

Entwicklung und Implementierung eines digitalen Funktionsgenerators in VHDL

Markus Hartlage

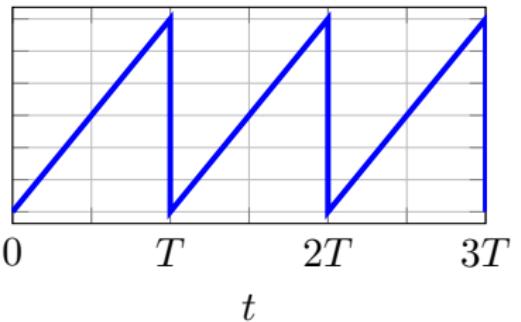
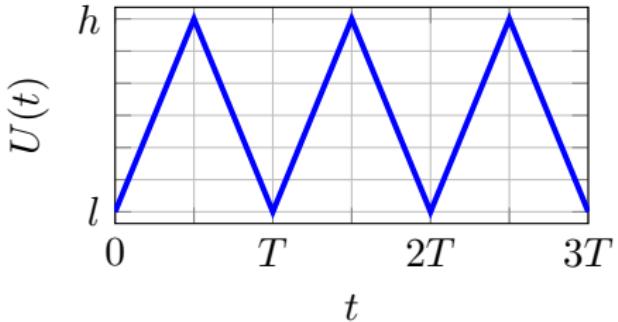
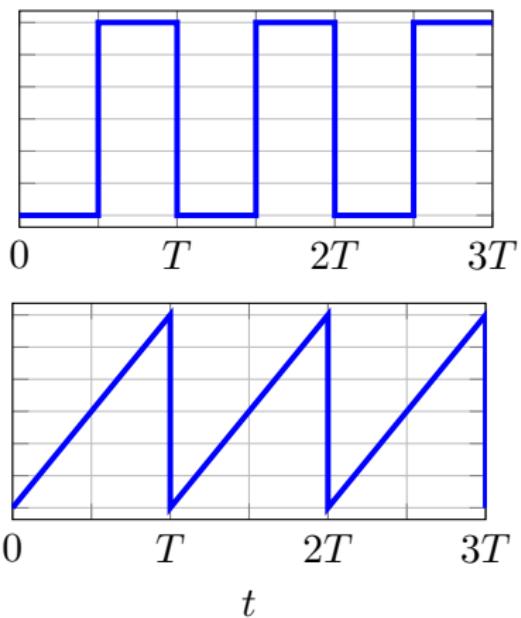
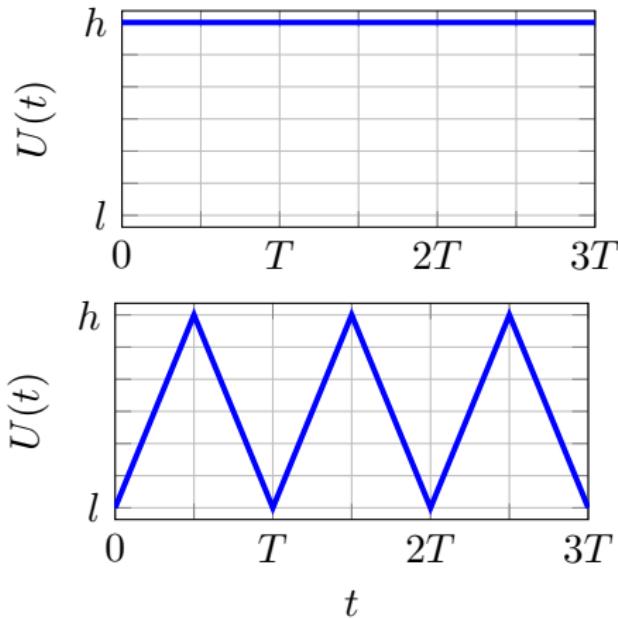
FH-Bielefeld

April 3, 2022



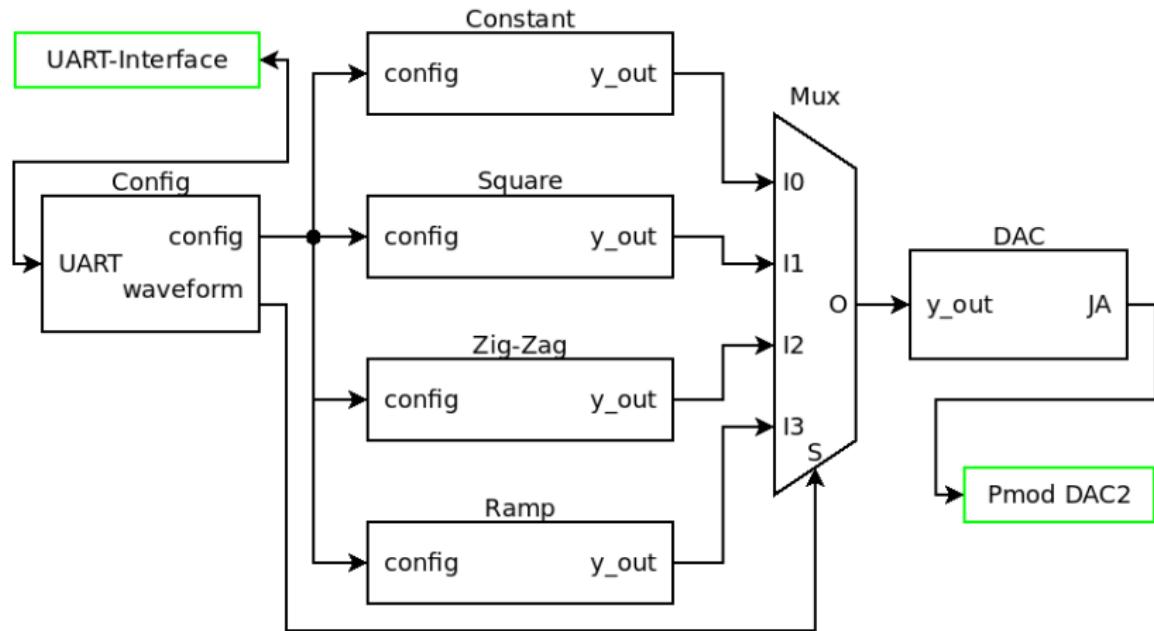
Konzept - Anforderungen

- ▶ Ausgabe vier verschiedener Funktionsverläufe
 - ▶ Konstante, Rechteck, Zick-Zack, Rampe
- ▶ Konfiguration per UART-Schnittstelle



