xDI = '1' AND CountTimeoutxS = '0' THEN
                STATExDN <= GRANT_SS1;
            END IF;

        -- WAIT_REQ2: Prioritize requests from user 2, otherwise serve user 1
        WHEN WAIT_REQ1 =>

            CountCLRxS <= NOT Key0xDI OR Key1xDI;

            IF Key1xDI = '1' THEN
                STATExDN <= GRANT_SS1;
            ELSIF Key0xDI = '1' AND CountTimeoutxS = '0' THEN
                STATExDN <= GRANT_SS0;
            END IF;

        -- GRANT_SS1: Currently serving user 1 until request is removed or
        -- user 2 makes a request after timeout (in WAIT_REQ2)
        WHEN GRANT_SS0 =>

            CountCLRxS <= NOT Key0xDI;

            IF Key0xDI = '0' OR CountTimeoutxS = '1' THEN
                STATExDN <= WAIT_REQ1;
            END IF;

        -- GRANT_SS2: Currently serving user 2 until request is removed or
        -- user 1 makes a request after timeout (in WAIT_REQ1)
        WHEN GRANT_SS1 =>

            CountCLRxS <= NOT Key1xDI;

            IF Key1xDI = '0' OR CountTimeoutxS = '1' THEN
                STATExDN <= WAIT_REQ0;
            END IF;

        WHEN OTHERS => NULL;
    END CASE;

END PROCESS p_FSMComb;
```

VHDL

- FSM combinational logic with (single process possible as well)
  - process for next state assignments and counter control
  - concurrent statements for controlling output LEDs

```
-------------------------------------------------------------------------------
p_FSMComb: PROCESS (KEY0xDI, KEY1xDI, STATExDP, CountTimeoutxS) IS
BEGIN

    -- Defaults registers
    STATExDN <= STATExDP;

    -- Defaults combinational
    CountCLRxS <= '0';

    -- FSM implementation of arbiter. Note that if a request is removed or we
    -- switch to serving another user, the counter is cleared to guarantee
    -- 2 seconds of access
    CASE STATExDP IS

        -- WAIT_REQ1: Prioritize requests from user 1, otherwise serve user 2
        WHEN WAIT_REQ0 =>

            CountCLRxS <= NOT Key1xDI OR Key0xDI;

            IF Key0xDI = '1' THEN
                STATExDN <= GRANT_SS0;
            ELSIF Key1xDI = '1' AND CountTimeoutxS = '0' THEN
                STATExDN <= GRANT_SS1;
            END IF;

        -- WAIT_REQ2: Prioritize requests from user 2, otherwise serve user 1
        WHEN WAIT_REQ1 =>

            CountCLRxS <= NOT Key0xDI OR Key1xDI;

            IF Key1xDI = '1' THEN
                STATExDN <= GRANT_SS1;
            ELSIF Key0xDI = '1' AND CountTimeoutxS = '0' THEN
                STATExDN <= GRANT_SS0;
            END IF;
```

```
        -- GRANT_SS1: Currently serving user 1 until request is removed or
        -- user 2 makes a request after timeout (in WAIT_REQ2)
        WHEN GRANT_SS0 =>

            CountCLRxS <= NOT Key0xDI;

            IF Key0xDI = '0' OR CountTimeoutxS = '1' THEN
                STATExDN <= WAIT_REQ1;
            END IF;

        -- GRANT_SS2: Currently serving user 2 until request is removed or
        -- user 1 makes a request after timeout (in WAIT_REQ1)
        WHEN GRANT_SS1 =>

            CountCLRxS <= NOT Key1xDI;

            IF Key1xDI = '0' OR CountTimeoutxS = '1' THEN
                STATExDN <= WAIT_REQ0;
            END IF;
```

```
--============================================================================
-- OUTPUT LOGIC
--============================================================================'

-- GRANT_SS0 and WAIT_REQ1 represent states where user 1 can be served
GLED0xDO <= '1' WHEN Key0xDI = '1' AND     (STATExDP = GRANT_SS0 OR STATExDP = WAIT_REQ1) ELSE '0';
RLED0xDO <= '1' WHEN Key0xDI = '1' AND NOT (STATExDP = GRANT_SS0 OR STATExDP = WAIT_REQ1) ELSE '0';

-- GRANT_SS1 and WAIT_REQ0 represent states where user 2 can be served
GLED1xDO <= '1' WHEN Key1xDI = '1' AND     (STATExDP = GRANT_SS1 OR STATExDP = WAIT_REQ0) ELSE '0';
RLED1xDO <= '1' WHEN Key1xDI = '1' AND NOT (STATExDP = GRANT_SS1 OR STATExDP = WAIT_REQ0) ELSE '0';
```