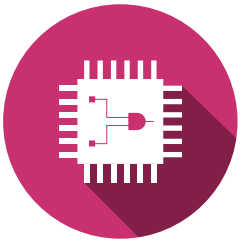


Funktionsgenerator

Inhaltsverzeichnis

1 Einreichen **1**

2 Start **2**

3 Ziele **2**

3.1 Sägezahn . . . 2

3.1.1 Todo . . . 3

3.2 Rechteck . . . 3

3.2.1 Todo . . . 3

3.3 Dreieck . . . 3

3.3.1 Todo . . . 4

3.4 Polygon . . . 4

3.4.1 Todo . . . 5

3.5 Sinus . . . 5

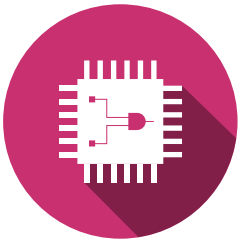
3.5.1 Todo . . . 6

1 Einreichen

In diesem Labor wird das Schreiben von VHDL-Code für kombinatorische Blöcke und synchrone sequentielle Blöcke anhand der Realisierung eines digitalen Funktionsgenerators vorgestellt. Der Generator erstellt :

- ein Sägezahnsignal (**sawtooth**)
- ein Rechtecksignal (**square**)
- ein Dreiecksignal (**triangle**)
- ein Polygonsignal (**polygon**)
- ein Sinussignal (**sine**)

Signale werden als Zahlen codiert (von Typ **positive**) von **bitNb** bits (auf 16 bits festgelegt).



2 Start

Das Schema befindet sich in der Bibliothek **WaveformGenerator**. Der Teststand ist unter **WaveformGenerator_test**.

*Zur Erinnerung: Das Modellierungsprogramm muss über die Datei **sineGen.bat** gestartet werden.*

