Vorlesung 12

Thema dieser Vorlesung sind einige Komponenten für digitale Datenübertragung

- Serialisierer
- Frequenzteiler
- PLL
- Oszillator
- Leitungscode
- Taktrückgewinnung

Digitale Datenübertragung ist ein breites Thema. Wir werden sie dieser Vorlesung nicht systematisch und vollständig behandeln, sondern nur einige Ideen und Schaltungsblöcke vorstellen, die besonders für die Datenübertragung zwischen einem ASIC und einer FPGA auf einem Board geeignet sind.

Die Datenübertragung zwischen FPGA und Rechner wird normalerweise mithilfe von kommerziellen Designblöcken und Komponenten gemacht. Die ASIC Komponenten müssen oft selbst gemacht werden.

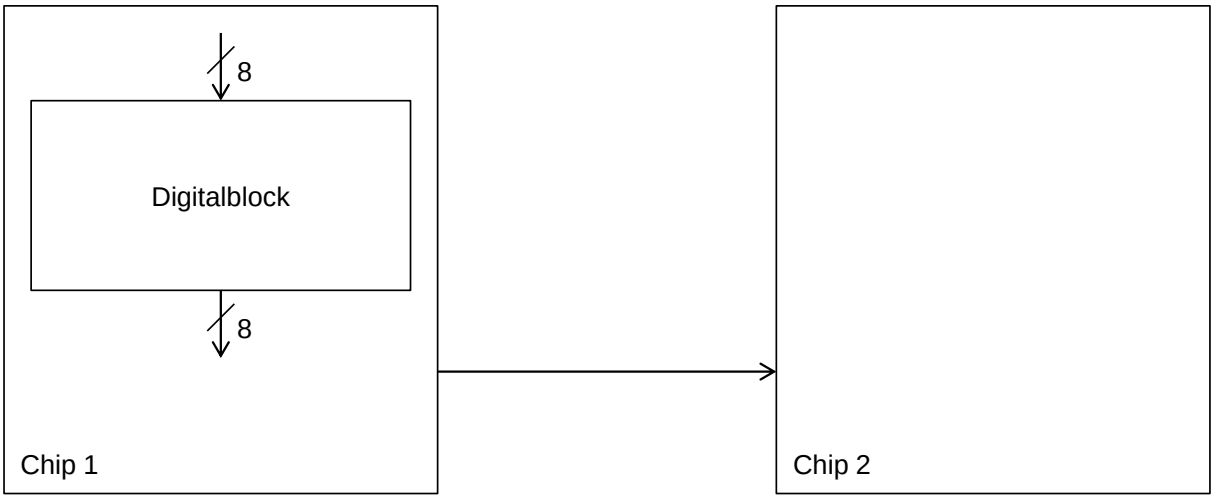

Abbildung 1: Datenübertragung zwischen einem ASIC (Chip1) und FPGA (Chip2)

Wir betrachten ein System (Chip 1 in Abbildung 1) mit einem Digitalblock der die 8-bit Daten bearbeitet. Das Ergebnis soll an einen anderen Chip (oder FPGA) übertragen werden. Dabei soll die Zahl von Leitungen zwischen den Chips minimiert werden.

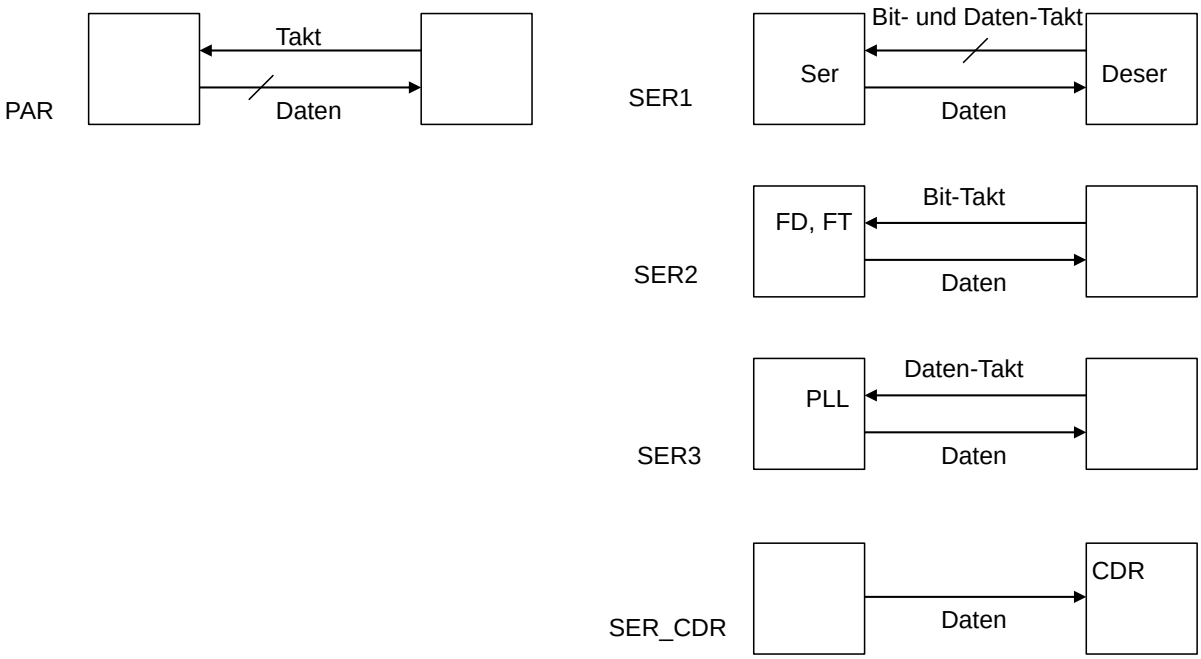

Abbildung 2: Verschiedene Möglichkeiten für Datenübertragung

Für diese Aufgabe gibt es einige Lösungen (Abbildung 2):

PAR: Parallele Datenübertragung

SER1: Serielle Datenübertragung mit Taktübertragung (Bit- und Daten-Takt)

SER2: Serielle Datenübertragung mit Taktübertragung (Bit-Takt)

SER3: Serielle Datenübertragung mit Taktübertragung (Daten-Takt)

SER_CDR: Serielle Datenübertragung mit Taktrückgewinnung (clock data recovery CDR)

Jedes Beispiel benötigt einige Komponenten, die wir beschreiben:

Serialisierer, Deserialisierer, Flankendetektor, Frequenzteiler, PLL, CDR, Schrambler, Leitungcode.

Parallele Datenübertragung

Chip 2 sendet einen Referenz-Taktsignal, z.B. über eine Differenzleitung (Abbildung 3). Die Daten werden vom Chip 1 zum Chip 2 parallel gesendet.

Nehmen wir an, dass der Digitalblock die 8-bit Daten mit 100MHz Takt bearbeitet. Das bedeutet, dass das Ergebnis einmal in 10ns auf dem Parallelausgang auftaucht.

Ethernetkarte (NIC) für den PCI-X-Slot.