Konzept - Aufbau

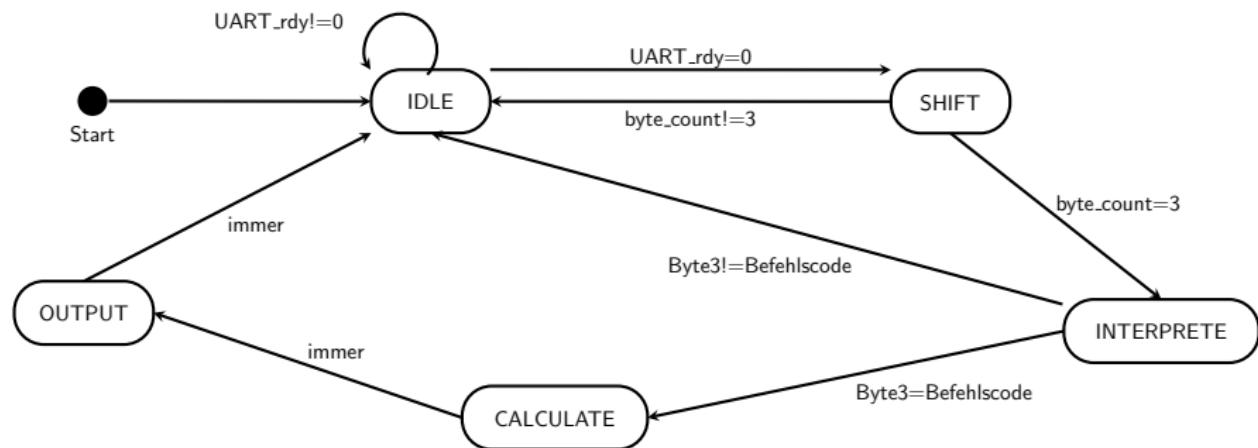
- ▶ Aufbau aus Konfigurations-, Funktions- und DAC Komponente
- ▶ zusätzliche Hardware: Uart-Interface und digital-analog Konverter



Komponenten - Konfiguration

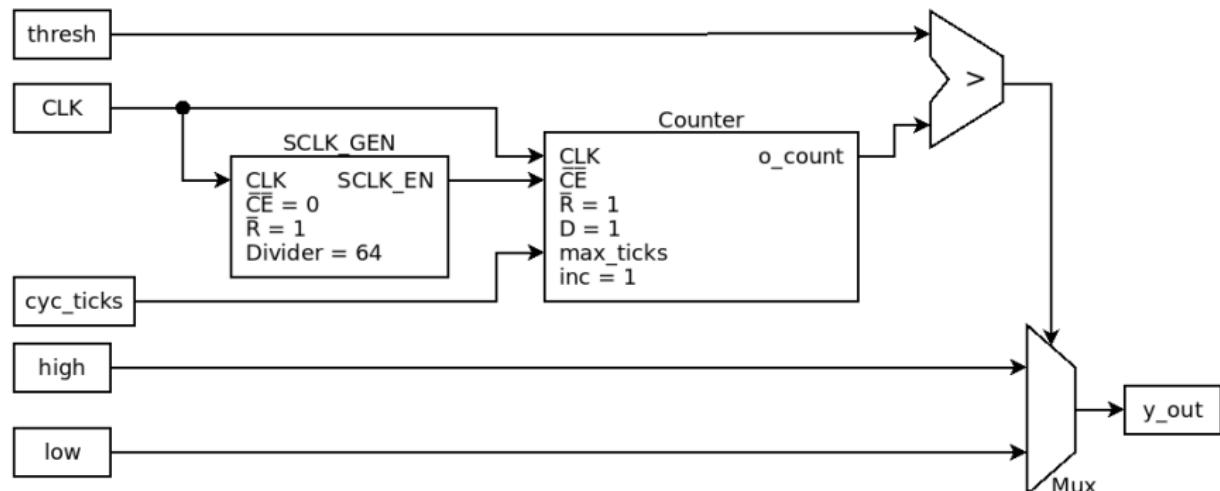
Aufbau als state machine:

- ▶ Byteweises Einlesen der UART Rx Signale
- ▶ Interpretation von vier Bytes als Befehl (Befehlscode + Argumente)
- ▶ Berechnung und Speicherung der neuen Konfiguration



