Hardware Modeling [VU] (191.011)
– WS25 –

Introduction to Hardware Design

Florian Huemer & Sebastian Wiedemann & Dylan Baumann

WS 2025/26

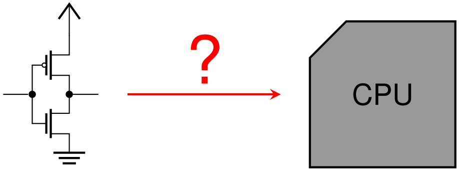
Modified: 2025-12-16, 16:03 (f8a58e9)

Motivation

- How to go from simple circuits to complex ones?

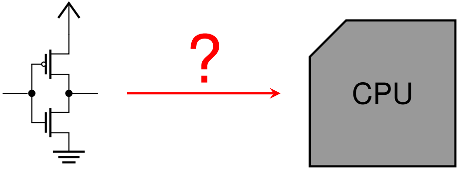
Motivation

- How to go from simple circuits to complex ones?
  - Up to **billions** of transistors



Motivation

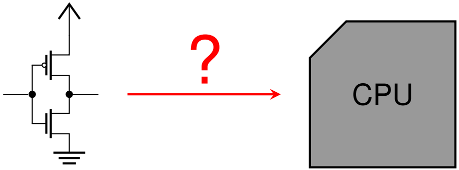
- How to go from simple circuits to complex ones?
  - Up to **billions** of transistors
  - Complexity continuously increasing (*Moore's Law*)



Motivation

- How to go from simple circuits to complex ones?
  - Up to **billions** of transistors
  - Complexity continuously increasing (*Moore's Law*)

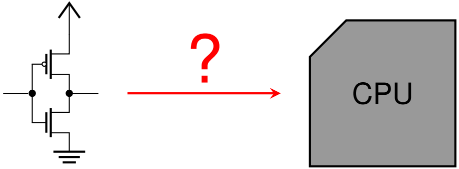
⇒ Hardware Modeling



Motivation

- How to go from simple circuits to complex ones?
  - Up to **billions** of transistors
  - Complexity continuously increasing (*Moore's Law*)

⇒ Hardware Modeling
  - Tools and techniques to bridge the gap



Why bother?

- Why should you care about designing hardware?

Why bother?

- Why should you care about designing hardware?

⇒ Same as for software

Why bother?

- Why should you care about designing hardware?

⇒ Same as for software

  - Custom requirements ⇒ custom solution

Why bother?

- Why should you care about designing hardware?

⇒ Same as for software

  - Custom requirements ⇒ custom solution
  - Required for niche applications

Why bother?

- Why should you care about designing hardware?

⇒ Same as for software

  - Custom requirements ⇒ custom solution
  - Required for niche applications
  - Reduce overhead

Why bother?

- Why should you care about designing hardware?

⇒ Same as for software

  - Custom requirements ⇒ custom solution
  - Required for niche applications
  - Reduce overhead

Why bother? (Cont'd)

- Become a better programmer

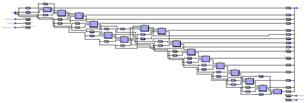
Why bother? (Cont'd)

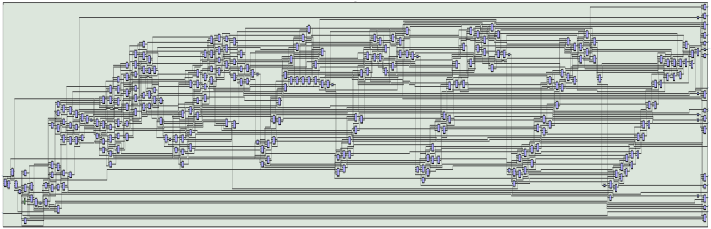
- Become a better programmer
  - Understand hardware limits

Why bother? (Cont'd)

- Become a better programmer
  - Understand hardware limits

- Example: addition and division in same technology

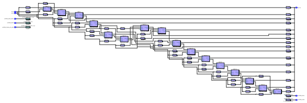


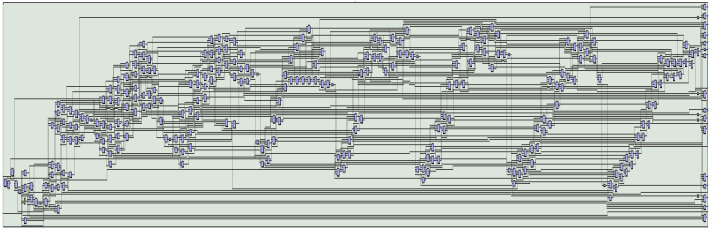


Why bother? (Cont’d)