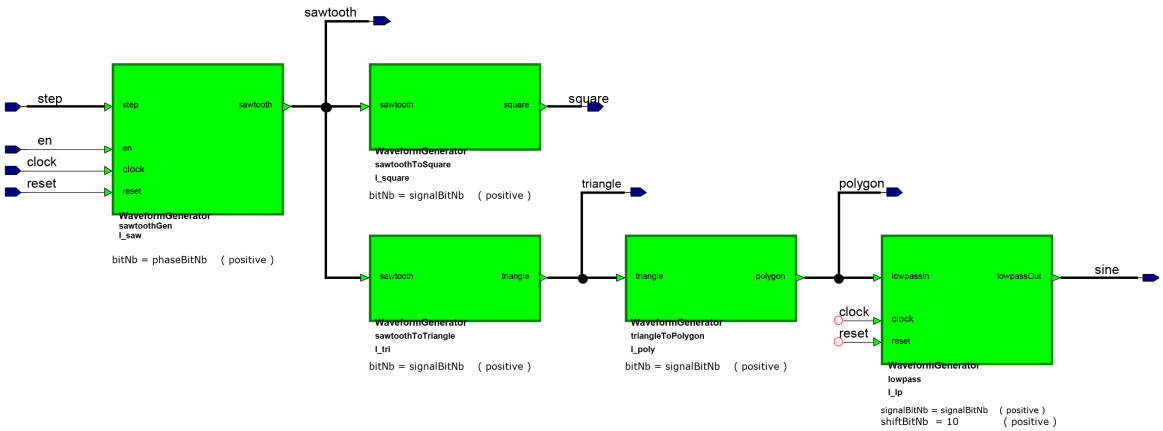


Abbildung 1: Funktionsgenerator

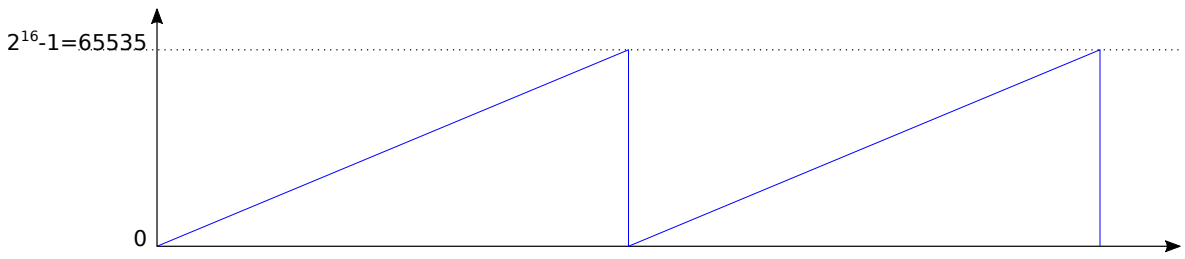
3 Ziele

3.1 Sägezahn

Ein Sägezahnsignal wird mithilfe eines Zählers erzeugt, der sich im Kreis dreht. Mit einer Zahl am Eingang (**step**) kann die Frequenz des Signals gesteuert werden (je größer die Zahl, desto schneller läuft der Zähler über und damit ist die Frequenz groß). Bei jeder steigenden Flanke des Takts, wenn **en = '1'**, addiert der Zähler den Wert von **step** zu seinem aktuellen Wert. Je größer also der Eingabewert ist, desto schneller dreht sich der Zähler.

Dieser Block enthält einen generischen Parameter (**generic**): **bitNb**. Dieser Parameter wird sowohl für die Entität als auch für die zu schreibende Architektur festgelegt.

*Das Ausgangssignal **sawtooth** kann nur zugewiesen werden. Es kann nicht gelesen werden, sodass es auch nicht verwendet werden kann, um den Wert von **step** zu addieren. Sie müssen also ein internes Signal definieren, das gelesen und zugewiesen werden kann, und dieses Signal kombinatorisch auf den Ausgangsport kopieren.*



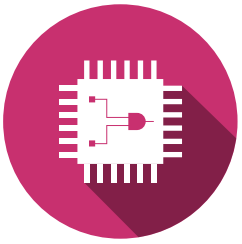


Abbildung 2: Sägezahnsignal

3.1.1 Todo

1. Schreiben Sie die VHDL-Architektur des Sägezahnsignalgenerators, wobei Sie wissen, dass diese Schaltung sequentiell und synchron ist.
2. Untersuchen Sie die Entität des Sägezahnsignalgenerators auf die Arten der Ein-/Ausgangssignale. Durchsuchen Sie die Definition des Pakets **numeric_std**, um sich ein Bild von den Funktionen zu machen, die für Signale vom Typ **unsigned** zur Verfügung stehen.
3. Schätzen Sie die Periode des Sägezahnsignals für ein Taktsignal von 66 MHz, einen 16-Bit-Zähler und einen **step**-Wert gleich 8.
4. Geben Sie die allgemeine Formel für die Frequenz des erzeugten Signals in Abhängigkeit von der Taktfrequenz, der Anzahl der Bits des Zählers und dem Wert der Schrittweite an.
5. Kompilieren und simulieren Sie den Block **waveformGen_tb**. Überprüfen Sie, ob der Sägezahnsignalgenerator ordnungsgemäß funktioniert.

3.2 Rechteck

Die Umwandlung des Sägezahnsignals in ein Rechtecksignal erfolgt, indem nur das höchstwertige Bit des Sägezahnsignals betrachtet wird. Das erzeugte Signal wird ebenfalls mit 16 Bits codiert.

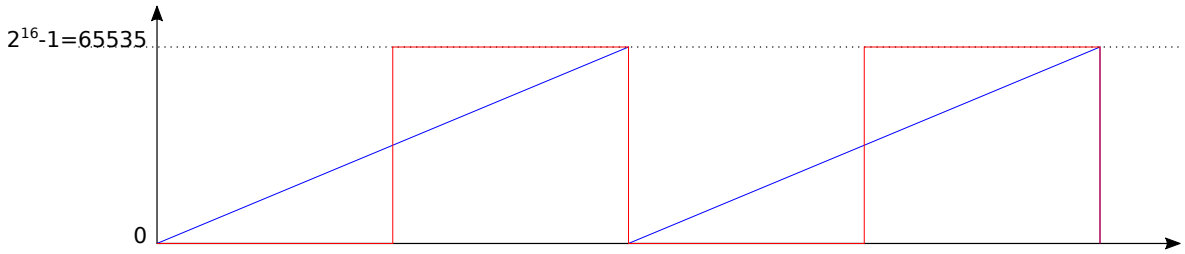


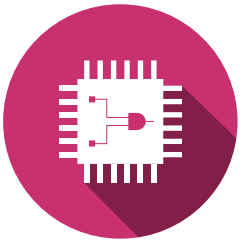
Abbildung 3: Rechtecksignal

3.2.1 Todo

1. Bestimmen Sie die binäre Darstellung der kleinsten ganzen Zahl, die dem unteren Wert des Rechtecksignals entsprechen wird. Bestimmen Sie auch die binäre Darstellung des oberen Wertes des Rechtecksignals.
2. Schreiben Sie die VHDL-Architektur des Blocks zur Umwandlung des Sägezahnsignals in ein Rechtecksignal, wobei zu beachten ist, dass die Schaltung kombinatorisch ist. Die Ausgabe ist ein **unsigned** mit 16 Bit.
3. Kompilieren und simulieren Sie den Block **waveformGen_tb**. Überprüfen Sie die Form des Rechtecksignals.