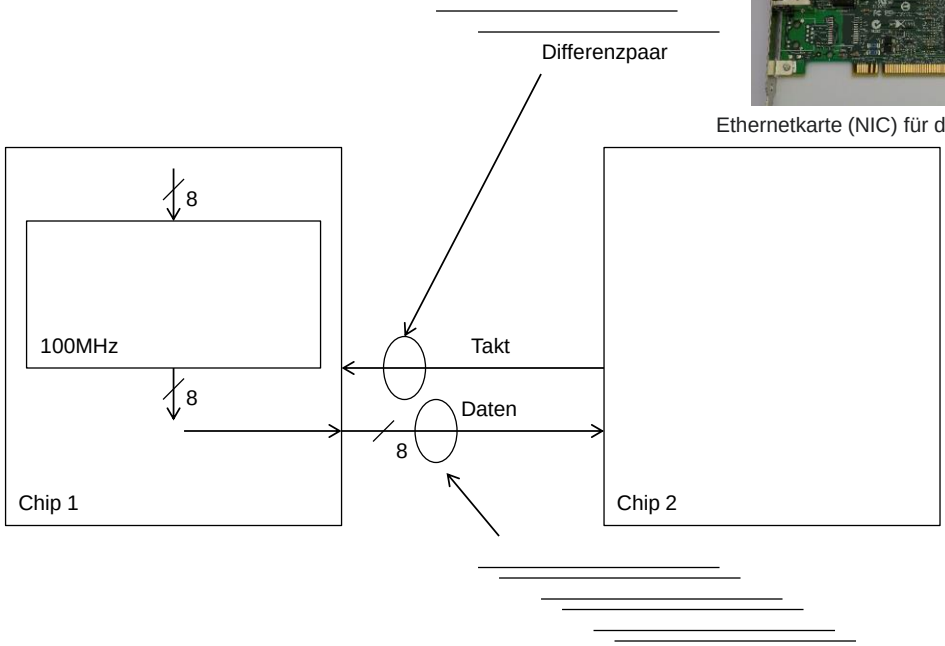

Abbildung 3: Parallele Datenübertragung

Parallele Datenübertragung wird in Computern benutzt:

https://de.wikipedia.org/wiki/Parallele_Schnittstelle

Beispiele: PCI

Parallele Datenübertragung ist oft unpraktisch, da viele Leitungen benötigt werden. Es wären 16 Leitungen, wenn man acht Bits jeweils mit einem Differenzpaar überträgt. Ein weiteres Problem wären die verschiedenen Signalverzögerungen an verschiedenen Leitungen.

Serielle Datenübertragung

Um diese Probleme zu vermeiden werden die Daten oft seriell übertragen (Abbildung 4). In dem Fall brauchen wir für die Daten nur eine Leitung, oder ein Differenzpaar.

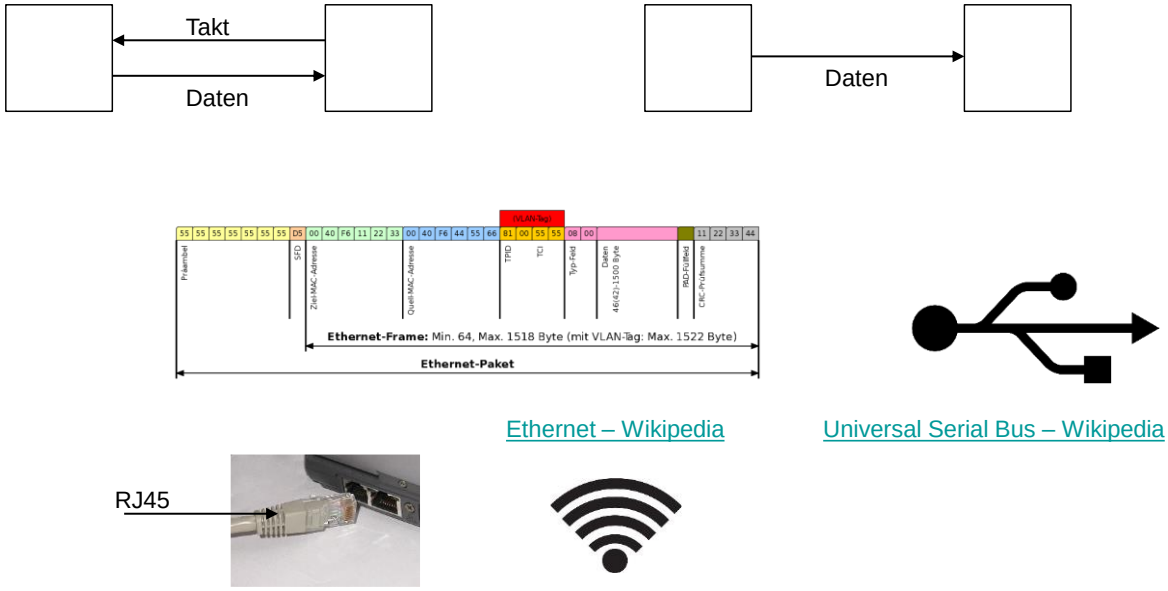

Abbildung 4: Serielle Datenübertragung

Wir werden hier einige Varianten für die serielle Datenübertragung vorstellen. Wir unterteilen sie auf die Varianten mit einer separaten Leitung für die Taktübertragung („synchrones Design“) und die Varianten ohne Taktleitung, wobei Takt im Empfänger aus der Bitsequenz bestimmt wird. Der Takt wird unter anderem für Abtasten vom Eingangssignal benutzt.

Einige Links

https://de.wikipedia.org/wiki/Serielle_Schnittstelle

https://de.wikipedia.org/wiki/Ethernet

https://de.wikipedia.org/wiki/Universal_Serial_Bus

https://de.wikipedia.org/wiki/Leitungscode

https://en.wikipedia.org/wiki/6b/8b_encoding

https://de.wikipedia.org/wiki/6b8b-Code#cite_note-pat687-1

https://de.wikipedia.org/wiki/Controller_Area_Network

Synchrones Design serielle Datenübertragung

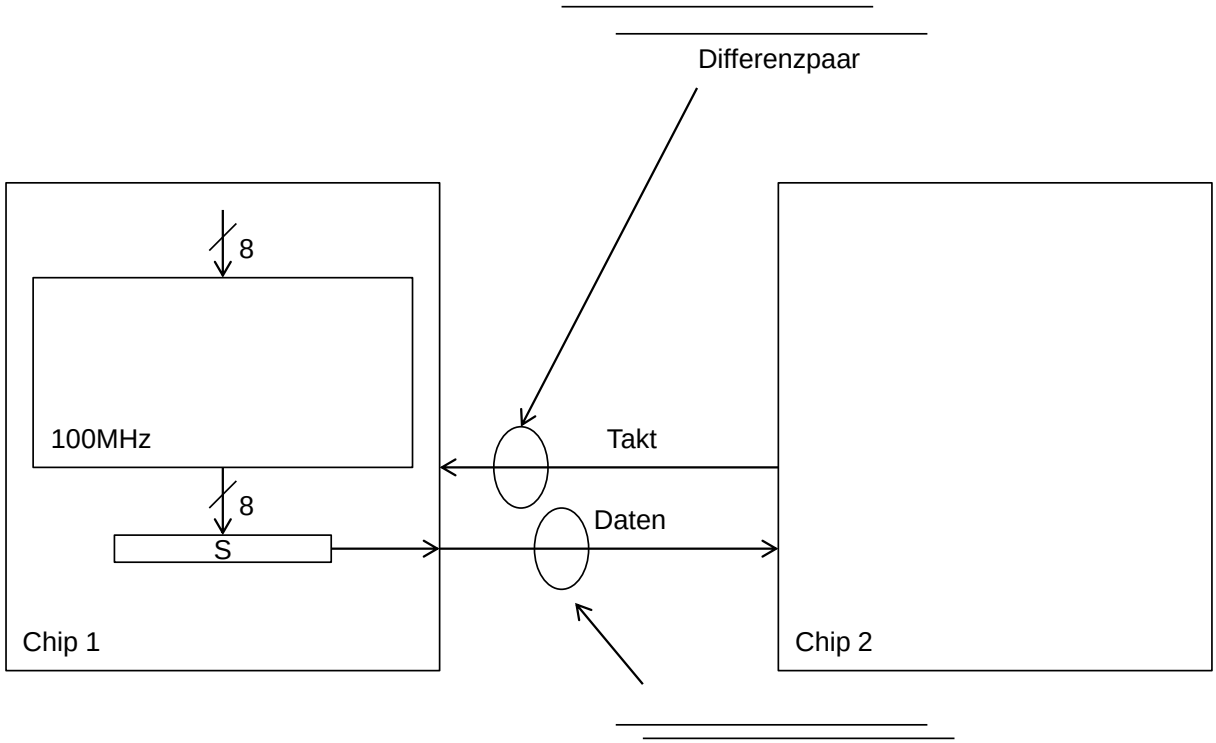

Abbildung 5: Serielle Datenübertragung mit separaten Taktleitung

Betrachten wir das synchrone Design. Es gibt mindestens eine Signalleitung für den Takt (Abbildung 5). Beispiel I2C (Inter-Integrated Circuit). Solche synchrone Übertragung ist nur für lokale Anwendungen geeignet und nicht für Übertragung über große Distanzen.

