

Tutorium zu Computer-Engineering im WS19

Termin 9

Jakob Otto

HAW Hamburg

12. Dezember 2019

Ablauf

- Aufgabenzettel Nr 5
- Metastabile Zustände
- Asynchrone Kommunikation
 - ▶ 4 Phasen Handshake
- Statemachines in VHDL
- VHDL-teil 4-Phasen Handshake



Aufgabenzettel 5

Metastabile Zustände

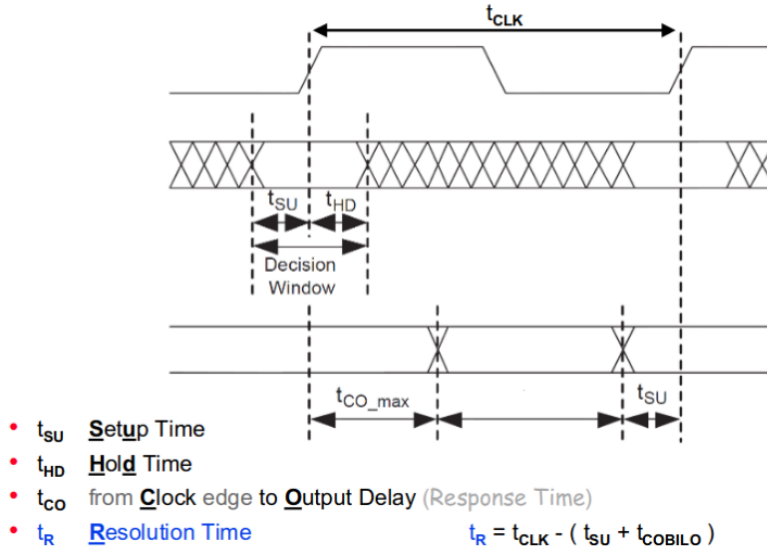
Metastabile Zustände

Zustand des Signals ist nicht klar definiert
Irgendwas zwischen '1' und '0'

Was ist das?

- Signale sind nicht unmittelbar stabil
 - ▶ Brauchen kurz bis gewünschter Zustand erreicht ist
- Set-up hold-time greift hier (\rightarrow DT)
- Signal kann also zur „falschen“ Zeit abgetastet werden
 - \rightarrow Metastabile Zustände sind die Folge

Metastabile Zustände



Asynchrone Kommunikation

- Endpunkte nutzen oft unterschiedlichen Takt
- Deshalb Kommunikation zwischen Endpunkten asynchron
- Es muss also synchronisiert werden

Kurzschlüsse und Korrupte Daten können entstehen!

4 Phasen Handshake

- Protokoll zum synchronen Übermitteln von Daten
- Bidirektional über einen 'x'-bit Datenbus
- Verhindert:
 - ▶ Metastabile Zustände
 - ▶ Kurzschlüsse

4 Phasen Handshake

Welche Leitungen werden benötigt?

- RD/nWR - signal zum lesen/schreiben signalisieren
- REQ - Request vom Master zu Slave
- ACK/nRDY - acknowledge vom Slave zum Master
- Data - 'x' bit Datenbus

Lesender Zugriff

① Master initiiert $RD/nWR = '1'$
Datenbus auf high-impedance
REQ auf '1'

② Slave reagiert
legt geforderte Daten auf
Datenbus
acknowledged ($ACK = '1'$)

③ Master quittiert Empfang
 $REQ = '0'$

④ Slave nimmt Daten vom Bus
Datenbus auf High-impedance

Lesender Zugriff

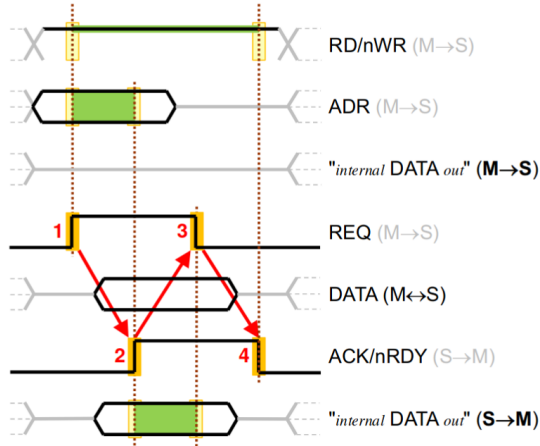


Abbildung: 4-Phasen-Handshake lesender Zugriff

```
uint16_t  rxdat;
// wait for the fpga to be ready
while(READY == 0);

// setup the connection to the FPGA
OUTPUT_DISABLE;
SET_READ;

// handle transaction by 4 phase handshake
REQ_ENABLE;
while(ACKNOWLEDGE == 0);
rxdat = (GPIOE->IDR & 0x0000FFFF);
REQ_DISABLE;
while(ACKNOWLEDGE != 0);

return rxdat;
```

Schreibender Zugriff

① Master initiiert RD/nWR = '0'
legt Daten auf Bus
REQ auf '1'

② Slave quittiert
ACK = '1'

③ Master setzt Bus auf
High-impedance
REQ = '0'