- Become a better programmer
  - Understand hardware limits
  - Know which knobs to turn

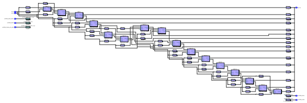
- Example: addition and division in same technology

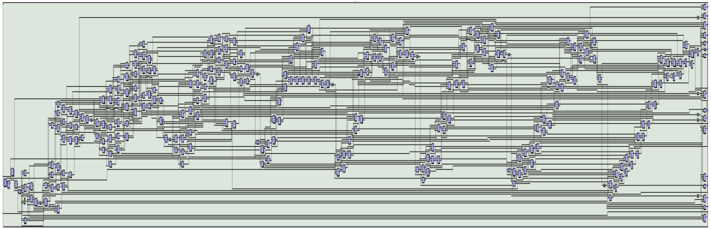




Why bother? (Cont’d)

- Become a better programmer
  - Understand hardware limits
  - Know which knobs to turn
  - New way of thinking
- Example: addition and division in same technology





Differences to Software Design

- Software

- Hardware

Differences to Software Design

- Software
  - Typically sequential
  - Concurrency possible but takes care
- Hardware

Differences to Software Design

- Software
  - Typically sequential
  - Concurrency possible but takes care
- Hardware
  - Typically concurrent
  - Sequential possible but takes care

Differences to Software Design

- Software
  - Typically sequential
  - Concurrency possible but takes care
  - Asymptotic behavior (mostly)

- Hardware
  - Typically concurrent
  - Sequential possible but takes care

Differences to Software Design

- Software
  - Typically sequential
  - Concurrency possible but takes care
  - Asymptotic behavior (mostly)

- Hardware
  - Typically concurrent
  - Sequential possible but takes care
  - Details matter

Differences to Software Design

- Software
  - Typically sequential
  - Concurrency possible but takes care
  - Asymptotic behavior (mostly)
  - Easy to update
- Hardware
  - Typically concurrent
  - Sequential possible but takes care
  - Details matter

Differences to Software Design

- Software
  - Typically sequential
  - Concurrency possible but takes care
  - Asymptotic behavior (mostly)
  - Easy to update
- Hardware
  - Typically concurrent
  - Sequential possible but takes care
  - Details matter
  - First-time-right paradigm

Differences to Software Design

- Software
  - Typically sequential
  - Concurrency possible but takes care
  - Asymptotic behavior (mostly)
  - Easy to update
- Hardware
  - Typically concurrent
  - Sequential possible but takes care
  - Details matter
  - First-time-right paradigm

Differences to Software Design

- Software
  - Typically sequential
  - Concurrency possible but takes care
  - Asymptotic behavior (mostly)
  - Easy to update
- Hardware
  - Typically concurrent
  - Sequential possible but takes care
  - Details matter
  - First-time-right paradigm

Takeaway

This duality makes hardware design hard but also rewarding

Comparison to Software Design (Cont’d)

- Software

- Hardware

Comparison to Software Design (Cont’d)

- Software

```
if (x > y)
  z = x * y;
else
  z = x + y;
return z;
```

- Hardware

Comparison to Software Design (Cont'd)

- Software
  - Sequential execution
  - Either multiplication or addition

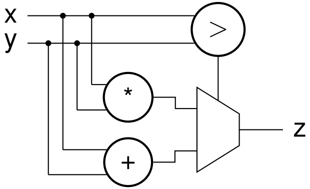
```
if (x > y)
  z = x * y;
else
  z = x + y;
return z;
```

- Hardware

Comparison to Software Design (Cont'd)

- Software
  - Sequential execution
  - Either multiplication or addition
- Hardware

```
if (x > y)
  z = x * y;
else
  z = x + y;
return z;
```

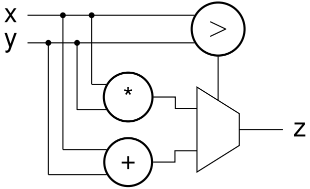


Comparison to Software Design (Cont’d)