Die Daten müssen am Chip 1 vom parallelen in serielles Format umgewandelt werden. Das wird mit einem Serialisierer erreicht. Einen Serialisierer realisiert man am einfachsten mit einem Schieberegister, Abbildung 6. Das Schieberegister hat die Möglichkeit die Daten parallel zu laden.

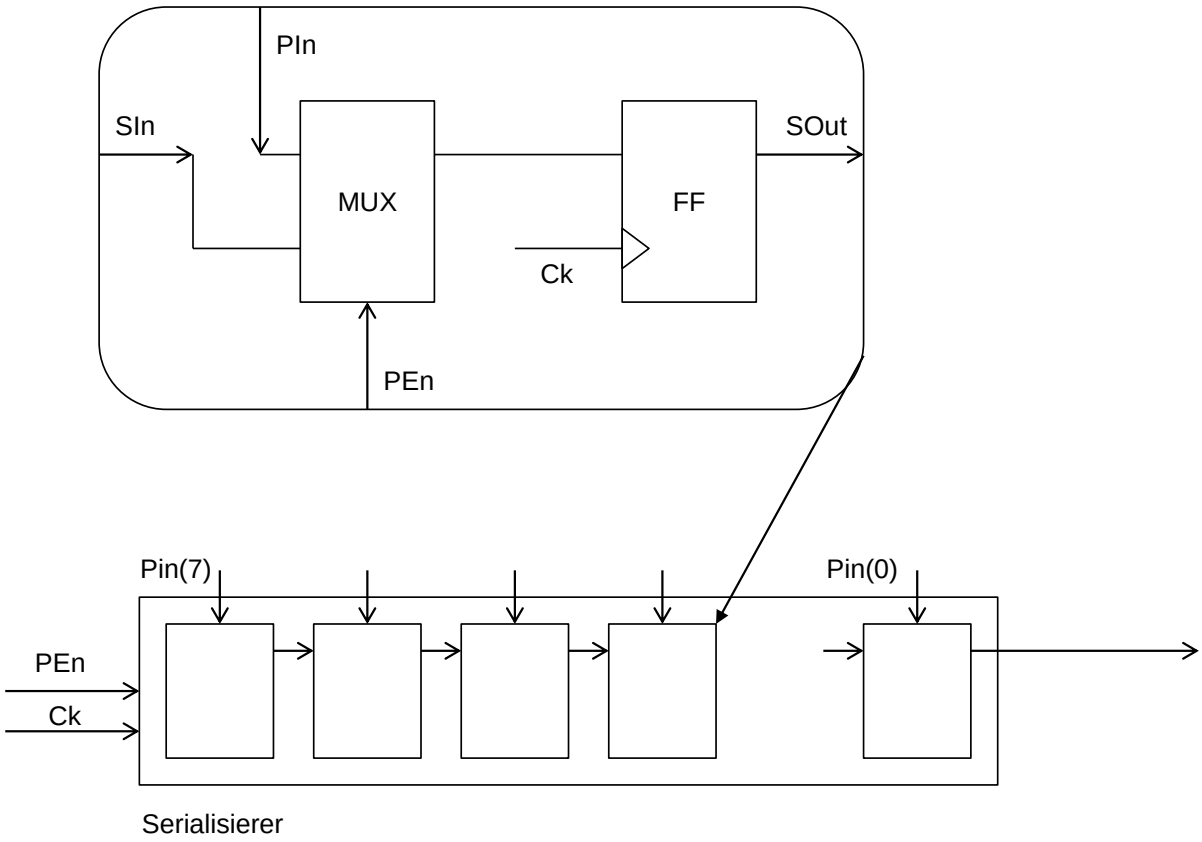

Abbildung 6: Serialisierer

Abbildung 6 zeigt ebenfalls die Implementierung eines Bits im Serialisierer. Wenn das Signal PEn Eins ist, selektiert der Multiplexer (MUX) den Paralleleingang. Also, bei PEn = 1 und auf die steigende Taktflanke wird eine 8-Bit Zahl ins Register geladen. Auf PEn = 0 werden die Daten bitweise zum Ausgang geschoben. PEn soll genau ein Takt lang Eins sein.

Verilog Code der Serialisierers

```
reg[7:0] Par2Ser;

reg[7:0] ParIn;

wire SerOut;

assign SerOut = Par2Ser[0];

always@(posedge clock) begin

        if(PEn) Par2Ser <= ParIn;

        else Par2Ser[7:0] <= {1'b0, Par2Ser[7:1]};

end
```

Wenn der Digitalteil mit 100MHz getaktet wird (100MHz Takt nennen wir Datentakt, langsamer Takt oder slow clock) muss der Serialisierer mit 800MHz getaktet werden (800MHz Bit-Takt, schneller Takt, fast clock). Abbildung 7 zeigt das Zeitdiagramm.