3.3 Dreieck

Die Umwandlung des Sägezahnsignals in ein Dreieckssignal erfolgt durch Falten des Sägezahns nach unten", wenn das höchstwertige Bit des Sägezahnsignals auf '1' gesetzt wird. Diese Schal-



tung ist kombinatorisch. Wenn das höchstwertige Bit des Sägezahnsignals auf '1' ist, invertieren Sie alle Bits des Sägezahnsignals. Da das so erzeugte Signal sein höchstwertiges Bit immer auf '0' haben wird, nutzt es nur die Hälfte des zur Verfügung stehenden Bereichs. So muss das resultierende Signal mit zwei multipliziert werden, was einer Linksverschiebung des binären Ausgabewerts entspricht.

Es wird vorgeschlagen, ein internes Signal zu definieren, das aus dem ersten Faltungsvorgang resultiert und zur Erzeugung der Ausgabe verwendet wird. Es ist auch übersichtlicher, ein weiteres internes Signal zu definieren: das höchstwertige Bit des Eingangssignals.

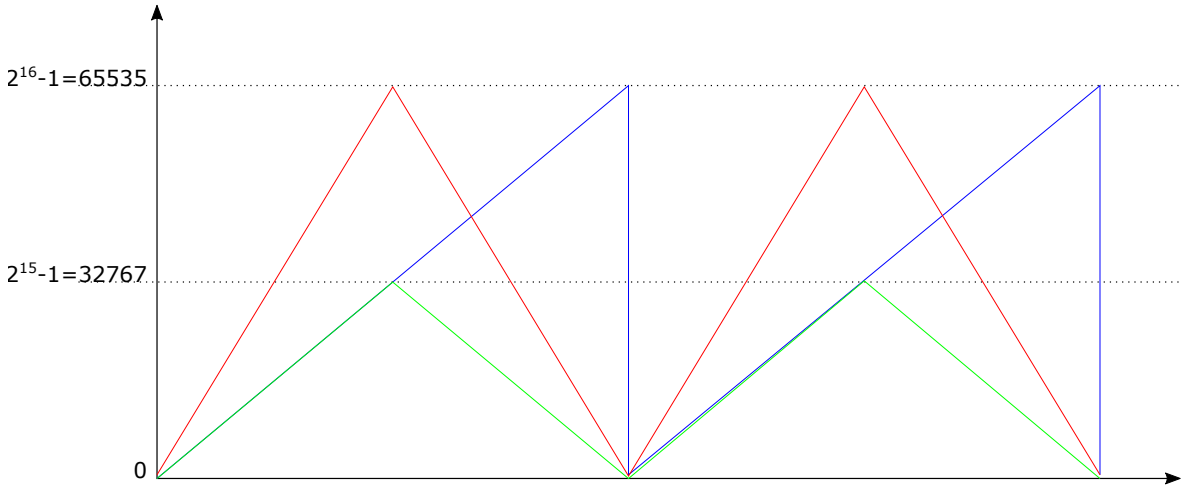


Abbildung 4: Dreiecksignal

3.3.1 Todo

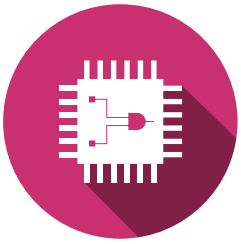
1. Schreiben Sie die VHDL-Architektur des Blocks zur Umwandlung des Sägezahnsignals in ein Dreieckssignal.
2. Kompilieren und simulieren Sie den Block **waveformGen_tb**. Überprüfen Sie die Form des Dreiecksignal.

3.4 Polygon

Die Transformation von Dreieck zu Sinus kann durch eine Wahrheitstabelle (ROM) erfolgen, die alle diese Werte enthält. Diese Lösung ist zwar die genaueste, aber auch sehr materialintensiv.