- Software
  - Sequential execution
  - Either multiplication or addition

- Hardware
  - Computations done concurrently
  - All operations always active

```
if (x > y)
  z = x * y;
else
  z = x + y;
return z;
```



Gajski Y-Chart

- Abstraction is key

Gajski Y-Chart

- Abstraction is key
  - Start on high abstraction and (automatically) move inwards

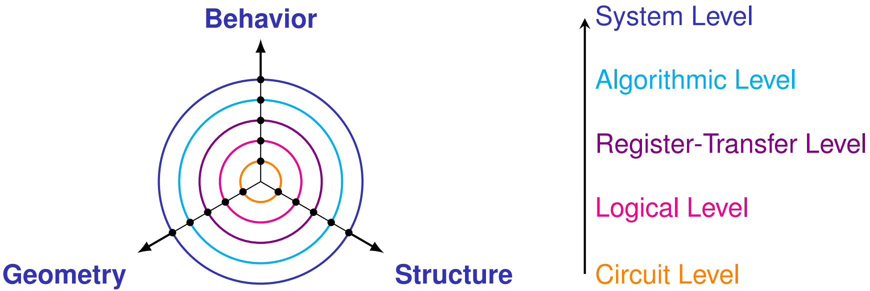
Gajski Y-Chart

- Abstraction is key
  - Start on high abstraction and (automatically) move inwards
- All points of view describe same circuit

Gajski Y-Chart

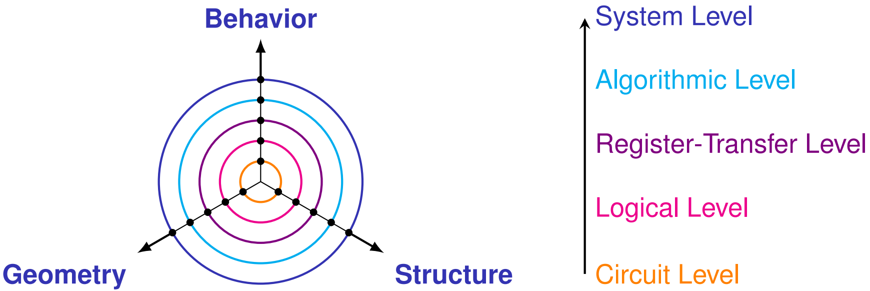
- Abstraction is key
  - Start on high abstraction and (automatically) move inwards

- All points of view describe same circuit



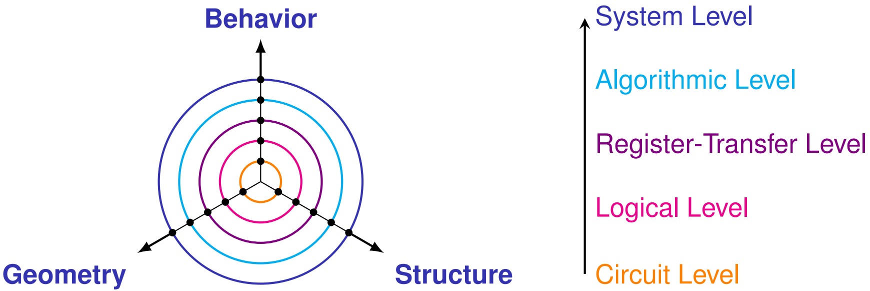
Gajski Y-Chart

- Abstraction is key
  - Start on high abstraction and (automatically) move inwards
- All points of view describe same circuit
  - Translate between them as beneficial



Gajski Y-Chart

- Abstraction is key
  - Start on high abstraction and (automatically) move inwards
- All points of view describe same circuit
  - Translate between them as beneficial
  - Harnessed by tools