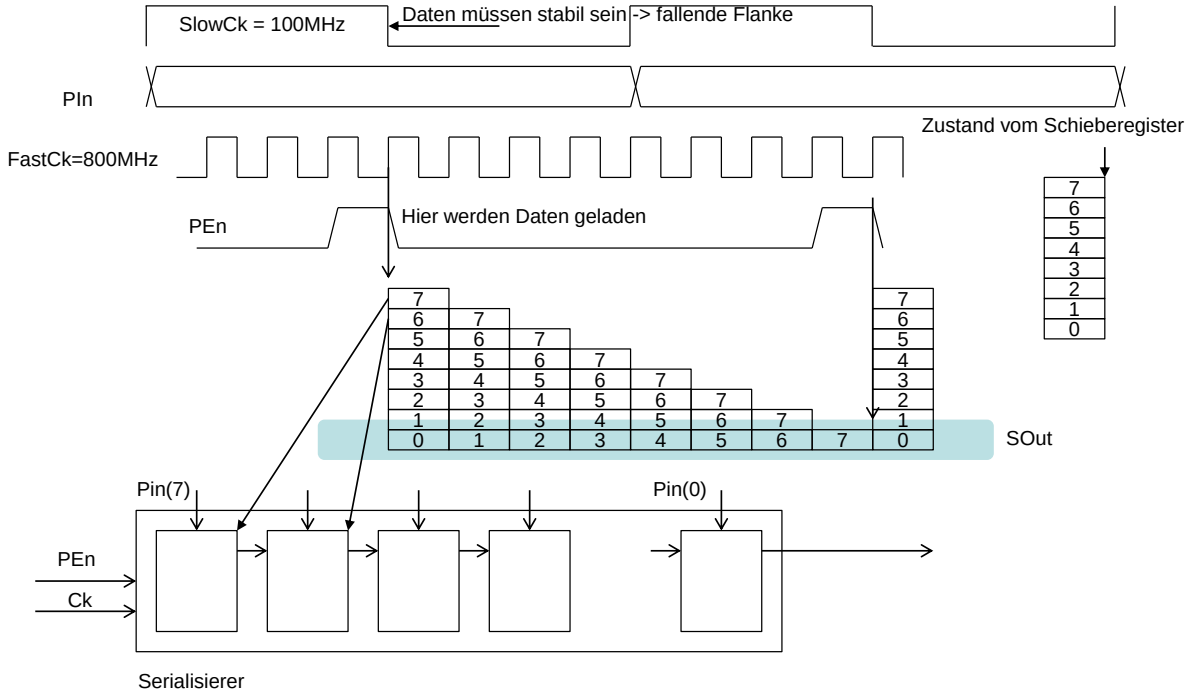

Abbildung 7: Signalverlauf im Serialisierer

Das Signal PEn sollte in diesem Moment aktiv werden, wo die Daten am Paralleleingang stabil sind. Eine Möglichkeit ist es PEn = 1 um die Fallende Datentakt-Flanke zu positionieren, wie in Fig zu sehen ist.

Takt Übertragung beim synchronen Design

Wir werden nun drei Möglichkeiten für die Takterzeugung und Übertragung vorstellen.

Am einfachsten wäre es, dass der Chip 2 sowohl Daten- als auch Bit-Takt sendet (Abbildung 8).

Wenn wir eine oder beide Taktleitungen sparen möchten gibt es folgende Lösungen:

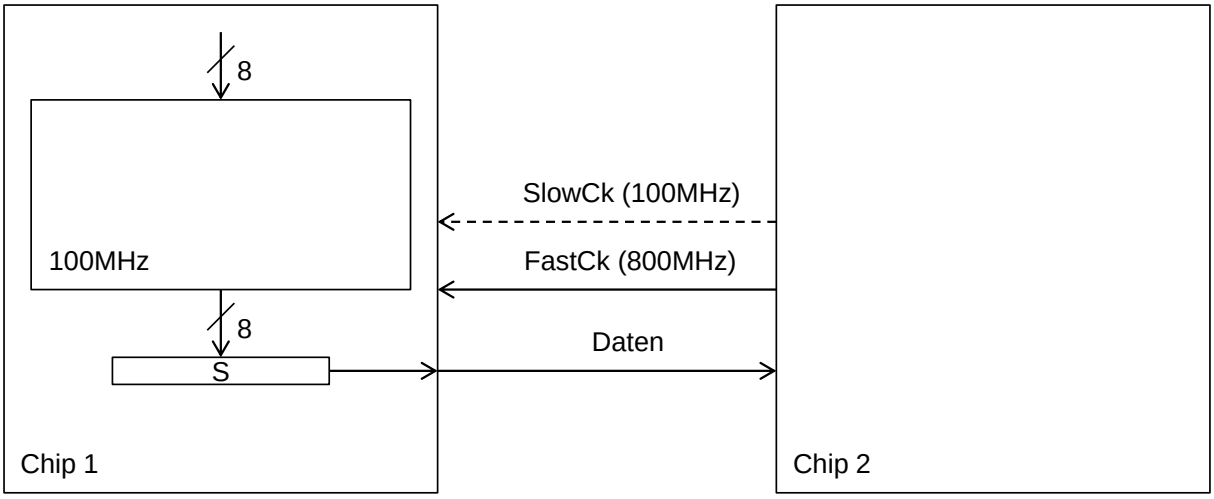

Abbildung 8: Serielle Datenübertragung mit separaten Taktleitungen – Bit- und Datentakt werden gesendet (SER1)

Möglichkeit SER2: Bit-Takt wird übertragen

Der Chip 2 sendet den Bit-Takt (Abbildung 9). Wir werden jetzt auf einige Komponenten eingehen.

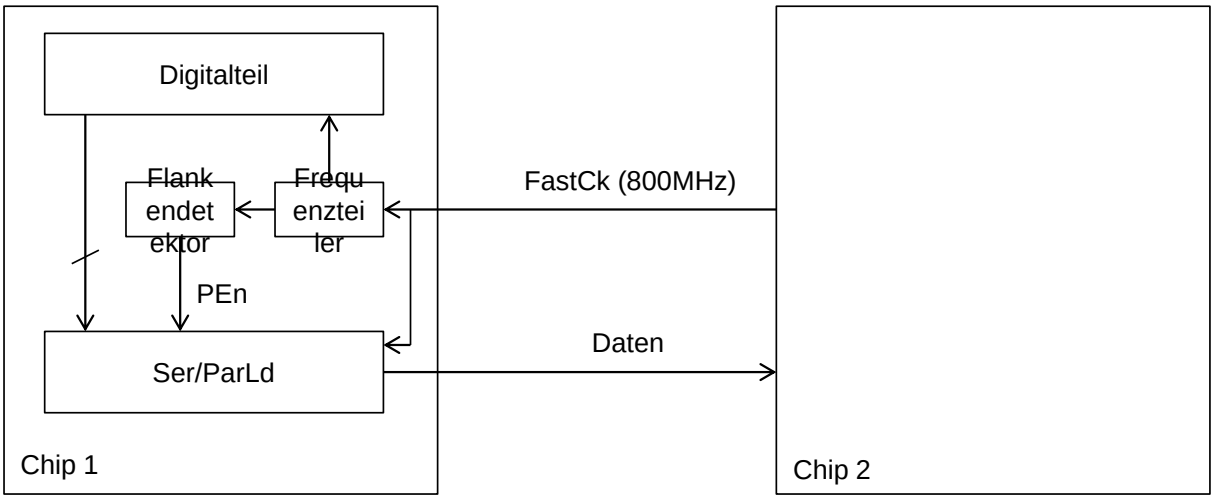

Abbildung 9: Serielle Datenübertragung mit separaten Taktleitung, Bit-Takt wird gesendet (SER2) - Komponenten

Der Daten-Takt muss auf dem Chip 1 generiert werden. Dafür brauchen wir einen Frequenzteiler – clock divider. Eine Implementierung ist in Abbildung 10. Drei-Bit ripple conter. Abbildung 11 zeigt Signalverlauf.