Komponenten - Rechteck

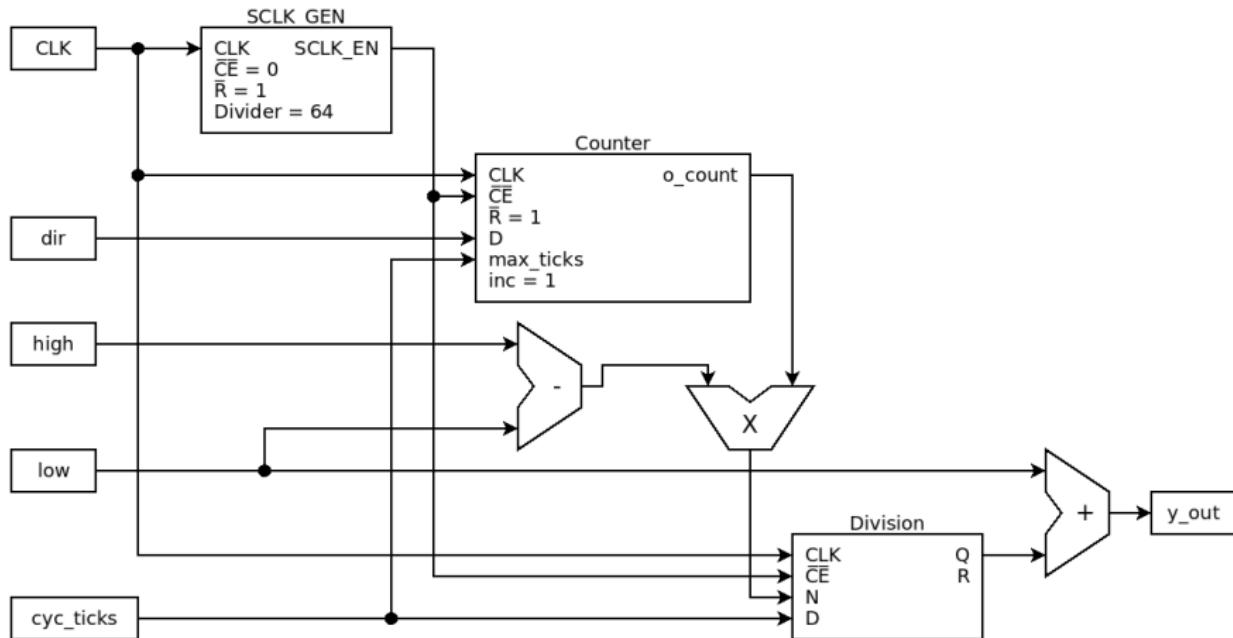
- nach 64 steigenden Flanken *Counter* aktivieren
- Ausgang auf *low* wenn *o_count* > *thresh*, sonst *high*



Komponenten - Rampe

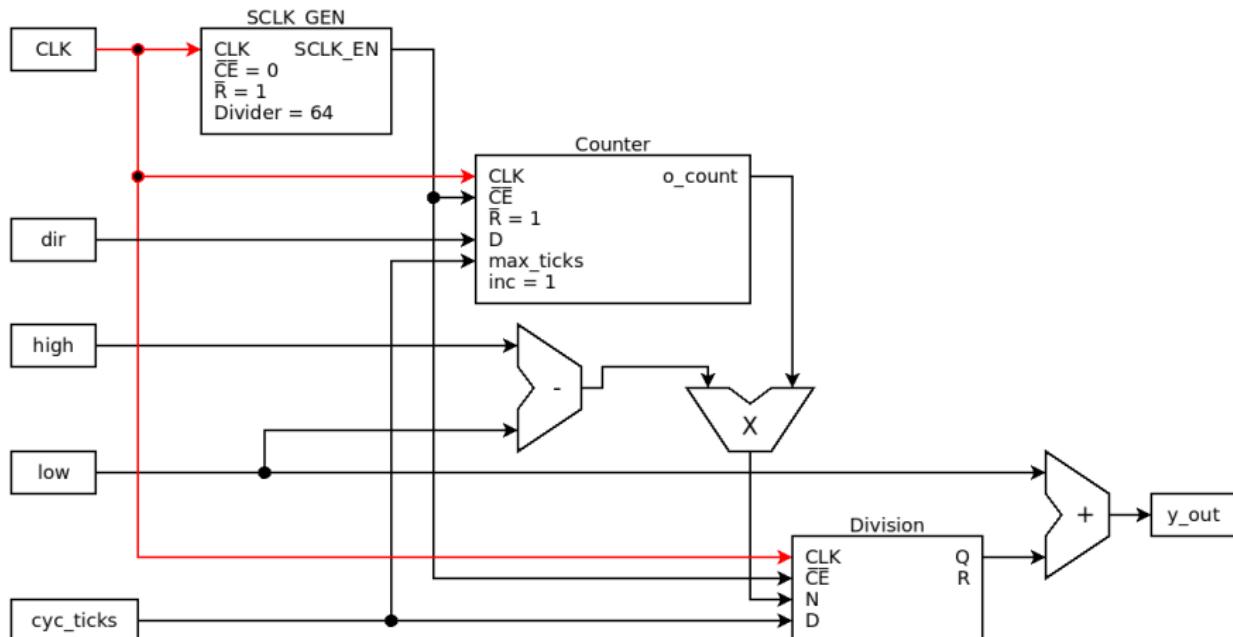
- Abbildung des 24 Bit Zählstands auf 12 Bit Ausgang:

