

# Implementierung eines Softcore-Mikroprozessor für FPGAs

Gruppe 2

06. April 2022

# Microcontroller

## Mikroprozessor Architektur

Betrachtete  
Befehle

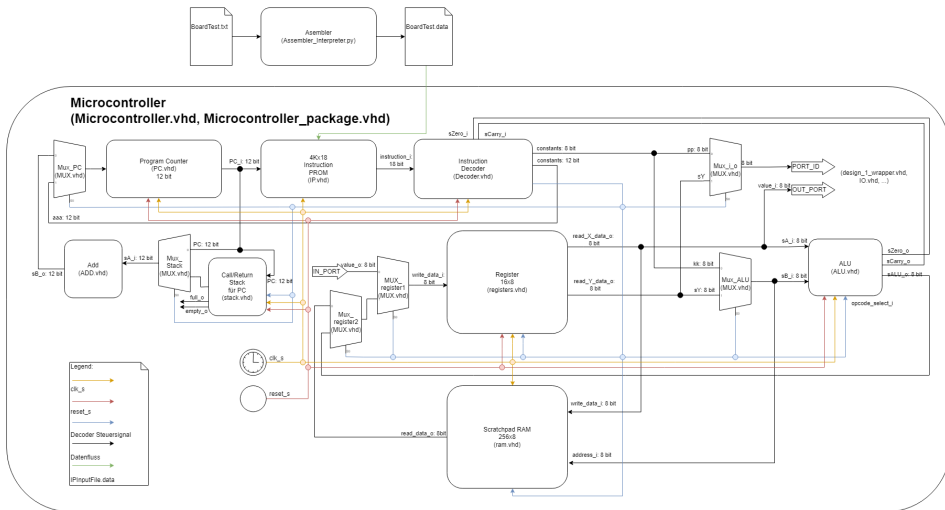
ADD sX, sY

CALL aaa

STORE sX, ss

OUTPUT sX, pp

enable\_i



# Betrachtete Befehle

Mikroprozessor  
Architektur

Betrachtete  
Befehle

ADD sX, sY

CALL aaa

STORE sX, ss

OUTPUT sX, pp

enable.i

- ADD-Befehl (z.B. ADD 17, 18)
- CALL-Befehl (z.B. CALL 23)
- STORE-Befehl (z.B. STORE 17, 42)
- OUTPUT-Befehl (z.B. OUTPUT 60, 240)
- Befehle benötigen 6 Taktzyklen, Sprünge (JUMP, CALL, RETURN) benötigen allerdings nur 4 Taktzyklen, da ALU, Register und Scratchpad RAM nicht benötigt werden

# Instruction PROM

Mikroprozessor  
Architektur

Betrachtete  
Befehle

ADD sX, sY

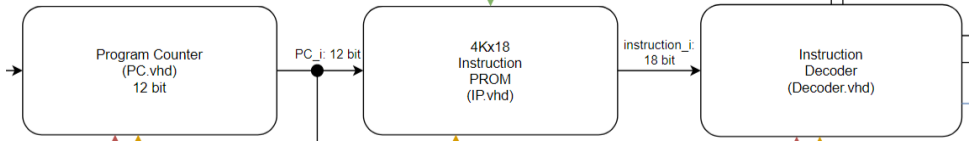
CALL aaa

STORE sX, ss

OUTPUT sX, pp

enable\_i

- Befehl steht im Instruction PROM
- Program Counter gibt an, dass Befehl ausgeführt werden soll
- 18 bit Befehl wird an Instruction Decoder weitergegeben



# Instruction Decoder

Mikroprozessor  
Architektur

Betrachtete  
Befehle

ADD sX, sY

CALL aaa

STORE sX, ss

OUTPUT sX, pp

enable.i

- steuert Komponenten basierend auf der Instruktion an
- → Register soll sX und sY laden
- → ProgramCounter soll Adresse für nächsten Befehl bestimmen
- → MUX vor Register soll Ergebnis des Befehls wieder in Register sX speichern
- → ALU soll einen ADD sX, sY Befehl ausführen
- → MUX vor der ALU gibt den Wert aus dem Register an die ALU durch

# Register

Mikroprozessor  
Architektur

Betrachtete  
Befehle

ADD sX, sY

CALL aaa

STORE sX, ss

OUTPUT sX, pp

enable.i

- Register lädt Wert aus Registern sX und sY (Adressen in den Bits 11-8, 7-4) und gibt diese weiter
- MUX vor Register haben 3 Mögliche Input:
  - → INPUT (extern)
  - → ALU output
  - → Scratchpad RAM output
- entscheiden, welcher Wert wieder in Register sX gespeichert wird
- bei Befehl ADD sX, sY: Output der ALU