Behavior

Geometry

Structure

System Level

Algorithmic Level

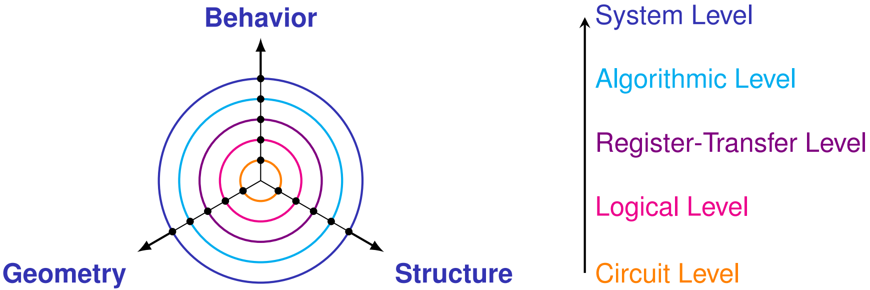
Register-Transfer Level

Logical Level

Circuit Level

Gajski Y-Chart

- Abstraction is key
  - Start on high abstraction and (automatically) move inwards
  - Catch: Increasing abstraction ⇒ decreased optimization potential
- All points of view describe same circuit
  - Translate between them as beneficial
  - Harnessed by tools



Behavior

Geometry

Structure

System Level

Algorithmic Level

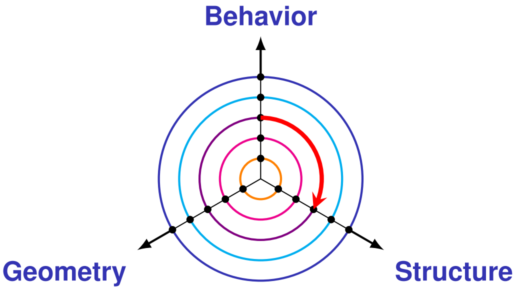
Register-Transfer Level

Logical Level

Circuit Level

Gajski Y-Chart

- Abstraction is key
  - Start on high abstraction and (automatically) move inwards
  - Catch: Increasing abstraction ⇒ decreased optimization potential
- All points of view describe same circuit
  - Translate between them as beneficial
  - Harnessed by tools



System Level

Algorithmic Level

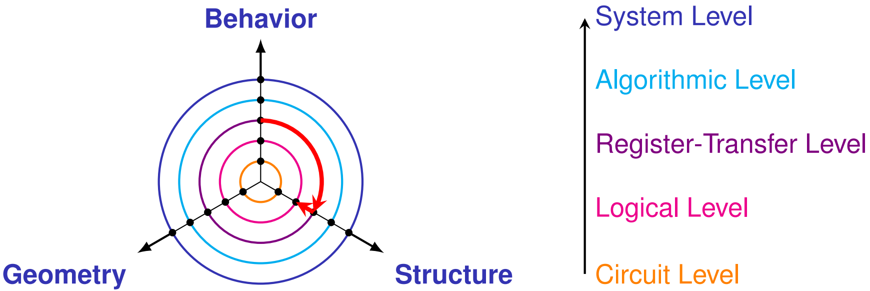
Register-Transfer Level

Logical Level

Circuit Level

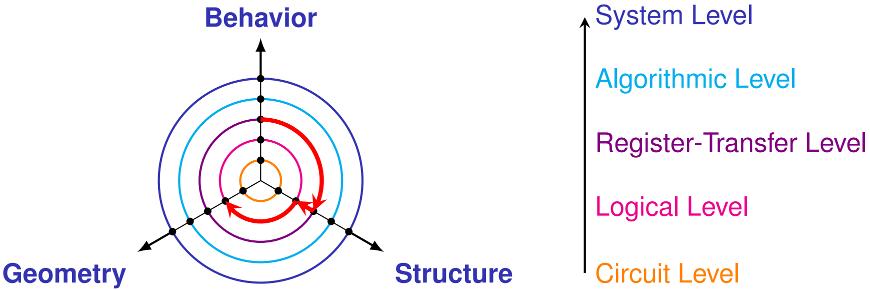
Gajski Y-Chart

- Abstraction is key
  - Start on high abstraction and (automatically) move inwards
  - Catch: Increasing abstraction ⇒ decreased optimization potential
- All points of view describe same circuit
  - Translate between them as beneficial
  - Harnessed by tools