$$y_{out} = \frac{(high - low) \cdot o_count}{cyc_ticks}$$



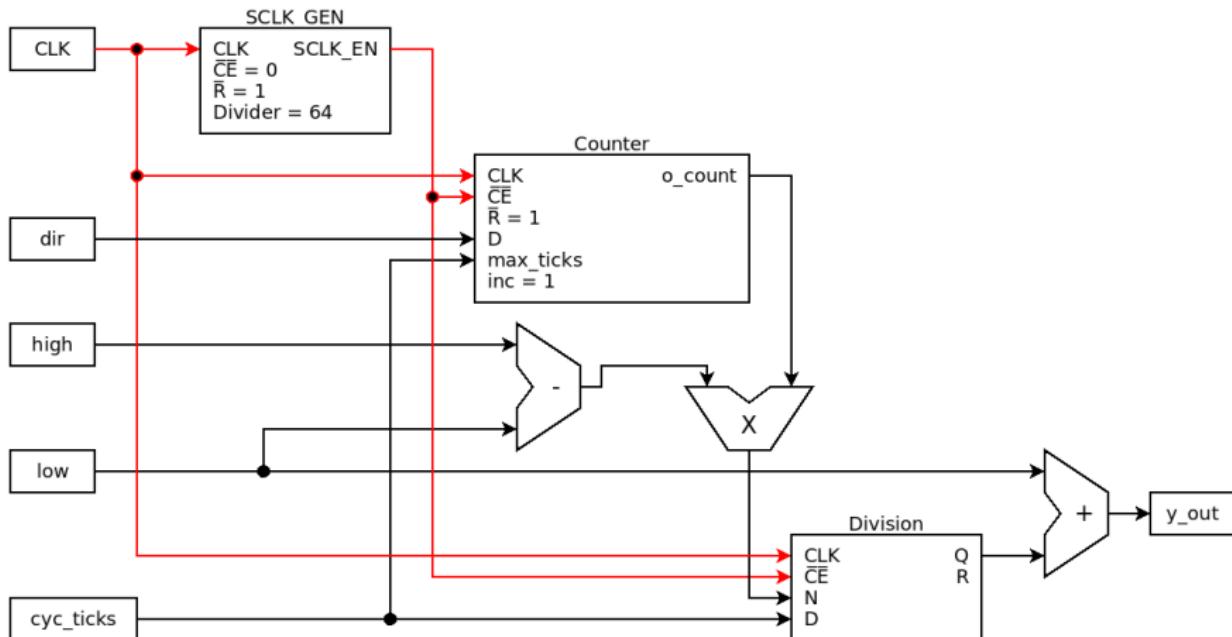
Komponenten - Rampe

- Abzählen von 64 Taktzyklen



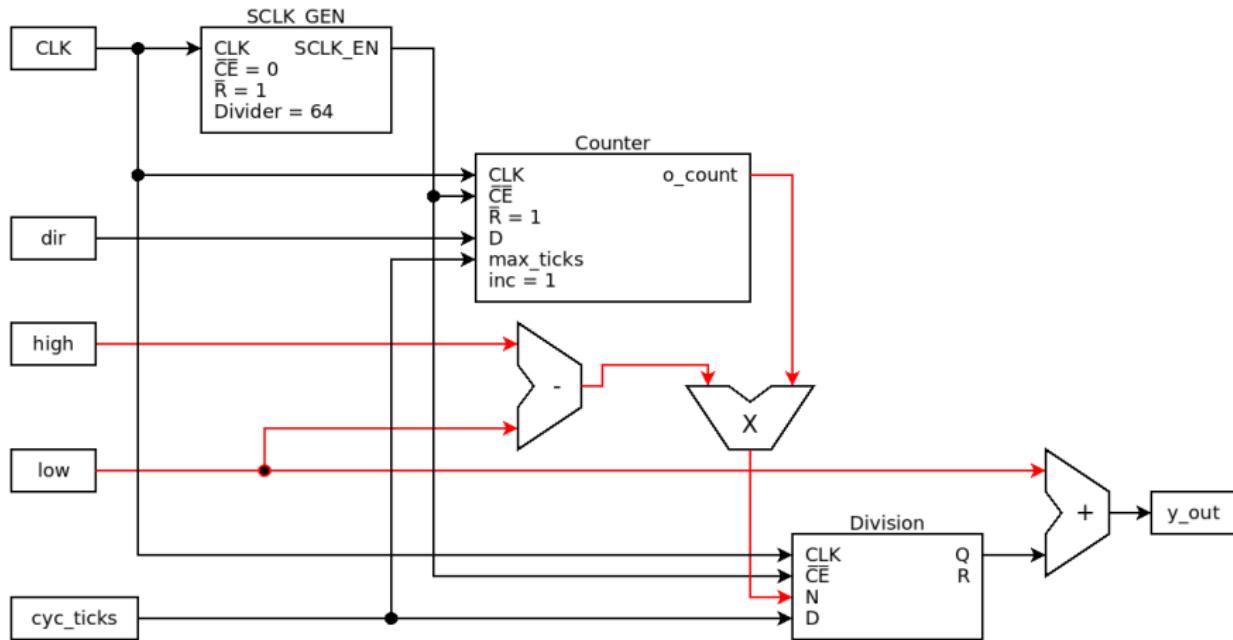
Komponenten - Rampe

- Aktivieren von *Counter* und *Division*



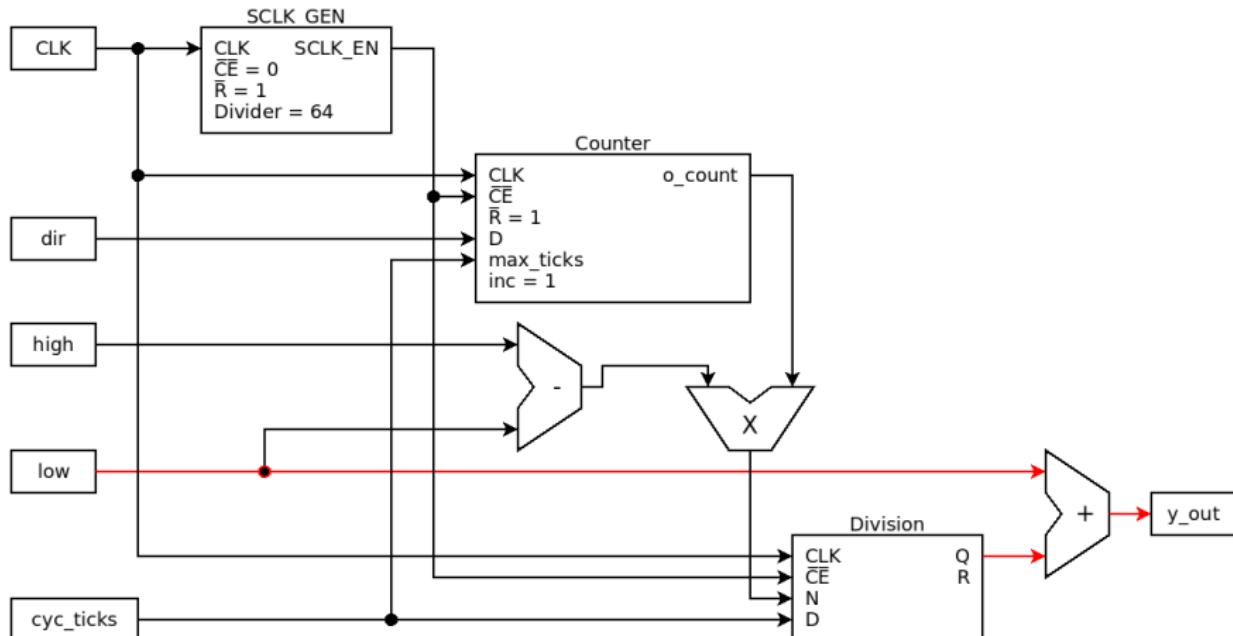
Komponenten - Rampe

- neuer Zählstand wird mit Amplitude multipliziert und *Division* beginnt zu teilen



