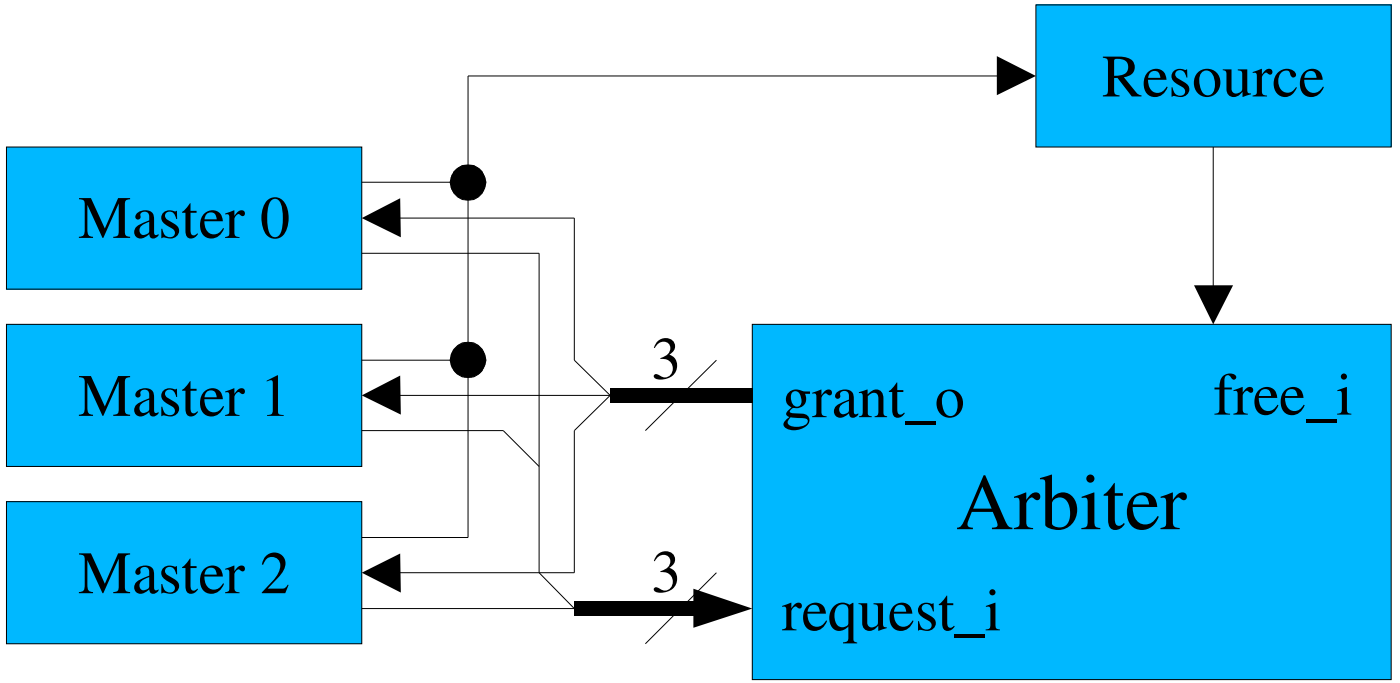
# Aufgabe 3: Implementierung eines Arbiters

- Ein Arbiter hat die Aufgabe, eine Resource verschiedenen Mastern zur Verfügung zu stellen



# Arbiter (2)

```
entity arbiter is
 generic (n : natural := 3);
 port    (request_i : in
             std_logic_vector(n-1 downto 0);
          free_i    : in  std_logic;
          grant_o   : out
             std_logic_vector(n-1 downto 0));
end arbiter;
```

- Zustände (mindestens):
  - IDLE –– warten auf einen Request
  - BUSY –– warten auf free_i

# Arbiter (3)

- Master *i*:
  - Anfordern der Resource mit request(*i*) = ’1’
  - Request bleibt stabil, bis grant(*i*) = ’1’
  - Wenn grant(*i*) = ’1’ geht request(*i*) im nächsten Takt wieder auf ’0’ zurück
- Resource:
  - Bekommt eine Aufgabe
  - Setzt free für einen Takt, wenn Aufgabe erledigt ist
  - Keine parallele Verarbeitung von Aufgaben!