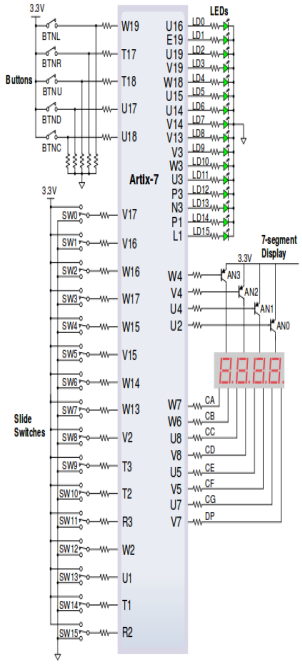
Vivado

Inhaltsverzeichnis

Vivado....1
- Basys3....2
  - Master Constraints File XDC....2
  - Referenz-Projekt....2
- Programmieren / Hardware Manager....2
  - Probleme: mögliche Lösungen....3
- Block Design....4
- FAQ....5
- XADC....5
- FIR....8
- Direct digital synthesizers (DDS)....10
    - SIN/COS LUT....11
    - Phase Generator and SIN/COS LUT (DDS)....11
  - Multichannel....12
- Basic Handshake....13
  - CONFIG Kanal....13
- Beispiel 3 Kanal....14

Basys3



https://reference.digilentinc.com/basys3:basys3

Master Constraints File XDC

https://reference.digilentinc.com/_media/basys3:basys3_master.zip

Referenz-Projekt

https://reference.digilentinc.com/lib/exe/fetch.php?tok=283489&media=https%3A%2F%2Fgithub.com%2FDigilent%2FBasys3%2Farchive%2Fmaster.zip

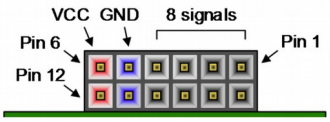
100MZh Clock auf Port W5

Xilinx University Basys3 Board Plugin und Projekte

http://www.instructables.com/id/Basic-Stopwatch-Using-VHDL-and-Basys3-Board/

PMOD Connectors

Pmod JA	Pmod JB	Pmod JC	Pmod JXADC
JA1: J1	JB1: A14	JC1: K17	JXADC1: J3
JA2: L2	JB2: A16	JC2: M18	JXADC2: L3
JA3: J2	JB3: B15	JC3: N17	JXADC3: M2
JA4: G2	JB4: B16	JC4: P18	JXADC4: N2
JA7: H1	JB7: A15	JC7: L17	JXADC7: K3
JA8: K2	JB8: A17	JC8: M19	JXADC8: M3
JA9: H2	JB9: C15	JC9: P17	JXADC9: M1
JA10: G3	JB10: C16	JC10: R18	JXADC10: N1



Programmieren / Hardware Manager

* Programmier-Jumper neben USB-A Buchse in Mittelstellung (JTAG)

* Strom einschalten

* USB Programmier-Kabel in Mikro-Usb Buchse

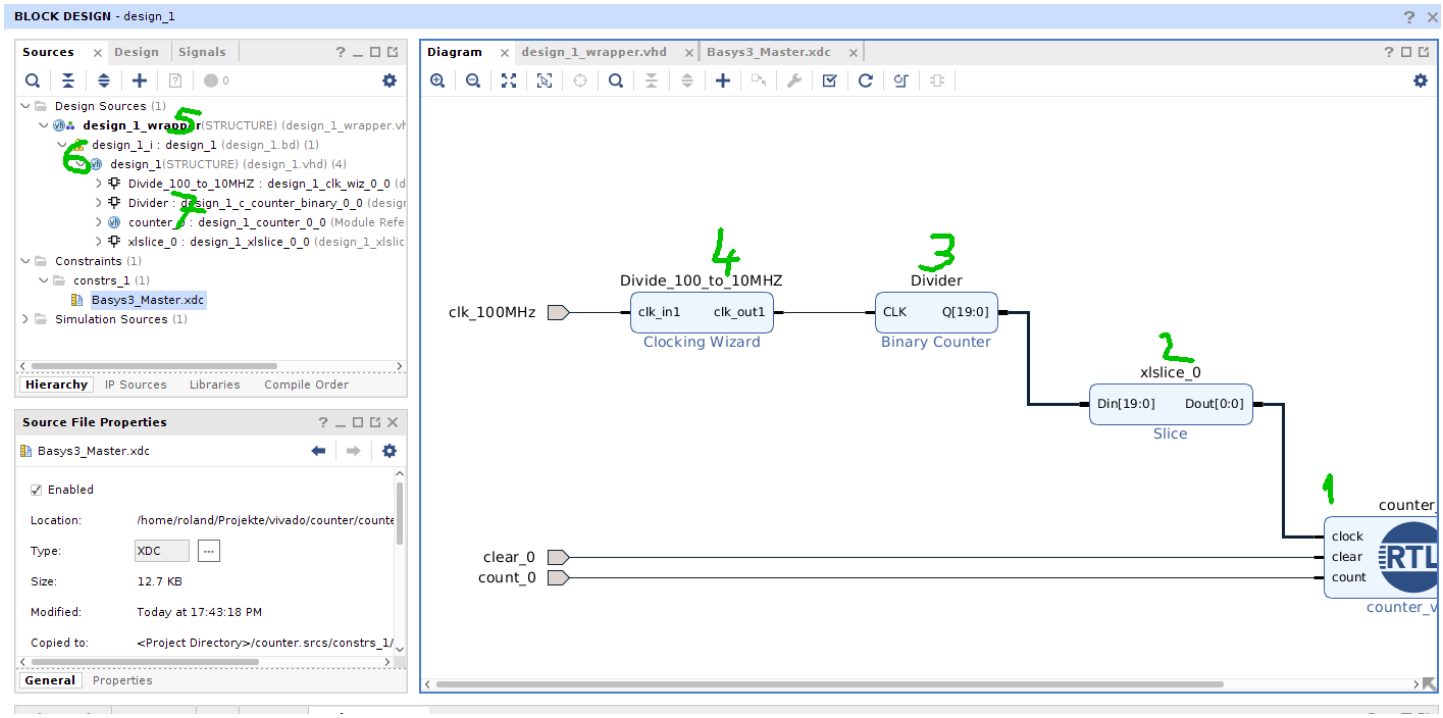
* Hardware-Manager starten und "Autoconnect"