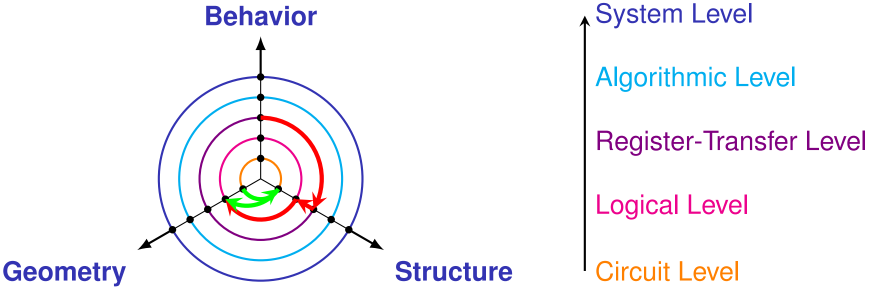
Gajski Y-Chart

- Abstraction is key
  - Start on high abstraction and (automatically) move inwards
  - Catch: Increasing abstraction ⇒ decreased optimization potential
- All points of view describe same circuit
  - Translate between them as beneficial
  - Harnessed by tools



Gajski Y-Chart

- Abstraction is key
  - Start on high abstraction and (automatically) move inwards
  - Catch: Increasing abstraction ⇒ decreased optimization potential
- All points of view describe same circuit
  - Translate between them as beneficial
  - Harnessed by tools



Behavior

Geometry

Structure

System Level

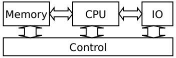
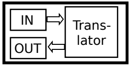
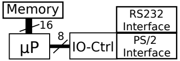
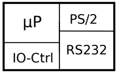
Algorithmic Level

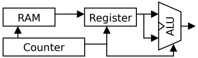
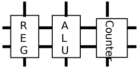
Register-Transfer Level

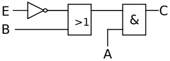
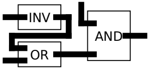
Logical Level

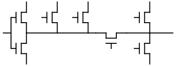
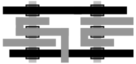
Circuit Level

Y-Table

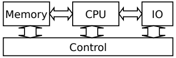
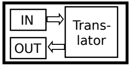
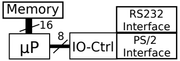
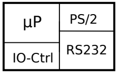
	Behavior	Structure	Geometry
System Level	Inputs : Keyboard Output: Display Function:		
Algorithmic Level	while input read English text translate to German output German Text	 Memory 16 8 µP IO-Ctrl RS232 Interface PS/2 Interface	

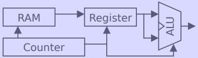
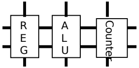
Register Transfer Level (RTL)	if A=`1` then
	B:= B+1
	else
	B:= B
	end if
			

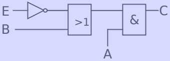
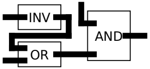
Logic Level	D = NOT E
	
	C = (D OR B) AND A
			

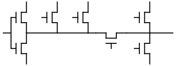
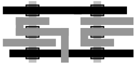
Circuit Level	$\frac{dU}{dt} = R\frac{dI}{dt} + \frac{I}{C} + L\frac{d^2I}{dt^2}$		

Y-Table

	Behavior	Structure	Geometry
System Level	Inputs : Keyboard Output: Display Function:		
Algorithmic Level	while input read English text translate to German output German Text		

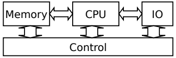
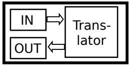
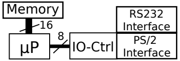
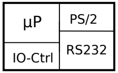
Register Transfer Level (RTL)	if A=`1` then
	B:= B+1
	else
	B:= B
	end if
			

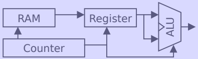
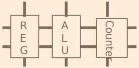
Logic Level	D = NOT E
	C = (D OR B) AND A
			

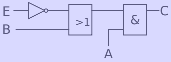
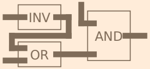
Circuit Level	$\frac{dU}{dt} = R\frac{dI}{dt} + \frac{I}{C} + L\frac{d^2I}{dt^2}$		

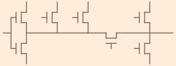
HDLs

Y-Table

	Behavior	Structure	Geometry
System Level	Inputs : Keyboard Output: Display Function:		
Algorithmic Level	while input read English text translate to German output German Text		