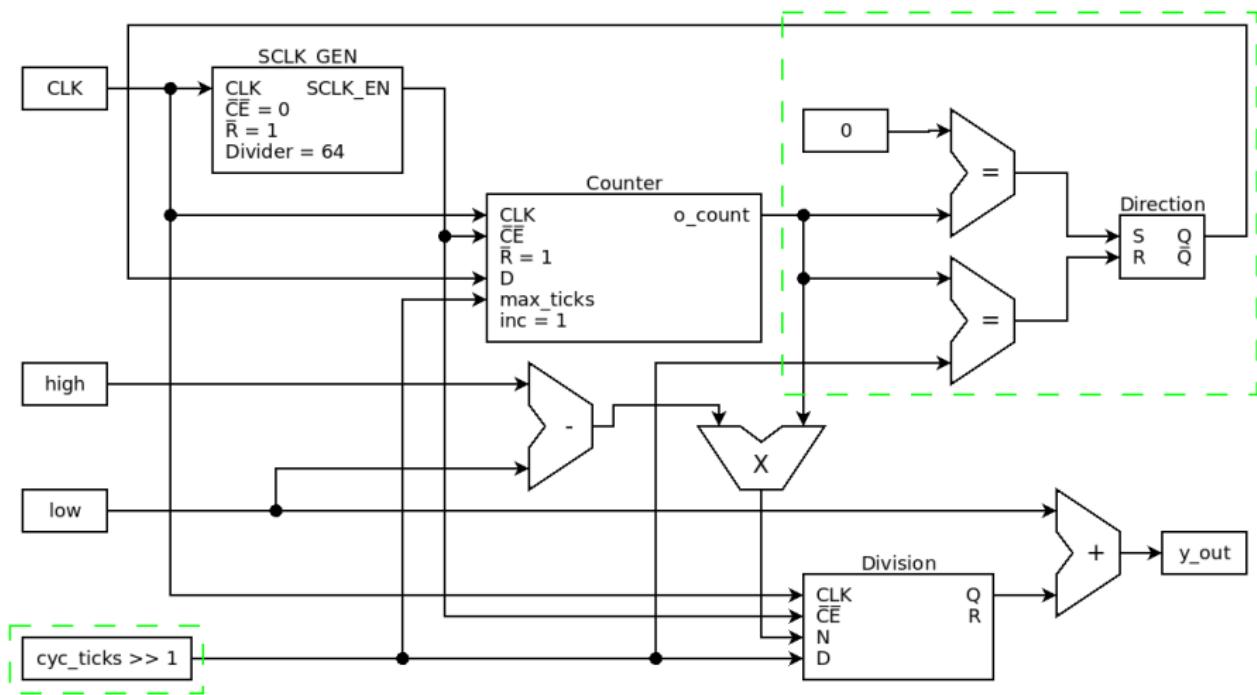
Komponenten - Rampe

- ▶ untere zwölf Bits von Q plus low ergeben y_{out}



Komponenten - Zick-Zack

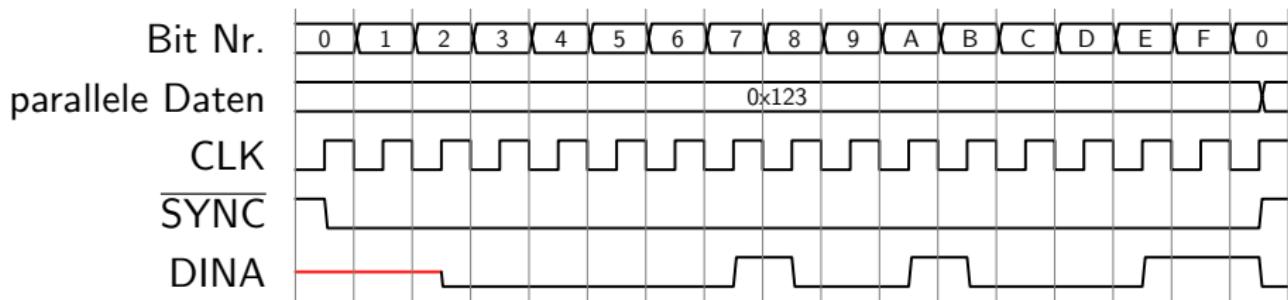
- Zählrichtung wird mit Komparatoren geregelt
- halbe Zykluszeit (*cyc_ticks* um ein Bit nach rechts geschoben)