Probleme: mögliche Lösungen

* USB-Kabel / mehrere Kabel ausprobieren

* Hardware-Server als Administrator starten
>gksudo -k -u root /mnt/opt/Xilinx/Vivado/2017.4/bin/hw_server

Block Design



1. Im Blockdesign IP- (4,3,2) und VHDL-Module (1) kombinieren

2. Make External erzeugt die Ports nach Außen

3. DRC Design Rules Check "Validate Design F6"

3. "Create HDL Wrapper" erzeugt einen Wrapper (5) für das Blockdesign (6), der dann implementiert werden kann.

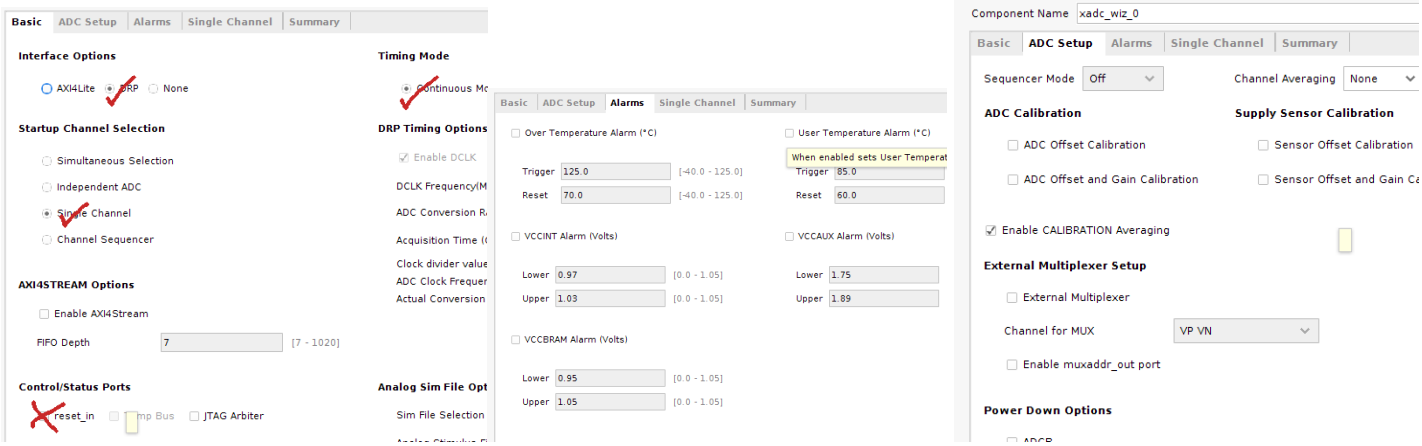
FAQ

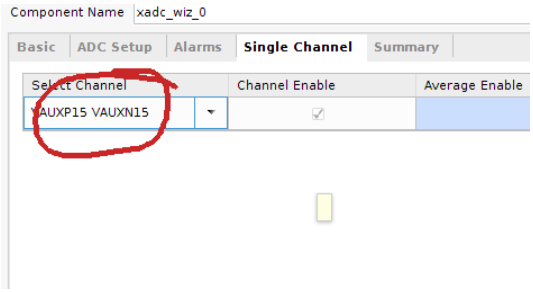
AR# 59355 Vivado IP Flows - How to use one Block Design inside another Block Design?