Register Transfer Level (RTL)	if A=`1` then
	B:= B+1
	else
	B:= B
	end if
			

Logic Level	D = NOT E
	C = (D OR B) AND A
			

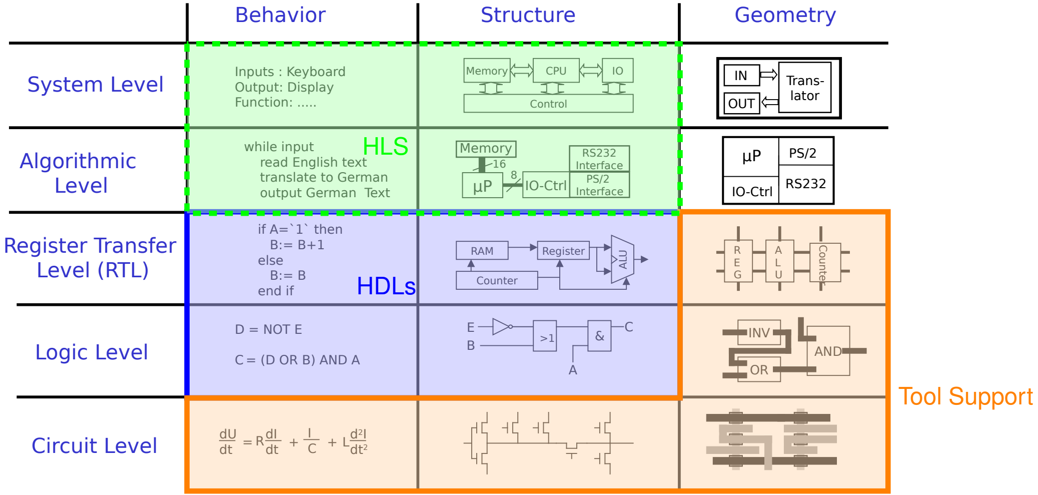
Circuit Level	$\frac{dU}{dt} = R\frac{dI}{dt} + \frac{I}{C} + L\frac{d^2I}{dt^2}$		

Memory
CPU
IO
Control
IN
OUT
Trans-lator
Memory
16
8
µP
IO-Ctrl
RS232 Interface
PS/2 Interface
µP
PS/2
IO-Ctrl
RS232
RAM
Register
ALU
Counter
REG
ALU
Counter
E
B
>1
&
C
A
INV
AND
OR

HDLs

Tool Support

Y-Table



	Behavior	Structure	Geometry
System Level	Inputs : Keyboard Output: Display Function:	Memory CPU IO Control	IN OUT Trans-lator
Algorithmic Level	while input read English text translate to German output German Text	Memory 16 8 µP IO-Ctrl RS232 Interface PS/2 Interface	µP PS/2 IO-Ctrl RS232

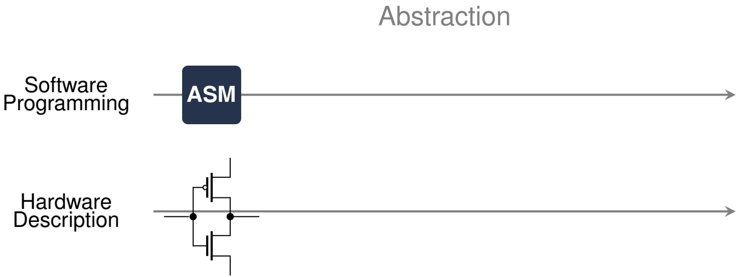
Register Transfer Level (RTL)	if A=`1` then
	B:= B+1
	else
	B:= B
	end if
		RAM
		Register
		Counter
		ALU
			REG
			ALU
			Counter

Logic Level	D = NOT E
	C = (D OR B) AND A
		E
		B
		>1
		&
		C
		A
			INV
			AND
			OR

Circuit Level	$\frac{dU}{dt} = R\frac{dI}{dt} + \frac{I}{C} + L\frac{d^2I}{dt^2}$		