Hier wird das Dreieckssignal so gefiltert, dass nur der Grundton erhalten bleibt (d.h. der Sinus). Um die Filteraufgabe zu vereinfachen, wird das dreieckige Signal in ein Polygonsignal umgewandelt, dessen Form dem gewünschten Sinus näher kommt :

- Die Umwandlung von Dreieck zu Polygon beginnt mit der Multiplikation des Dreiecks mit 1,5.
- Danach ersetzt man alle Werte, die kleiner als 1/8 sind, in der Skala des resultierenden Dreiecks durch diesen 1/8-Wert: was darunter liegt, wird abgeschnitten. Ebenso wird alles, was über 5/8 hinausgeht, abgeschnitten.
- Schließlich können wir das Signal um 1/8 nach unten verschieben.



Die Multiplikation mit 1,5 erfolgt durch eine Verschiebung und eine Addition. Das Ergebnis der Multiplikation benötigt ein Bit mehr als das Ausgangssignal. Nach der Abwärtsverschiebung kann man das höchstwertige Bit ignorieren und zur gleichen Anzahl von Bits wie bei der Eingabe zurückkehren.

Die Werte 1/8 und 5/8 beziehen sich auf die mögliche Vollskala (Full Scale, FS) des Signals und nicht auf die maximale Amplitude des Dreiecks.

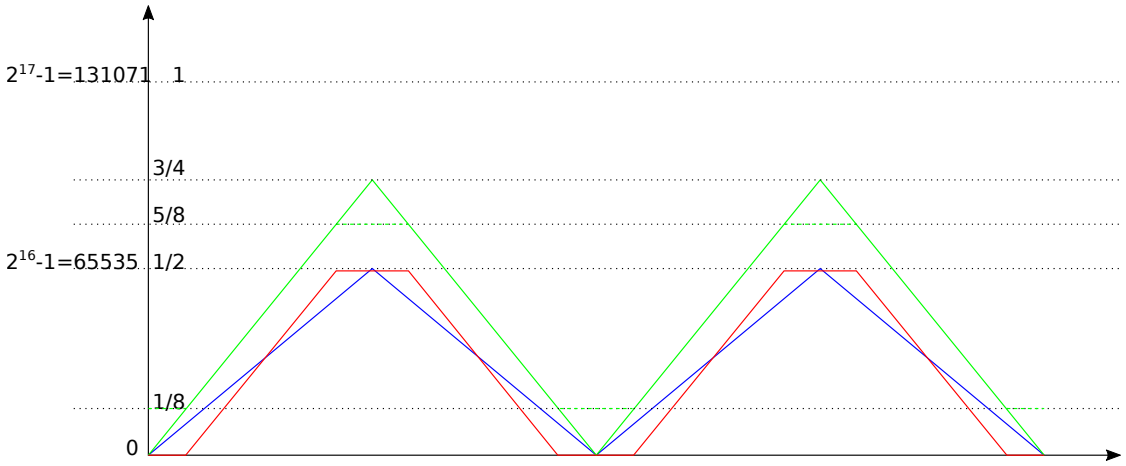


Abbildung 5: Polygonsignal

3.4.1 Todo

1. Schreiben Sie die VHDL-Architektur des Blocks, der das dreieckige Signal in ein Polygonsignal umwandelt.
2. Kompilieren und simulieren Sie den Block **waveformGen_tb**. Überprüfen Sie die Form des Polygonsignal.

3.5 Sinus

Die Tiefpassfilterung erfolgt mithilfe eines Integrators, der durch einen Akkumulator realisiert wird :

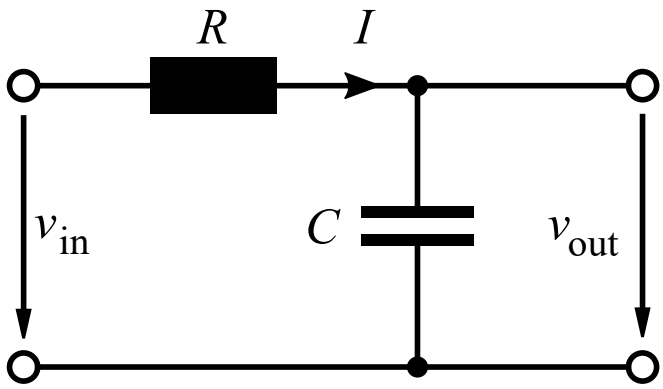
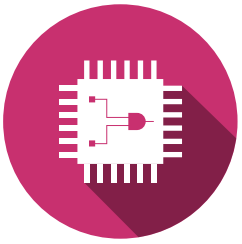


Abbildung 6: RC-Filter

Die folgenden Gleichungen regieren das System :



- $$Vout = \frac{1}{C} * \int Idt \tag{1}$$
- $$I = \frac{Vin - Vout}{R} \tag{2}$$

Ergibt die endgültige Gleichung

$$Vout = \frac{1}{R * C} * \int (Vin - Vout)dt \tag{3}$$

Man **integriert/akkumuliert** also das Spannungsdelta, indem man einen Faktor darauf anwendet.