XADC

https://www.xilinx.com/support/documentation/user_guides/ug480_7Series_XADC.pdf

Einstellung für Single Conversion Unipolar (Input an Stecker JXADC Pin 4 = +Input, darunter -Input);

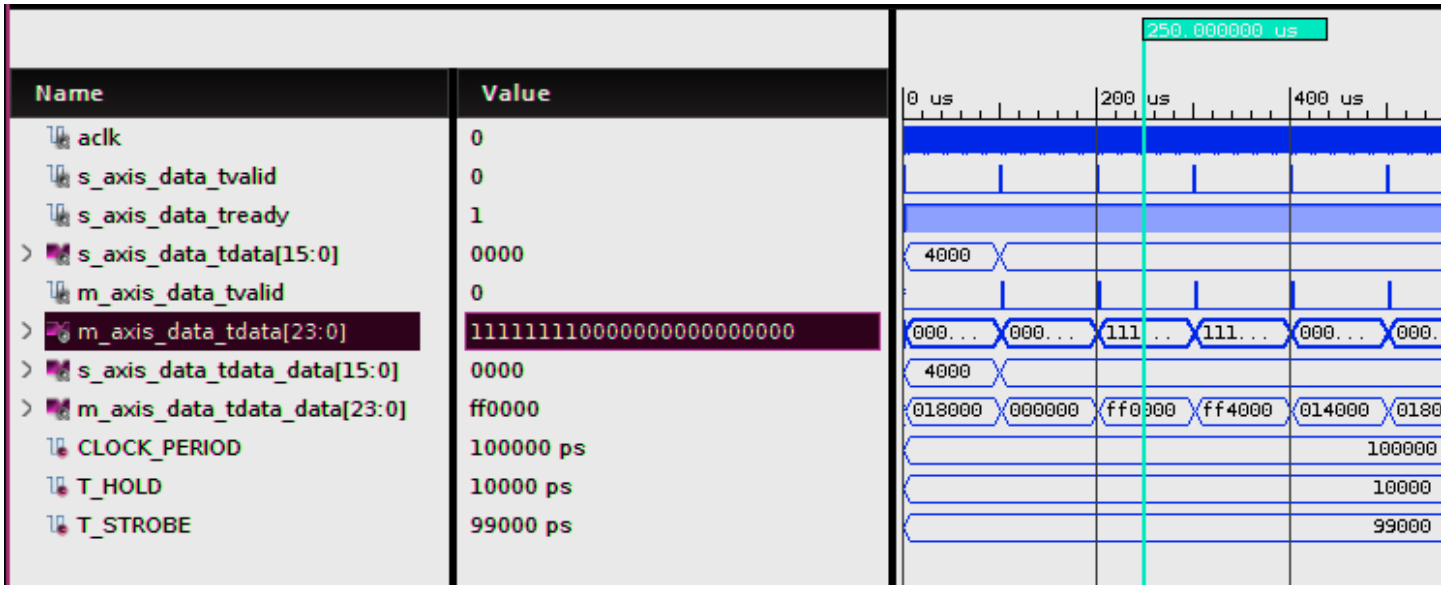




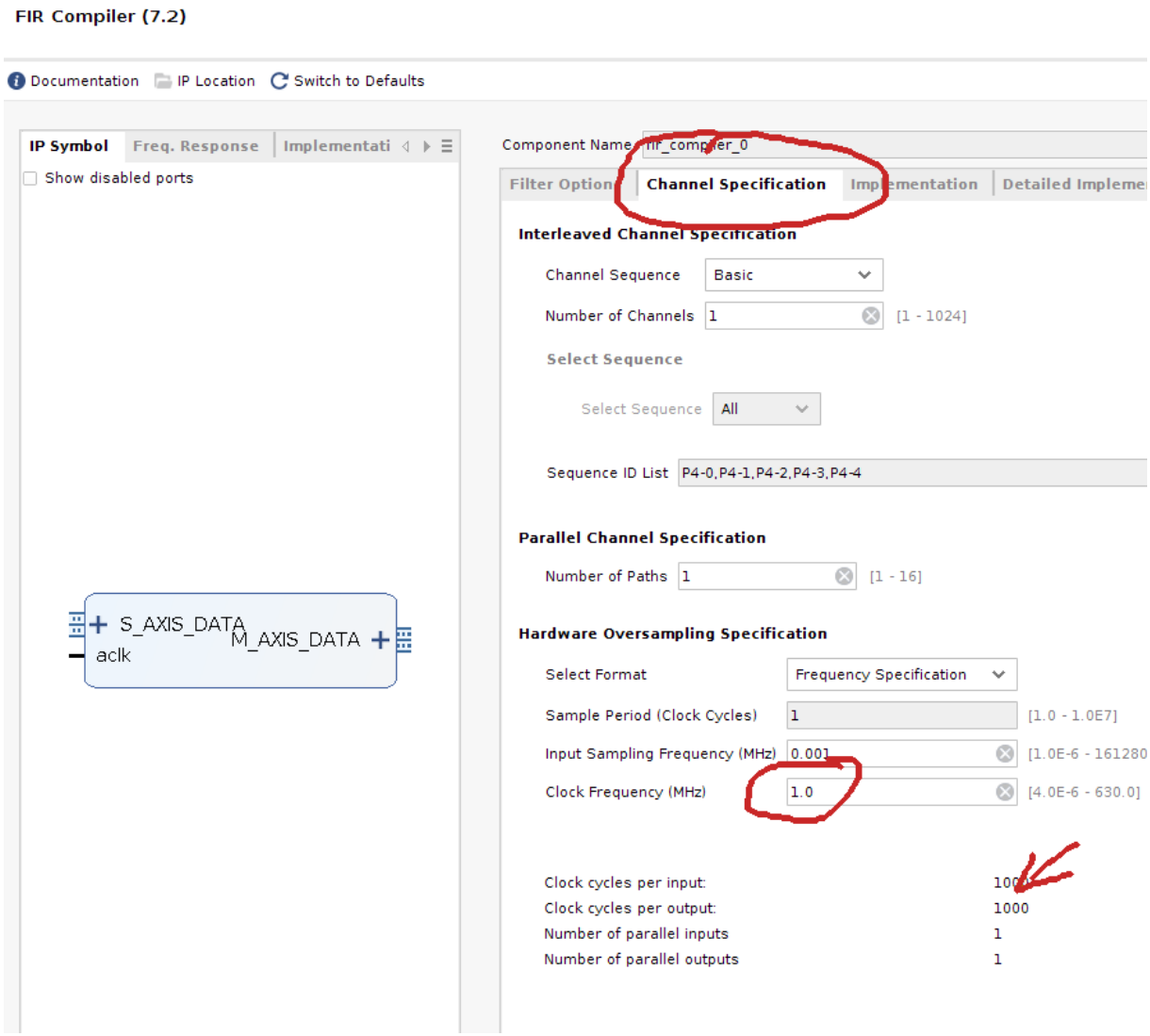
Continuous Sampling … automatische Konvertierung; für die Ausgabe der Daten werden die Pins EOC (end of conversion) mit DEN (… oder so ähnlich, data enable) verbunden, dadurch wird nach jeder Umwandlung das Datenregister ausgegeben.

```
        INIT_40 => X"0000", -- config reg 0 Avaraging 16 Samples/no ext. Mux/
                            Unipolar Mode/continuous sampling/
        INIT_41 => X"210F", -- config reg 1  continuous sequence/no alarms/disable
calibration
        INIT_42 => X"0400", -- config reg 2  ADC Clock = DCLK/4
        INIT_48 => X"0000", -- Sequencer channel selection
        INIT_49 => X"C0C0", -- Sequencer channel selection  VAUXP 6/7 14/15
        INIT_4A => X"0000", -- Sequencer Average selection
        INIT_4B => X"0000", -- Sequencer Average selection
        INIT_4C => X"0000", -- Sequencer Bipolar selection
        INIT_4D => X"0000", -- Sequencer Bipolar selection
        INIT_4E => X"0000", -- Sequencer Acq time selection
        INIT_4F => X"0000", -- Sequencer Acq time selection
        INIT_50 => X"B5ED", -- Temp alarm trigger
        INIT_51 => X"57E4", -- Vccint upper alarm limit
        INIT_52 => X"A147", -- Vccaux upper alarm limit
        INIT_53 => X"CA33",  -- Temp alarm OT upper