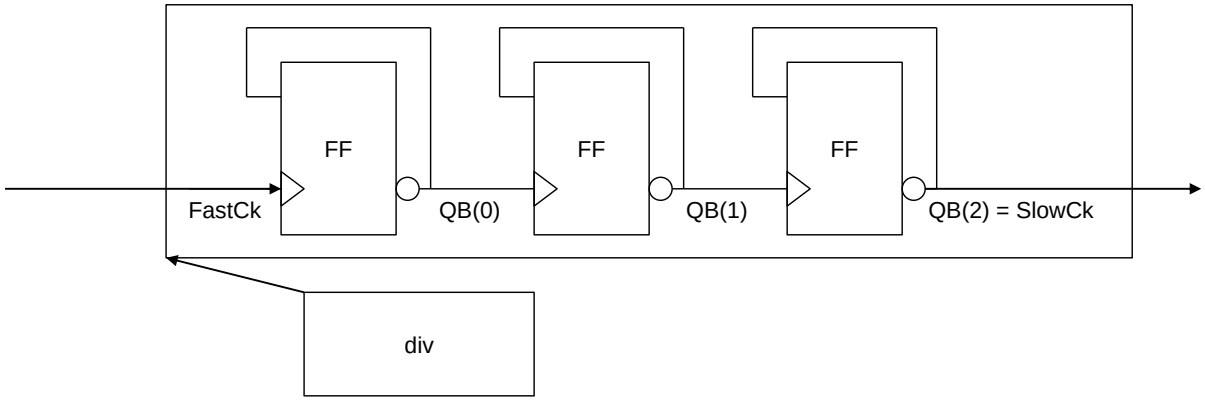

Abbildung 10: Frequenzteiler als ripple counter

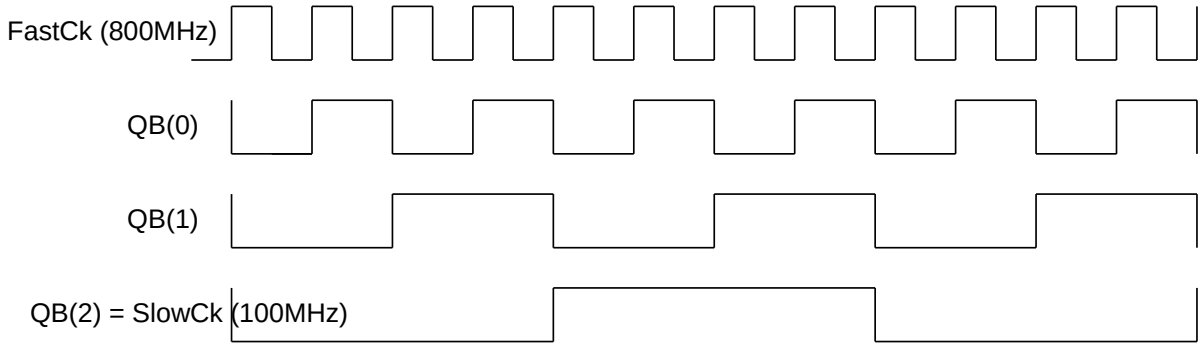

Abbildung 11: Frequenzteiler - Signale

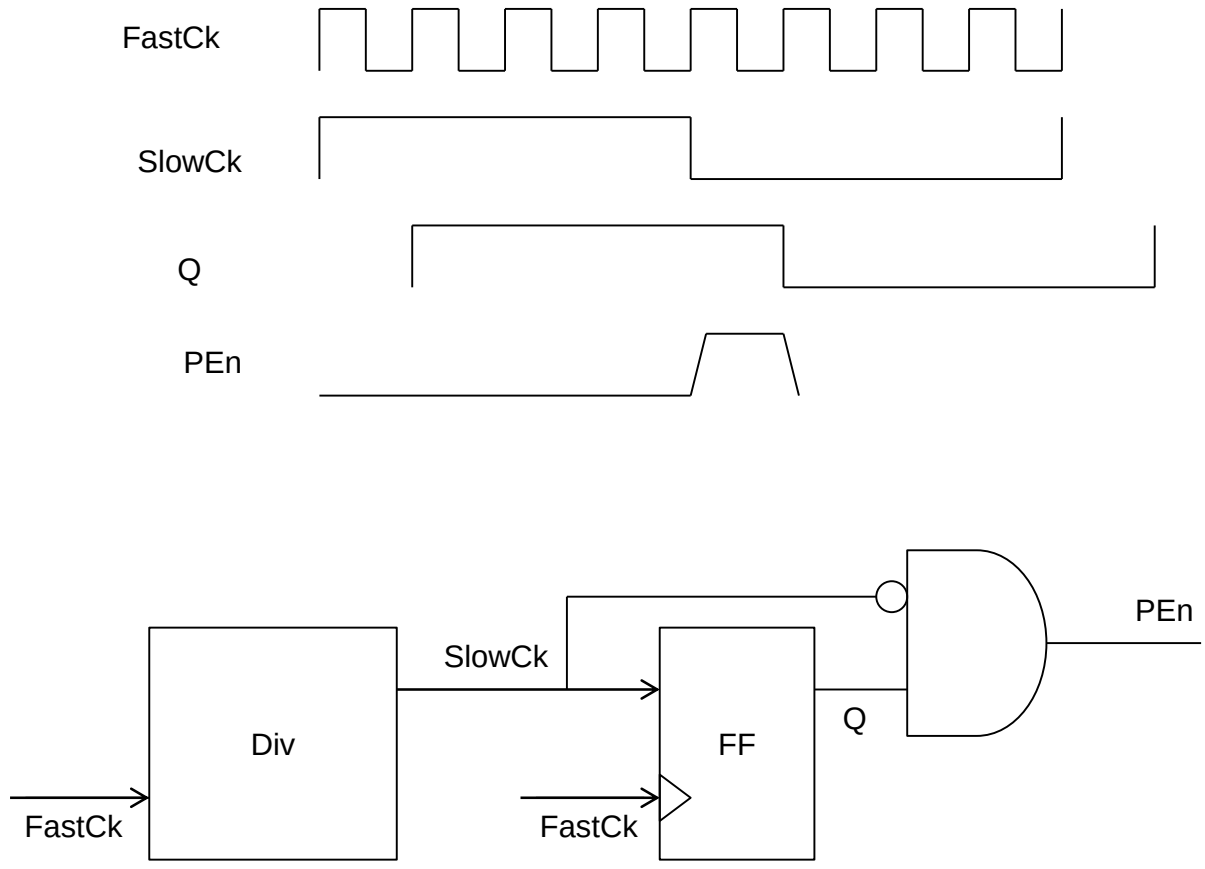

Abbildung 12: Flankendetektor

Das Signal PEn kann wie in Abbildung 12 erzeugt werden. Die Schaltung mit einem Flipflop und einem AND ist ein Flankendetektor. Der Ausgang der Schaltung ist für eine Taktperiode nach der fallenden clk-slow Flanke logisch Eins.

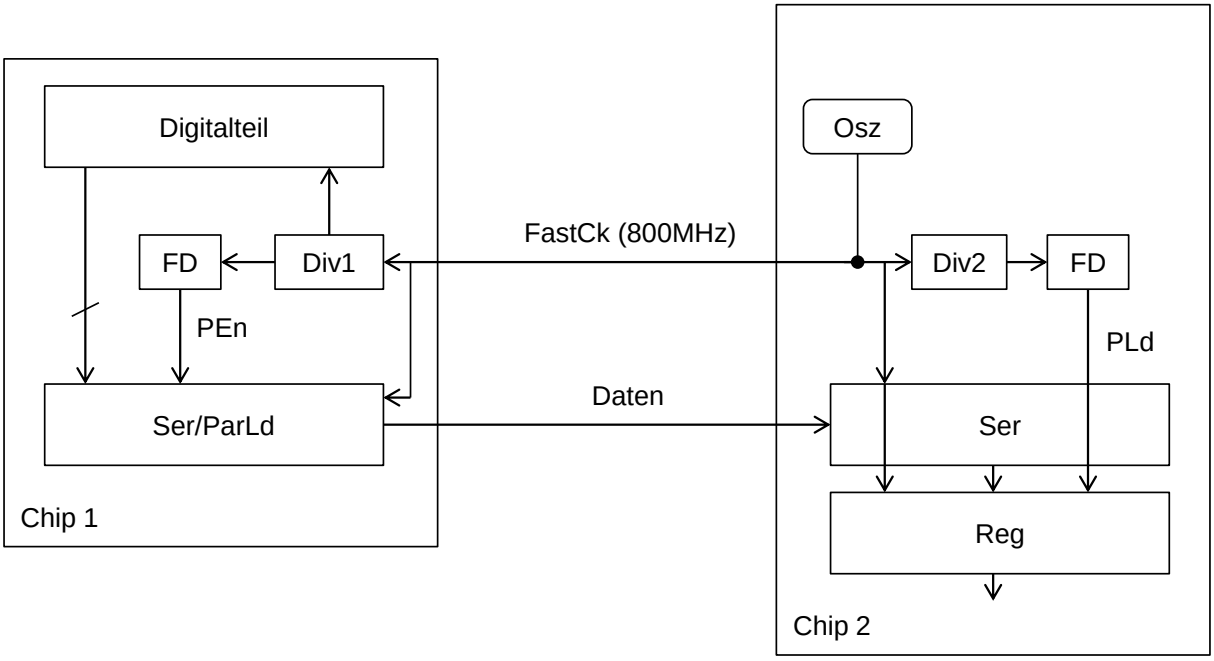

Abbildung 13: Serialisierer (Chip 1) und Deserialisierer (Chip 2). Ser/ParLd ist ein Schieberegister mit der Möglichkeit Daten parallel zu laden, FD sind Flankendetektoren, Div1/Div2 sind Taktuntersetzter (clock divider), Ser ist ein Schieberegister und Reg ein Parallelregister, Osz ist Oszillator.