④ slave quittiert quittung
ACK = '0'

Schreibender Zugriff

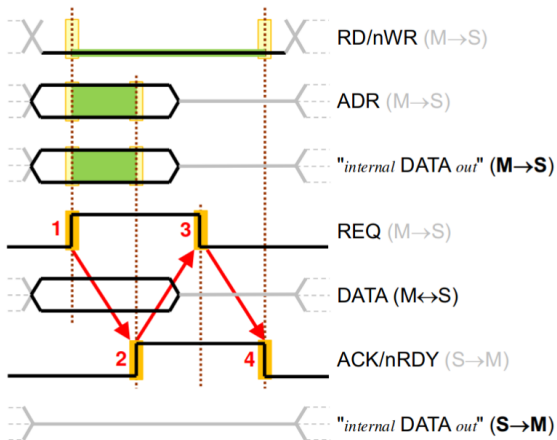


Abbildung: 4-Phasen-Handshake lesender Zugriff

```
// setup connection to the FPGA
SET_WRITE;

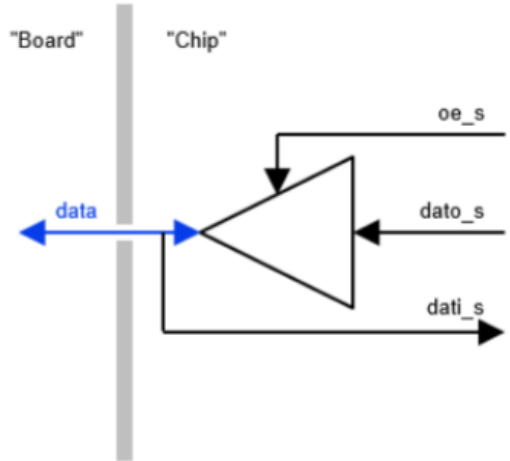
OUTPUT_ENABLE;
// set data
GPIOE->ODR = txdat;

// handle transaction by 4 phase handshake
REQ_ENABLE;
while(ACKNOWLEDGE == 0);
REQ_DISABLE;
while(ACKNOWLEDGE != 0);
```

Tristate buffer

Tri-State Treiber für Datenbus

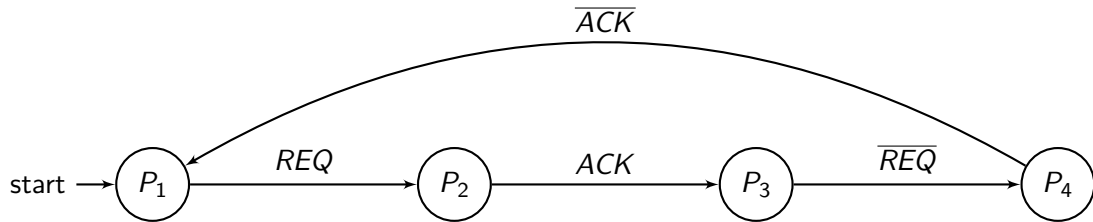
```
tristate:
process (oe_s, dato_s) is
begin
    if oe_s = '1' then
        data <= dato_s;
    else
        data <= (others=>'Z');
    end if;
end process tristate;
--
dati_s <= data;
```



4 Phasen handshake Controller

- STM-32 Seite (Master) nun klar
 - ▶ Wie also nun FPGA Seite (Slave)?
- 4 Phasen Handshake gut durch Statemachine darstellbar
- 4 Zustände
- Lesen/Schreiben nicht großartig unterschiedlich
 - ▶ Nur unterschiede bei Reaktion auf Request

4 Phasen Statemachine



Statemachine Aufbau I

Statemachines

- Sollten eigentlich noch aus DT bekannt sein
- hier trotzdem eine kurze Auffrischung!

```
-- 2 bit - 4 states
```

```
signal state_ns : std_logic_vector(1 downto 0);
```

```
signal state_cs : std_logic_vector(1 downto 0) := (others=>'0');
```

Statemachine Aufbau II

```
state_v := state_cs;  
case state_cs is  
  when "00" =>  
    -- state 1  
    state_v := "01";  
  when "01" =>  
    -- state 2  
    state_v := "10";  
  when "10" =>  
    -- state 3  
    state_v := "00";  
  when others =>  
    -- default case  
end case;  
state_ns <= state_v;
```

Die einzelnen States schaut ihr euch so an :)

State 1

State 1

- Read?
 - ▶ $oe \leftarrow 1$
 - ▶ $dato \leftarrow fx$
- Write?
 - ▶ $oe \leftarrow 0$
 - ▶ $req \leftarrow 1$
 - ▶ $rdy \leftarrow 0$
- $state \leftarrow 1$

State 2

State 2

- $\text{ack} \leftarrow 1$
- $\text{req} = 0?$
 - ▶ $\text{oe} \leftarrow 0$
 - ▶ $\text{state} \leftarrow 2$

State 3

State 3

- $\text{ack} \leftarrow 0$
- $\text{oe} \leftarrow 0$
- $\text{state} \leftarrow 0$

Aufbau Der Schaltung