        INIT_54 => X"A93A", -- Temp alarm reset
        INIT_55 => X"52C6", -- Vccint lower alarm limit
        INIT_56 => X"9555", -- Vccaux lower alarm limit
        INIT_57 => X"AE4E",  -- Temp alarm OT reset
        INIT_58 => X"5999",  -- Vccbram upper alarm limit
        INIT_5C => X"5111",  -- Vccbram lower alarm limit
```

FIR



hier sieht man: als Impulsantwort erhält man nach die Koeffizienten des Filters 24 Bit 2er Komplement



Wenn hier die 300MHz eingetragen sind dauert es 300000 Zyklen, bis der nächste Wert verarbeitet wird.

Direct digital synthesizers (DDS)

Phasengenerator + Lookuptable

PINC … Phaseninkrement

POFF … Phasenoffset

Modi: PINC/POFF

- fixed, programmed at compile time
- programmable using CONFIG channel‘s TDATA subfields
- streaming using input PHASE channel with optional RESYNC

Rasterized Mode: der Einheitskreis 360* wird in N Teile geteilt (modulus 0 – N-1)
z.B: N=100 für einen Winkel von 90° ist der Wert φ = 25, für -90°=270°=> φ = 75

im Rasterized Mode wird für jedes Phasenargument auch eine Stützstelle in der LUT zur Verfügung gestellt, die Phase braucht daher nicht nachträglich quantisiert zu werden; nur möglich, wenn die gewünschte Ausgangsfrequenz ein rationales Verhältnis zum Systemtakt hat fout = clk*m/n; man kann auf Dithering bzw. Taylor Korrektur verzichten

Systemparameter: PINC/POFF werden berechnet

SIN/COS LUT

die Phase wird über einen externen Zähler an die LUT angelegt; Taylor Reihen Korrektur erhöht den Störabstand.