Chip 2 enthält einen Empfänger – den Deserialisierer (Abbildung 13).

Abbildung 14 zeigt Zeitdiagramm der Signale im Serialisierer und Deserialisierer.

Die Bits aus dem Schieberegister werden auf PLd = 1 und auf die nächste Taktflanke in ein Parallelregister (Reg) geladen.

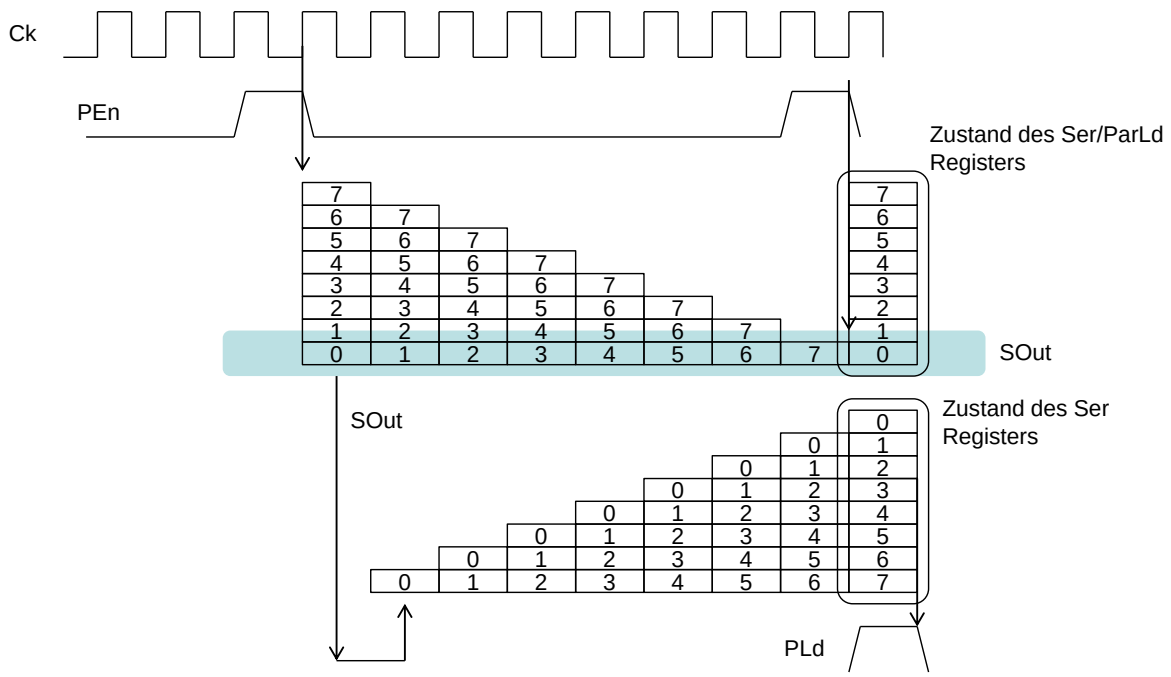

Abbildung 14: Zeitdiagramm der Signale im Serialisierer und Deserialisierer

Die beschriebene synchrone Datenübertragung hätte zwei Nachteile.

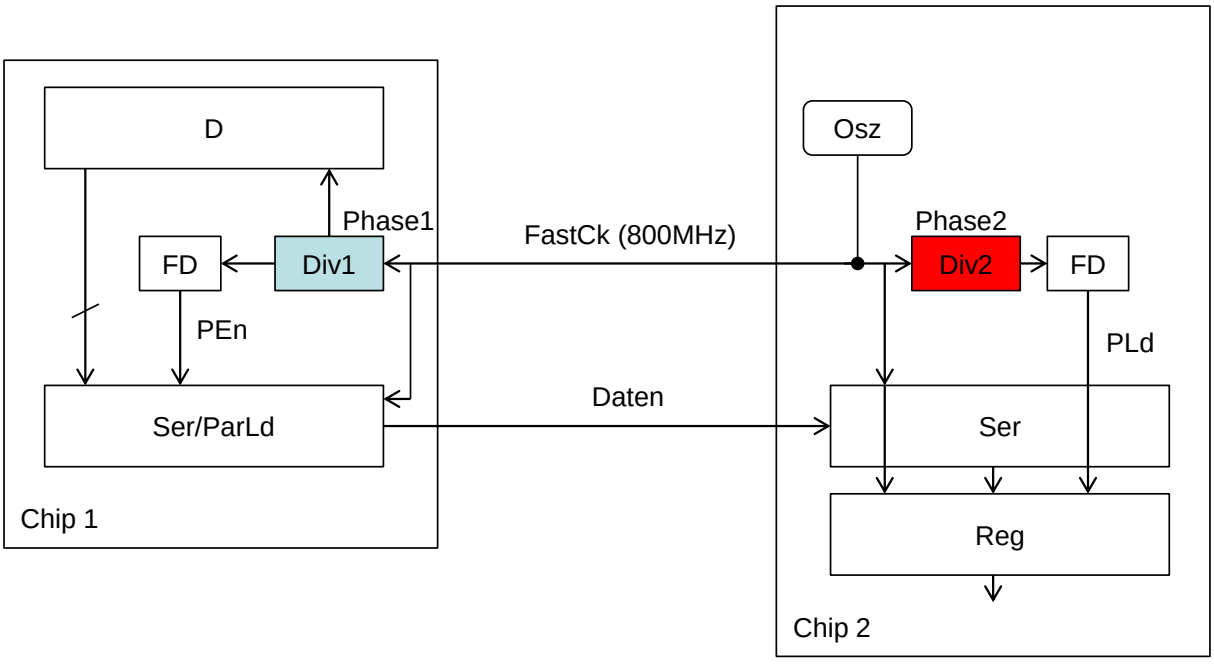

Abbildung 15: Phasen von Ld Signalen sind ungleich

Die Phase des Ripple Zählers auf dem Chip 1 ist unbekannt. Der Chip 2 „weiß nicht“ wann das höchstwertigste Bit (MSB) gerade gesendet wird. PEn ist nicht mit PLd synchron (Abbildung 15). Dieser Nachteil ist oft nicht so kritisch, da die digitalen Schaltungen normalerweise in einen Testmodus versetzt werden können, wo sie die definierten Bitsequenzen (Testmuster, Kontrollwort) senden. Der Chip 2 kann dann die Phase von PLd variieren bis die Daten korrekt empfangen werden (Abbildung 16). Eine weitere Möglichkeit

ist es einen Leitungscode zu verwenden. Diese Kodierung erlaubt versenden von Kontrollworten die in der Bitsequenz erkannt und für Phasenbestimmung verwendet werden können. Das wird später genauer erklärt.