In der digitalen Welt, in jeder Taktperiode wird der Eingangswert zum Inhalt des Akkumulators addiert (Integration) und der um eine bestimmte Anzahl von Bits verschobene Wert des Akkumulators von ihnen abgezogen. Die Anzahl der Verschiebebits ergibt die Grenzfrequenz: bei jeder Erhöhung dieser Zahl wird die Grenzfrequenz durch 2 geteilt. Der Akkumulator benötigt eine Anzahl von Bits, die der Anzahl des Eingangssignals plus der Anzahl der Offset-Bits entspricht. Bei der Ausgabe müssen Sie nur die höchstwertigen Bits aus dem Akkumulator übernehmen.

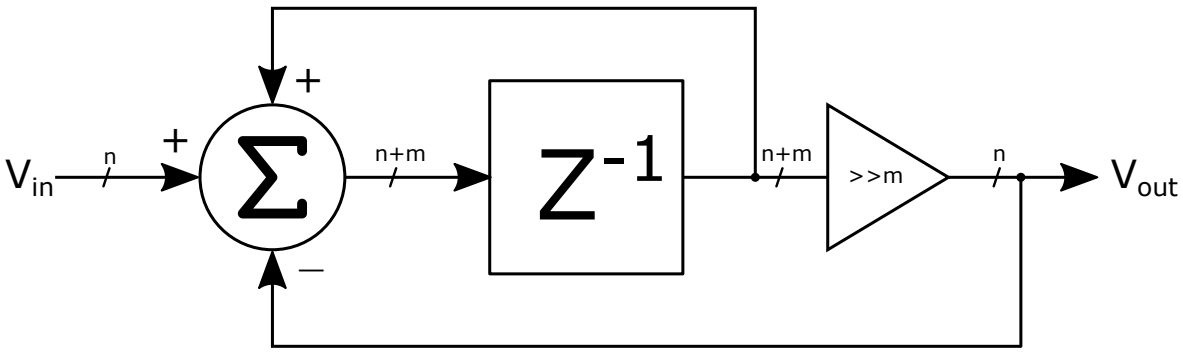


Abbildung 7: Digitalfilter

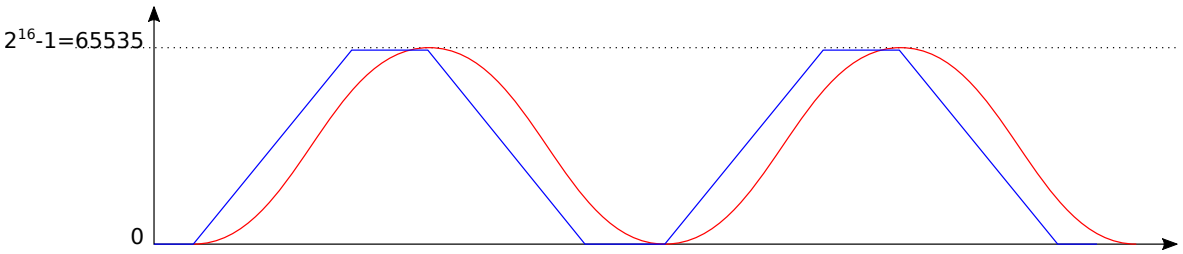
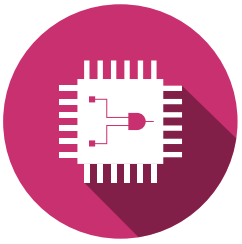


Abbildung 8: Sinussignal

3.5.1 Todo

1. Schreiben Sie die VHDL-Architektur des Tiefpassfilters erster Ordnung.
2. Kompilieren und simulieren Sie den Block **waveformGen_tb**. Überprüfen Sie die Form des Sinusignal.



3. Ändern Sie die Anzahl der Bits, um die der Tiefpassfilter verschoben wird, bis Sie ein ßufriedenstellendesSSinussignal erhalten.