Phase Generator and SIN/COS LUT (DDS)

interner Phasengenerator + LUT ; Taylor Reihen Korrektur und Phasen-Dithering optional

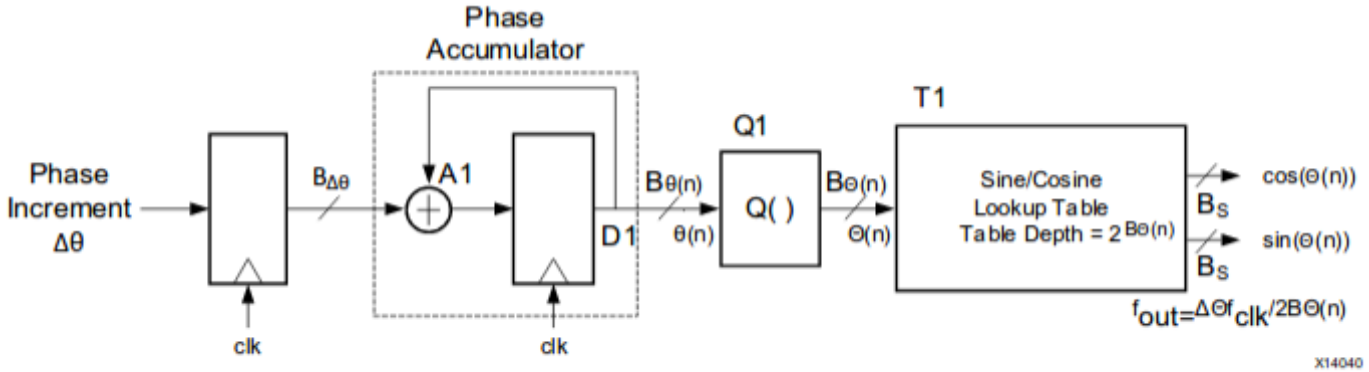


Figure 3-1: **Phase Truncation DDS (Simplified View of the DDS Core)**

Illustration 1: https://www.xilinx.com/support/documentation/ip_documentation/dds_compiler/v6_0/pg141-dds-compiler.pdf
Δθ … hochauflösender Phasenwinkel, ΔΘ … quantisierter Phasenwinkel, $B_{\delta\theta}$ … Bitanzahl
Anzahl der Stützstellen: $N = 2^{B\delta\theta}$

$$\Delta\theta = \frac{f_{out} 2^{B_{\theta(n)}}}{f_{clk}} \quad f_{out} = \frac{f_{clk}\Delta\theta}{2^{B_{\theta(n)}}}$$

Pharsen-Wortbreite $\Delta\theta$ am Eingang

Multichannel

Time-Multiplex

Basic Handshake

1. Daten anlegen => Master sendet TVALID

2. Slave TREADY => Master übernimmt die Daten mit dem nächsten Clock

3. bei Multichannel muss jetzt mit jedem Takt ein neues Datum angeboten werden, diese werden nacheinander eingelesen

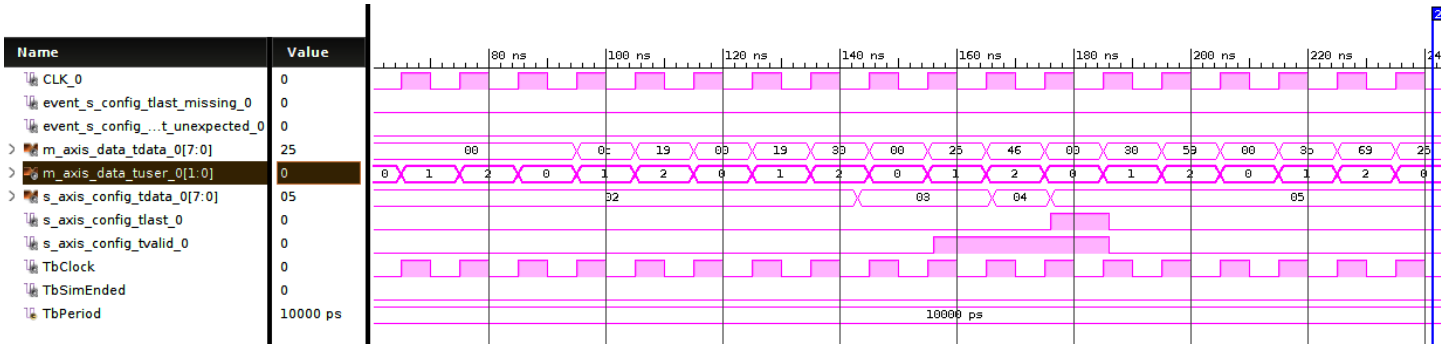
4. ist der letzte Weret eingelesen, dann geht TVALID wieder auf FALSE und der Slave stoppt die Eingabe

Alle Datenpfade können auf TREADY verzichten, CONFIG Kanal aber muss TREADY verwalten.