Komponenten - Digital-Analog-Konverter

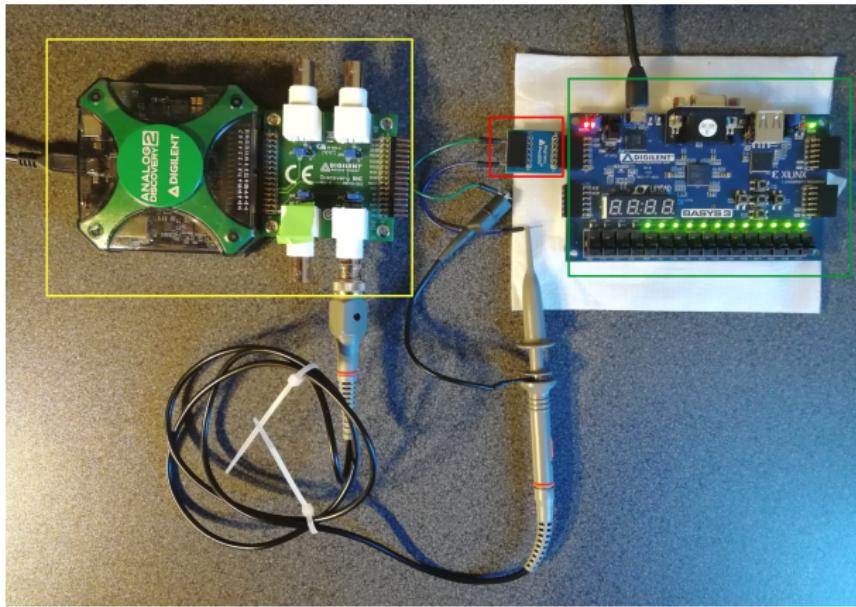
- Implementierung des PmodDA2-Protokolls

- SYNC: Chip-Enable Signal für den DAC
- DINA, DINB: serielle DAC-Daten, aktuell nur Kanal A genutzt
- CLK: DAC-Clock läuft mit 25 MHz



Funktionstest - Versuchsaufbau

- ▶ Implementierung auf Basys 3 Board
- ▶ Messung mit Analog Discovery 2



Funktionstest - Versuchsdurchführung

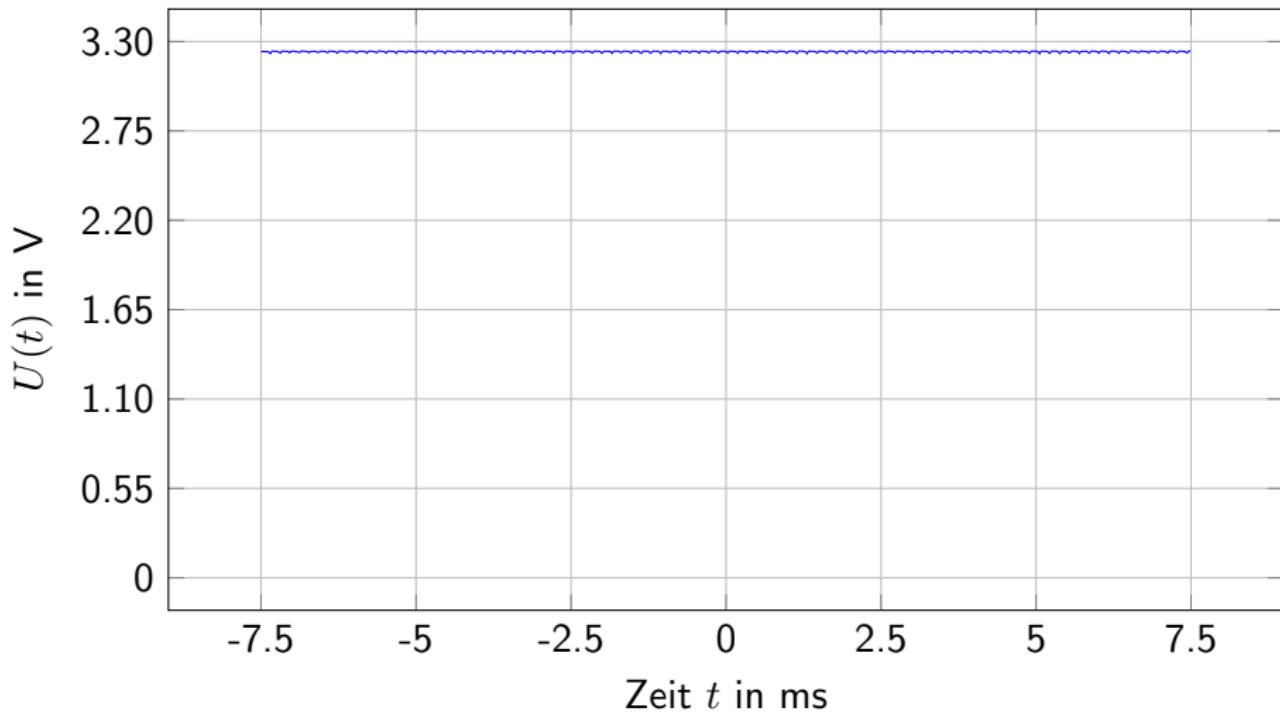
- ▶ Ausführen und Kontrollieren von jedem Konfigurationsbefehl
- ▶ Jede Funktion einmal bei 100 Hz mit $U_{low} = 0\text{ V}$ und $U_{high} = 3,3\text{ V}$ messen
- ▶ Zick-Zack-Funktion zur Kontrolle des Frequenzbereichs mit folgenden Frequenzen f :

f in Hz	0,08	0,1	1	10	100	10^3	10^4	10^5	$3,5 \cdot 10^5$	10^6
-----------	------	-----	---	----	-----	--------	--------	--------	------------------	--------