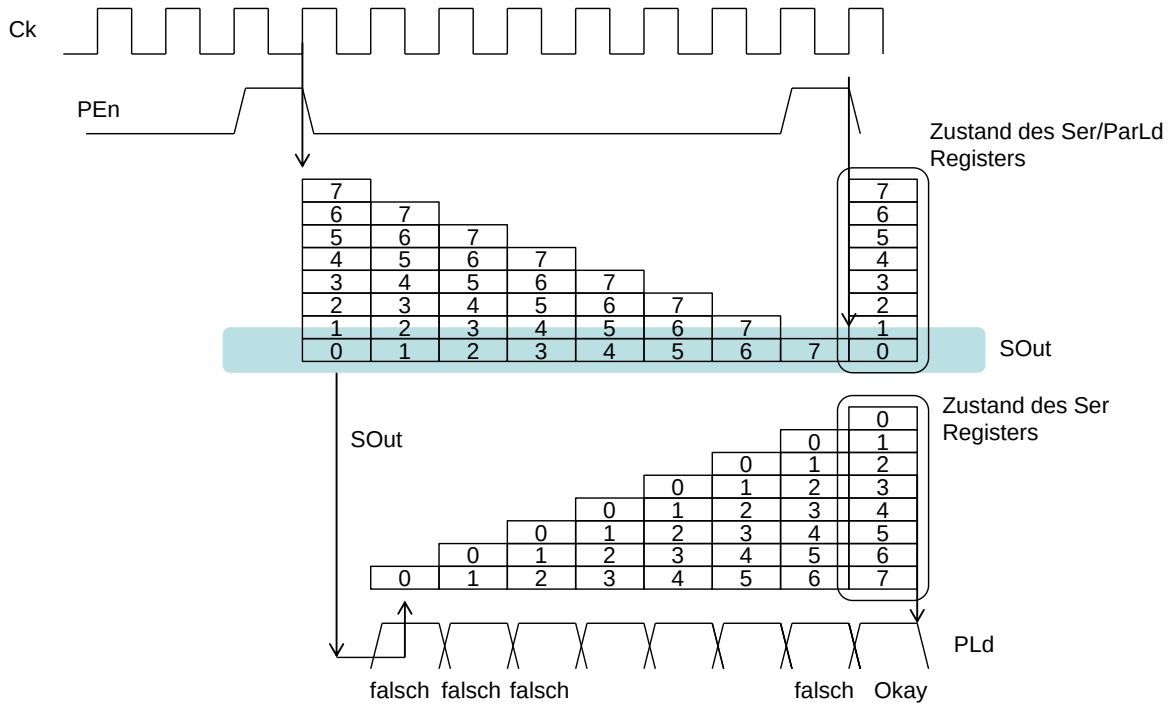

Abbildung 16: Suche nach der richtigen Phase

Der zweite Nachteil ist es, dass man das schnelle Taktsignal senden muss. Die Übertragung von schnellen Signalen ist nicht einfach. Die Signale werden in der Datenleitung verzerrt da die Leitung auch als Tiefpass-Filter wirkt (Abbildung 17).

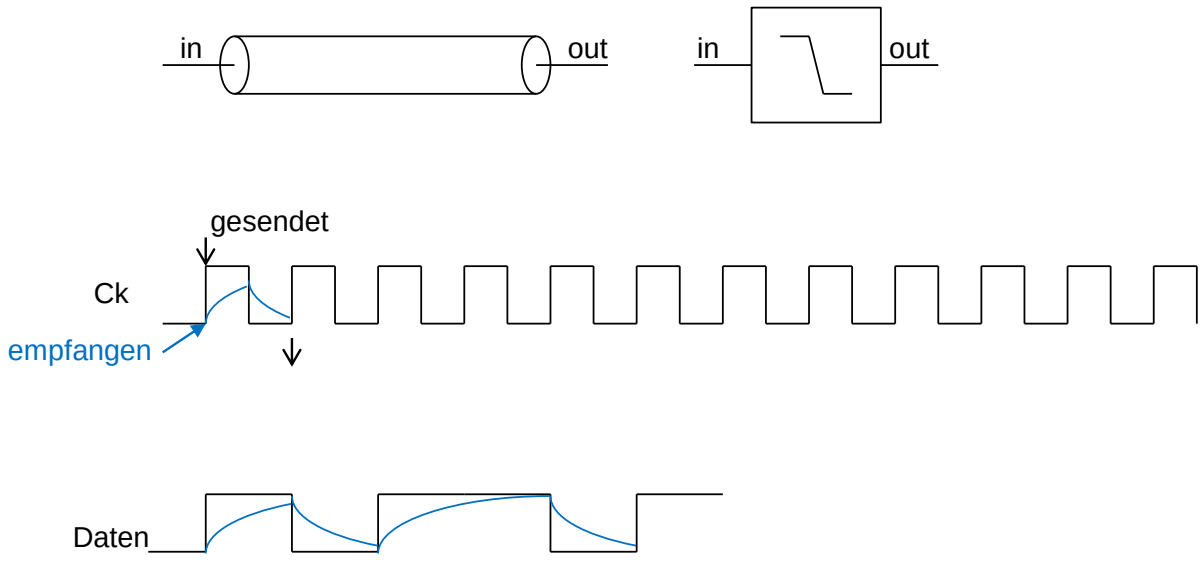

Abbildung 17: Signale werden in der Datenleitung verzerrt da die Leitung auch als Tiefpass-Filter wirkt

Beachten wir, dass sich das Taktsignal 2x schneller ändert als die Daten selbst, Abbildung 17.

Möglichkeit SER3: Langsamer Takt wird übertragen, Taktmultiplikation

Die Implementierung SER3 kann diese zwei Nachteile zu beheben, Abbildung 18.

Der Chip 2 sendet das langsame 100MHz Taktsignal (Daten-Takt). Das schnelle Taktsignal (Bit-Takt) wird mit einem Takt Multiplizierer auf dem Chip 1 erzeugt (Block Mult). Die Flanken vom SlowCk und vom multiplizierten Taktsignal sind synchron.

Diese Schaltung hat einige Vorteile im Vergleich zu der vorherigen Schaltung, wo der schnelle Takt übertragen wird.

Der erste Vorteil ist es, dass nur der langsame Takt übertragen werden muss. Ein weiterer Vorteil ist es, dass die Phase vom PEn Signal auch auf dem Chip 2 bekannt ist. Man kann das Signal PEn mit einem Flankendetektor (FD) aus der fallenden Flanke von SlowCk erzeugen. Da der Chip 1 das Signal SlowCk sendet, kennt er die Phase vom PEn und kann die Daten richtig abtasten.

Der komplizierteste Block ist der Takt-Multiplizierer. Wir werden jetzt aus dessen Struktur eingehen.