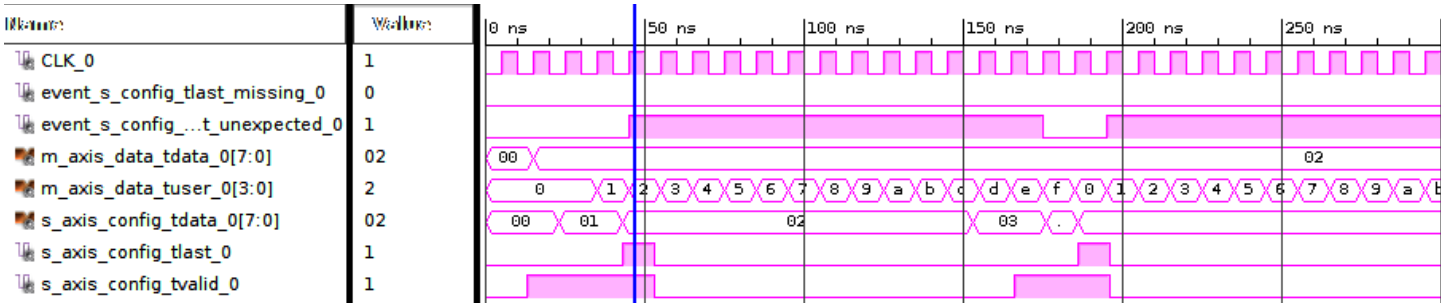
CONFIG Kanal

benötigt ein TLAST, wenn mehrere Kanäle, bei Single Channel gibt es kein TLAST

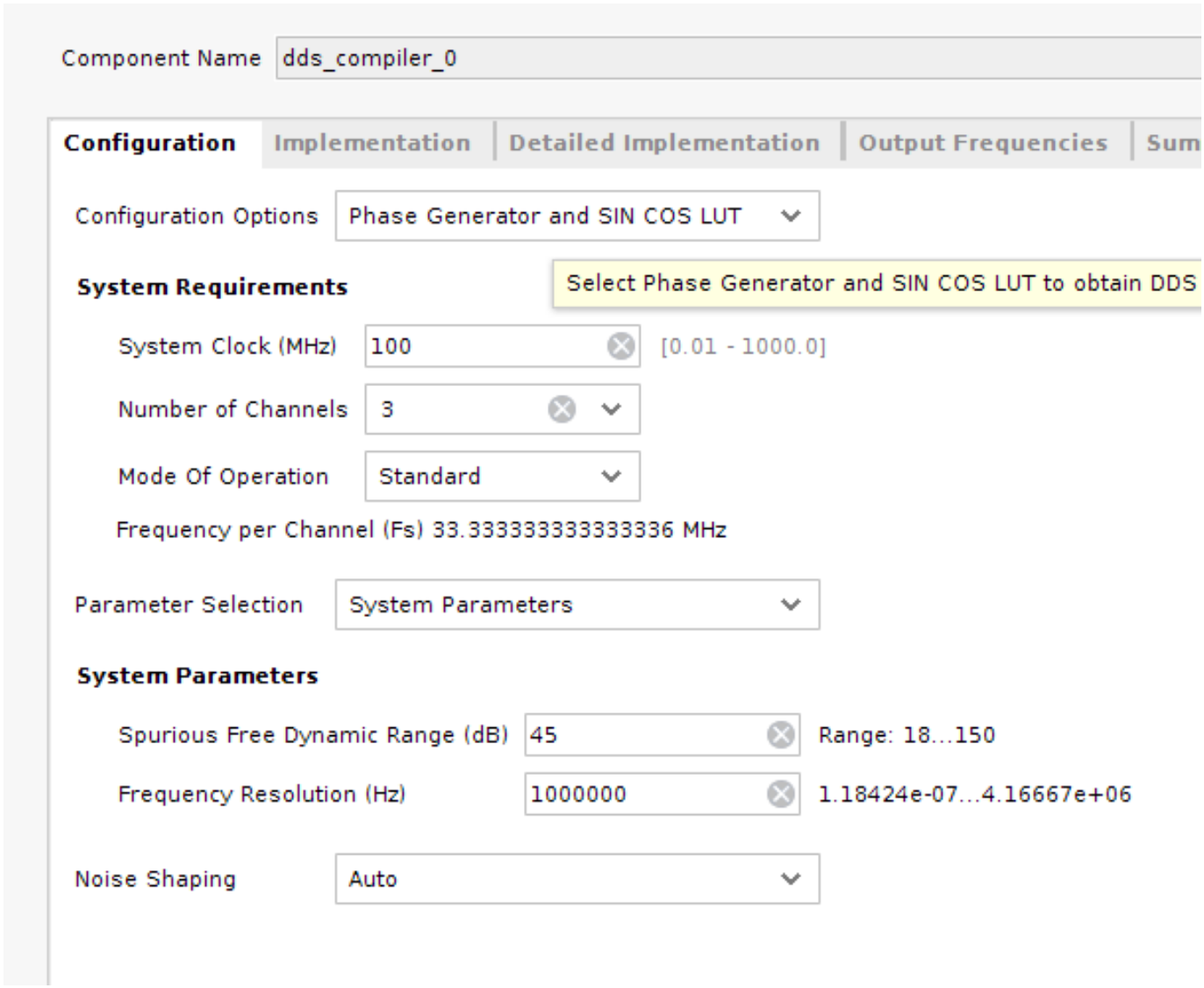
Beispiel 3 Kanal

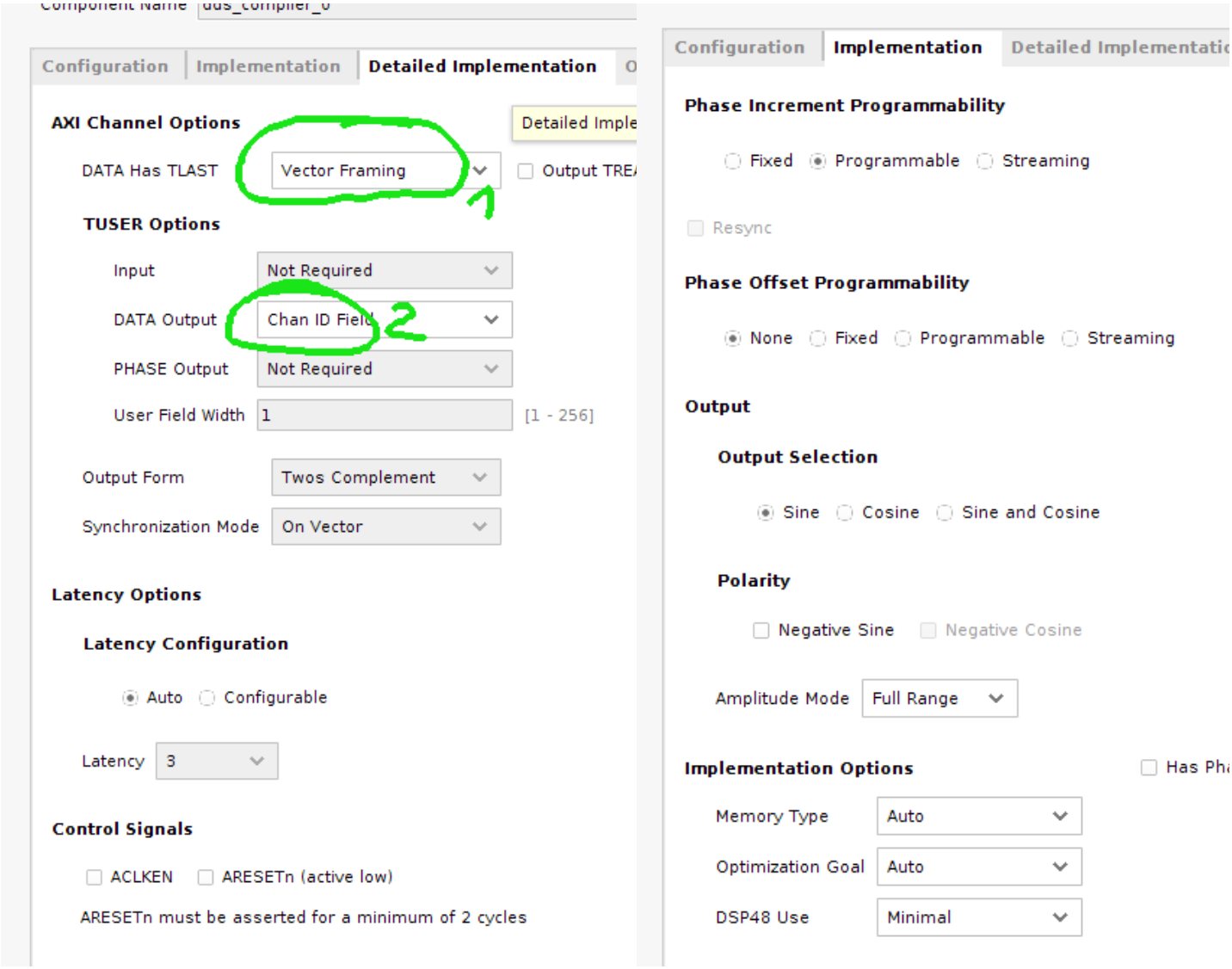


hier ist alles ok



wenn der DDS auf 16 Kanäle programmiert ist, müssen auch 16 Config-Daten übertragen werden. Hier kann man erkennen, wie die gleiche Sequenz wie oben zu einem tlast_unexpected Fehlersignal führt.