- ▶ $0,08\text{ Hz} < f_{min}$ und $1\text{ MHz} > f_{max}$

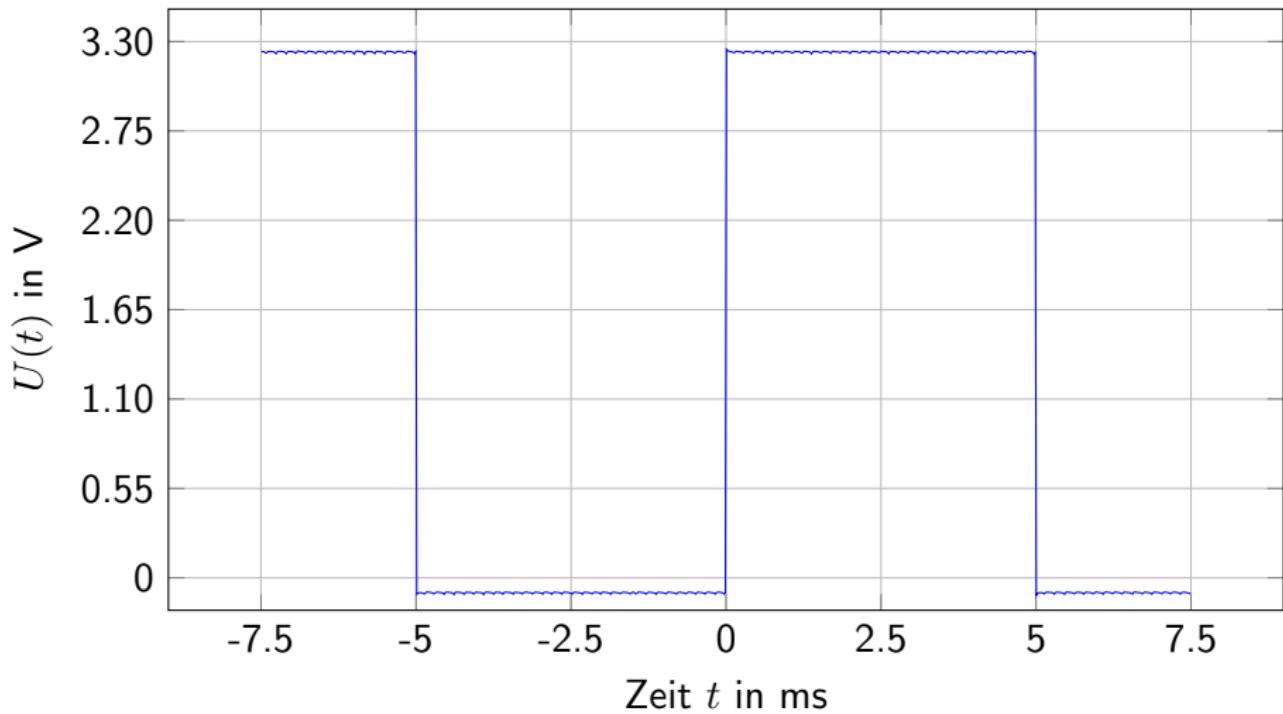
Funktionstest - Konstante bei 100 Hz

- 3,3 V kann nicht genau erreicht werden

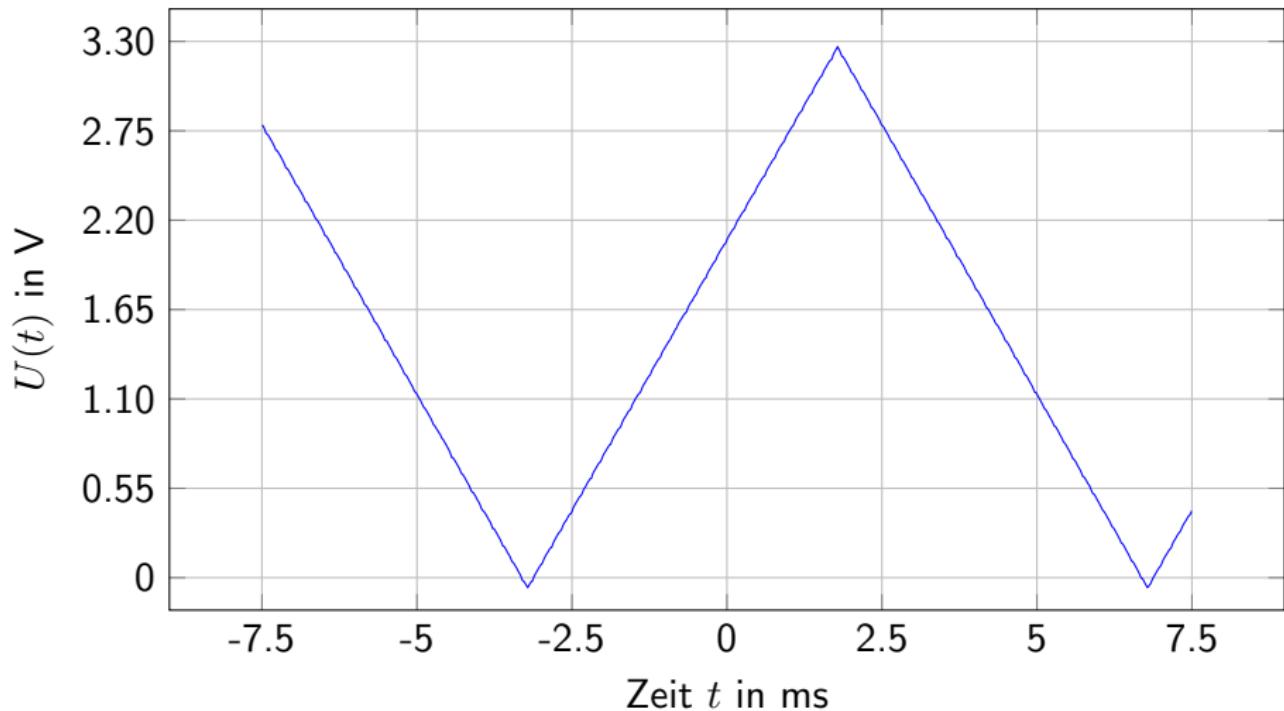


Funktionstest - Rechteck bei 100 Hz

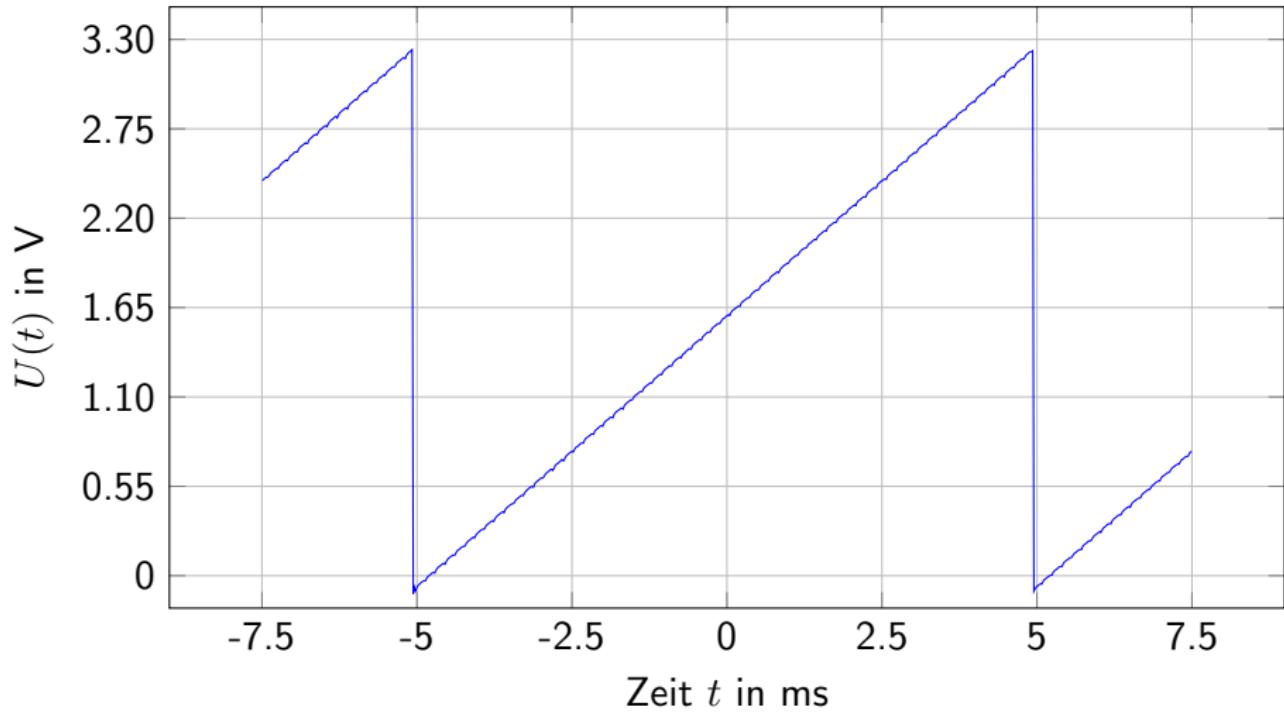