# Register

Mikroprozessor  
Architektur

Betrachtete  
Befehle

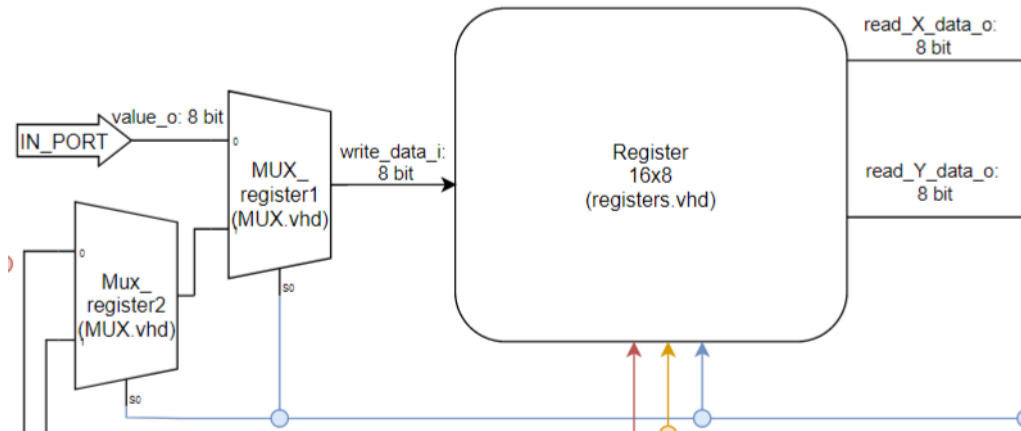
ADD sX, sY

CALL aaa

STORE sX, ss

OUTPUT sX, pp

enable.i



# ALU

Mikroprozessor  
Architektur

Betrachtete  
Befehle

ADD sX, sY

CALL aaa

STORE sX, ss

OUTPUT sX, pp

enable.i

- Befehl wird Abhängig von Bits 17-12 ausgeführt, bei OP-Code 000000 ist das der Befehl ADD sX, sY
- MUX vor der ALU entscheidet, ob geladener Wert aus Register oder Immediate Wert von Instruction Decoder an ALU weitergegeben wird, bei ADD sX, sY: Wert aus Register
- Ergebnis der Berechnung wird an Register zurück gegeben
- Zero und Carry Bits werden gesetzt und an Instruction Decoder gegeben



# Program Counter

Mikroprozessor  
Architektur

Betrachtete  
Befehle

ADD sX, sY

CALL aaa

STORE sX, ss

OUTPUT sX, pp

enable.i

- Program Counter bestimmt Adresse des nächsten Befehls
- MUX vor Program Counter entscheidet, ob Adresse inkrementiert wird, oder ob man an eine spezielle Adresse springt
- → nach einem ADD sX, sY Befehl wird die Adresse einfach inkrementiert

# Instruction Decoder

Mikroprozessor  
Architektur

Betrachtete  
Befehle

ADD sX, sY

CALL aaa

STORE sX, ss

OUTPUT sX, pp

enable.i

- Call/Return Stack speichert Adresse des aktuellen Befehls auf den Stack
- MUX vor Program Counter bekommt vom Instruction Decoder Adresse aaa des nächsten Befehls
- MUX vor PC bekommt auch die Information, dass er Adresse aaa wählen soll

# Instruction Decoder

Mikroprozessor  
Architektur

Betrachtete  
Befehle

ADD sX, sY

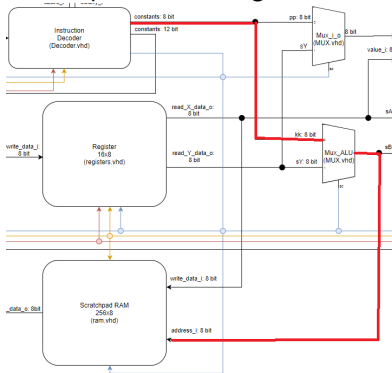
CALL aaa

STORE sX, ss

OUTPUT sX, pp

enable\_i

- Register soll Wert aus sX laden
- MUX vor ALU gibt Immediate Wert vom Instruction Decoder an Scratchpad RAM weiter
- Scratchpad RAM soll geladenen Wert an Adresse ss speichern



# Instruction Decoder

Mikroprozessor  
Architektur

Betrachtete  
Befehle

ADD sX, sY

CALL aaa

STORE sX, ss

OUTPUT sX, pp

enable.i

- MUX\_i\_o soll Port für Output bestimmen, bei OUTPUT sX, pp: immediate Wert pp
- Register soll Wert von Adresse sX ausgeben
- Wert von Register wird an OUT\_PORT ausgegeben
- → bei Port 60 mit Wert 240 in Register sX wird eine rote LED auf dem Board auf nahezu maximale Helligkeit gestellt

# Enable Bits

Mikroprozessor  
Architektur

Betrachtete  
Befehle

ADD sX, sY

CALL aaa

STORE sX, ss

OUTPUT sX, pp

enable.i

- Der Instruction Decoder setzt alle enable.i-Bits der Komponenten, die im aktuellen Befehl nicht verwendet werden auf 0.
- → bei einem OUTPUT Befehl ist das write\_enable.i Bit beim Register auf 0, da nur Werte gelesen werden etc.
- ist Grund, weshalb Befehle mehr als einen Taktzyklus brauchen

# Microcontroller

## Mikroprozessor Architektur

Betrachtete  
Befehle

ADD sX, sY

CALL aaa

STORE sX, ss

OUTPUT sX, pp

enable.i