Hardware Description Languages

- Drawing circuits does not scale



Abstraction

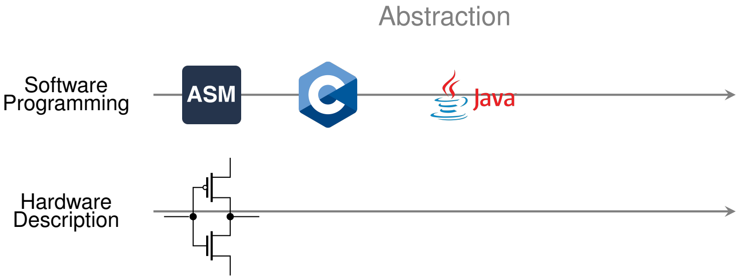
Software Programming

ASM

Hardware Description

Hardware Description Languages

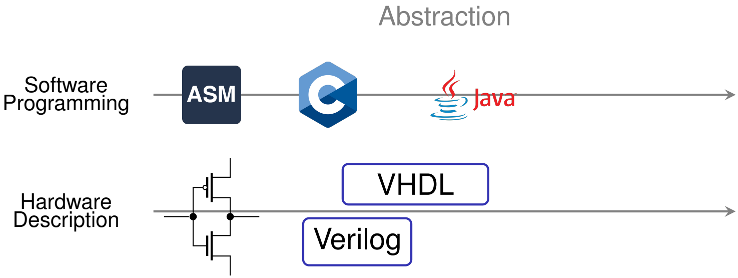
- Drawing circuits does not scale
  - Require more abstract method



Hardware Description Languages

- Drawing circuits does not scale
  - Require more abstract method

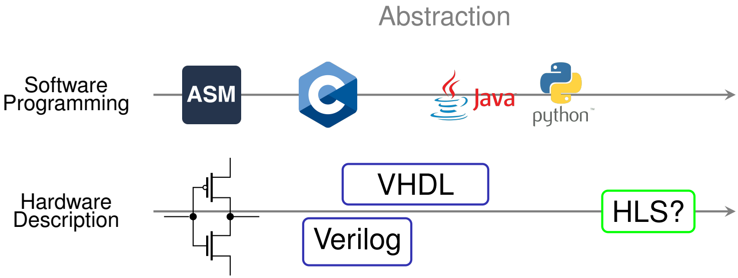
⇒ *Hardware Description Languages* (HDLs)
  - Most popular: VHDL, (System)Verilog



Hardware Description Languages

- Drawing circuits does not scale
  - Require more abstract method

⇒ *Hardware Description Languages* (HDLs)
  - Most popular: VHDL, (System)Verilog



Abstraction

Software Programming

Hardware Description

VHDL

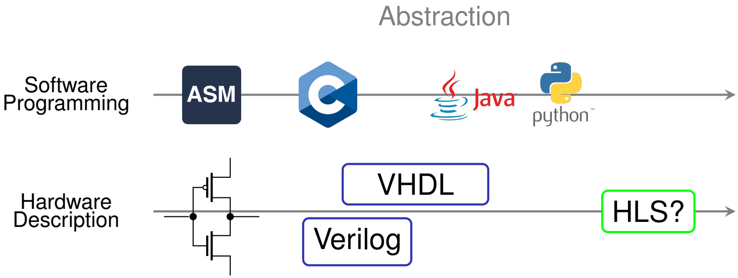
Verilog

HLS?

Hardware Description Languages

- Drawing circuits does not scale
  - Require more abstract method

⇒ *Hardware Description Languages* (HDLs)
  - Most popular: VHDL, (System)Verilog



We will use VHDL! But why?

We will use VHDL! But why?

- Verbose code

We will use VHDL! But why?

- Verbose code
- Strongly typed
    - Harder to make subtle mistakes

We will use VHDL! But why?

- Verbose code
- Strongly typed
  - Harder to make subtle mistakes
- Highly structured and modular

We will use VHDL! But why?

- Verbose code
- Strongly typed
  - Harder to make subtle mistakes
- Highly structured and modular
- Different from what you know

VHDL Standard

- The latest VHDL standard (2019) can be found here

VHDL Standard

- The latest VHDL standard (2019) can be found here
  - Download through the TU network (e.g., via eduroam or VPN connection)

VHDL Standard

- The latest VHDL standard (2019) can be found here
  - Download through the TU network (e.g., via eduroam or VPN connection)
- Watch out for VHDL standard and implementation references

1

VHDL Standard

- The latest VHDL standard (2019) can be found here
  - Download through the TU network (e.g., via eduroam or VPN connection)
- Watch out for VHDL standard and implementation references

1

IEEE SA OPEN

Clickable

Lecture Complete!