- 0 V wird unterschritten



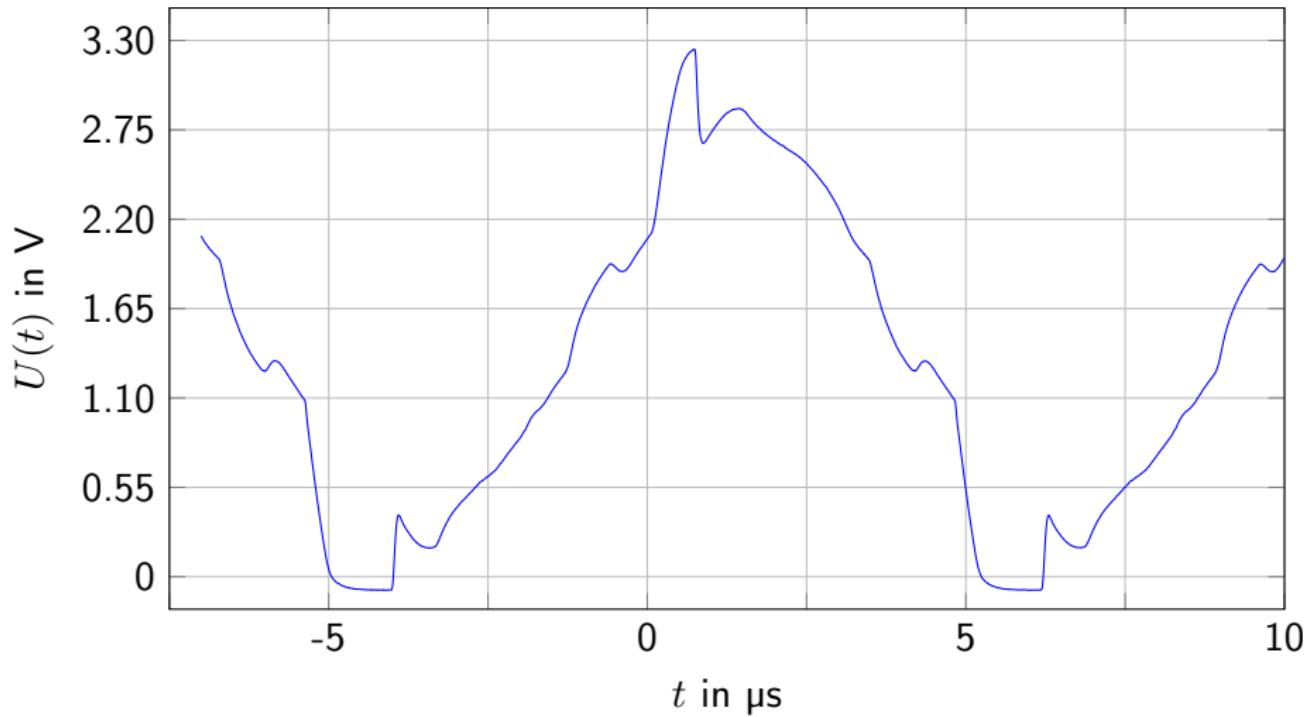
Funktionstest - Zick-Zack bei 100 Hz



Funktions test - Rampe bei 100 Hz



Funktionstest - bei 100 kHz



Referenzen

Source Code, Anleitung und Python Interface:

https://github.com/markushart/studienarbeit_function_generator.git