Component Name
dds_compiler_0
Configuration
Implementation
Detailed Implementation
Output Frequencies
Sum
Configuration Options
Phase Generator and SIN COS LUT
System Requirements
Select Phase Generator and SIN COS LUT to obtain DDS
System Clock (MHz)
100
[0.01 - 1000.0]
Number of Channels
3
Mode Of Operation
Standard
Frequency per Channel (Fs) 33.333333333333336 MHz
Parameter Selection
System Parameters
System Parameters
Spurious Free Dynamic Range (dB)
45
Range: 18...150
Frequency Resolution (Hz)
1000000
1.18424e-07...4.16667e+06
Noise Shaping
Auto



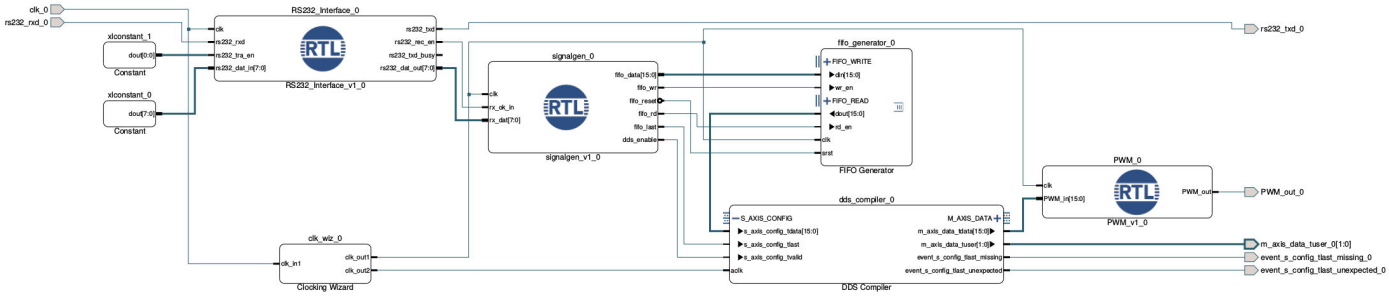
Anmerkung: Chan ID Field (2) gibt die Zusatzinfo aus, welcher Kanal gerade aktiv ist;

RS232 gesteuerter Mehrkanal-Sinusgenerator

Die drei Sinus-Signale werden im Zeitmultiplex vom DDS digital ausgegeben und über einen Sigma-Delta-Umsetzer als PWM Signal ausgegeben.

Serielle Schnittstelle sendet binäre Daten 00 (Start) – 03 (Anzahl der Kanäle) – MSB Kanal1 LSB Kanal1 – MSB Kanal2 LSB Kanal2 – MSB Kanal3 LSB Kanal3

Die Kanaldaten werden als 16bit Wort in ein Fifo abgelegt und dann mit dem richtigen Timing an den DDS übergeben. Der DDS erzeugt ein 3-Kanaliges Sinus-Signal und gibt es über einen Sigma-Delta Wandler analog aus.



Der Clocking-Wizard erzeugt zwei phasenverschobene Taktsignale damit die Übergabe der Daten an den DDS und an den DAC funktioniert. Achtung! DDS verwemdet 2er Komplement Darstellung, diese wird im PWM Modul in Offset Binary umgewandelt (durch Addition von 0x0fff).

Ansteuerung (Ubuntu Linux)

Binärfile erstellen

1) binmake installieren

```
$ git clone https://github.com/dadadel/binmake
$ cd binmake
$ make
```

2) Textdatei erstellen

https://stackoverflow.com/questions/8521240/create-binary-file-in-bash#8831573

3) Konvertieren

```
$ ./binmake file.txt file.bin
```

Umstellen der Baudrate

> stty -F /dev/ttyUSB1 57600

Senden des Binärfiles

> cat file.bin >/dev/ttyUSB1

Probleme (Version v6)

+ jeder Kanal arbeitet mit Amplude 1, sollte anpassbar sein (es wird vom DDS eine Kanalnummer ausgegeben; diese kann verwendet werden, um den einzelnen Kanal herauszufiltern)

+ die Qualität des Signals muss überprüft werden; es handelt sich um einen sehr einfachen SigmaDelta Wandler. Bei der Ausgabe von Signalen θ=0 (Gleichspannung) entstehen hochfrequente Artefakte.

+ um digitale Filter testen zu können müssen die einzelnen Kanäle addiert werden d.h. die Kanaldaten separiert, zeitversetzt werden bevor sie addiert werden können.

+ sollte das zu aufwändig sein könnte man 3 separate DDS Blöcke verwenden und anschließend addieren.