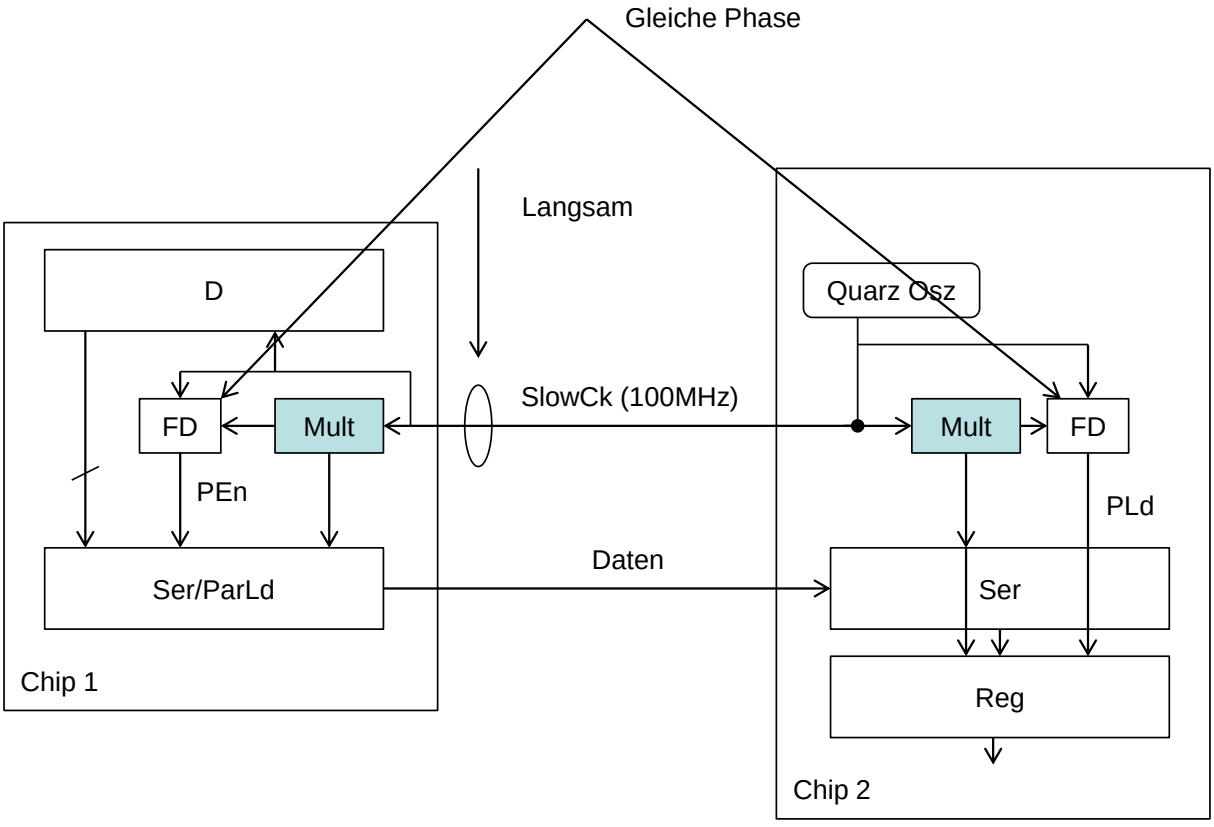

Abbildung 18: Serielle Datenübertragung mit separaten Taktleitung, Daten-Takt wird gesendet – Komponenten (SER3)

Phasenregelschleife (Phase-Locked Loop)

Takt-Multiplizierer wird normalerweise als Phase-Locked Loop (PLL) (Phasenregelschleife) realisiert.

Abbildung 19 zeigt die Struktur der PLL.

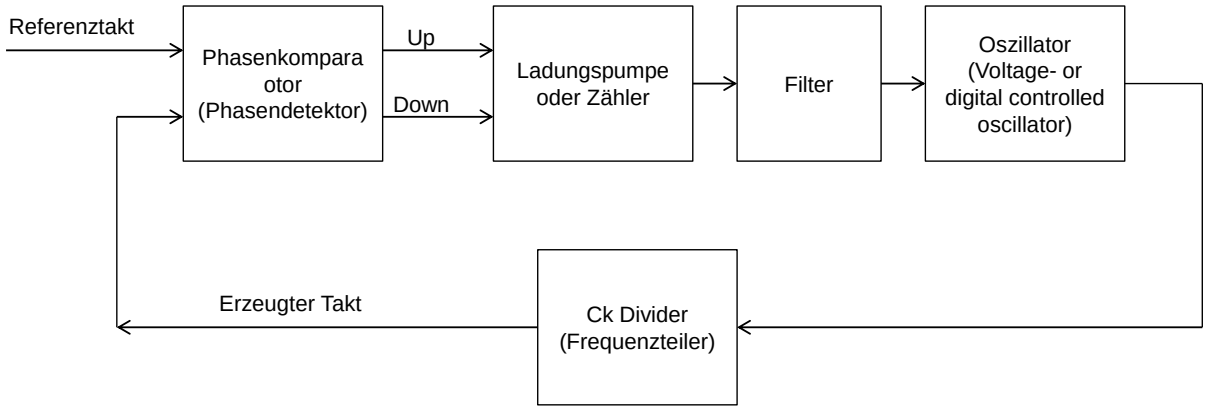

Abbildung 19: Takt-Multiplizierer wird als Phase-Locked Loop (PLL) (Phasenregelschleife) realisiert.

Das Eingangselement einer PLL ist der Phasenkomparator.

Phasenkomparator vergleicht die Phase des Referenztakts, in unserem Fall ein 100MHz Takt, mit der Phase des erzeugten Takts. Der erzeugte Takt wird mit einem Oszillator und Frequenzteiler erzeugt.

Phasenkomparator erzeugt die Up und Down Signale, je nachdem ob die Phase des erzeugten Takts kleiner oder größer als die Phase des Referenztakts ist. Anders gesagt, wenn der erzeugte Takt verspätet ist wird Up erzeugt (Abbildung 20) und wenn der Takt zu früh kommt wird Down erzeugt (Abbildung 24). Die Pulsweite ist gleich wie die Verspätung, bzw. die Phasendifferenz.

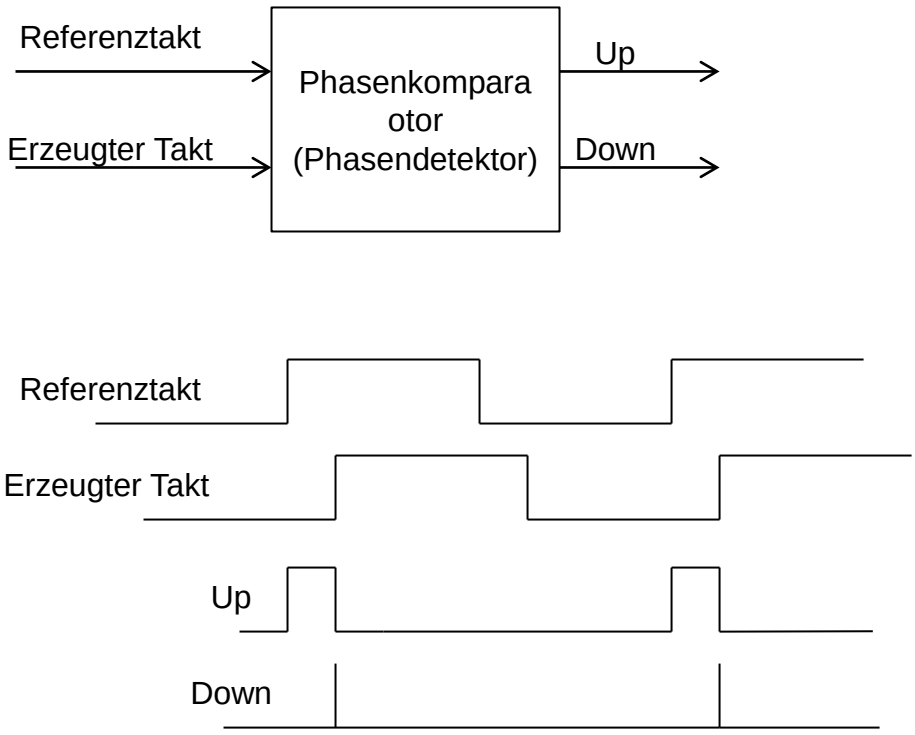

Abbildung 20: Phasenkomparator erzeugter Takt zu spät

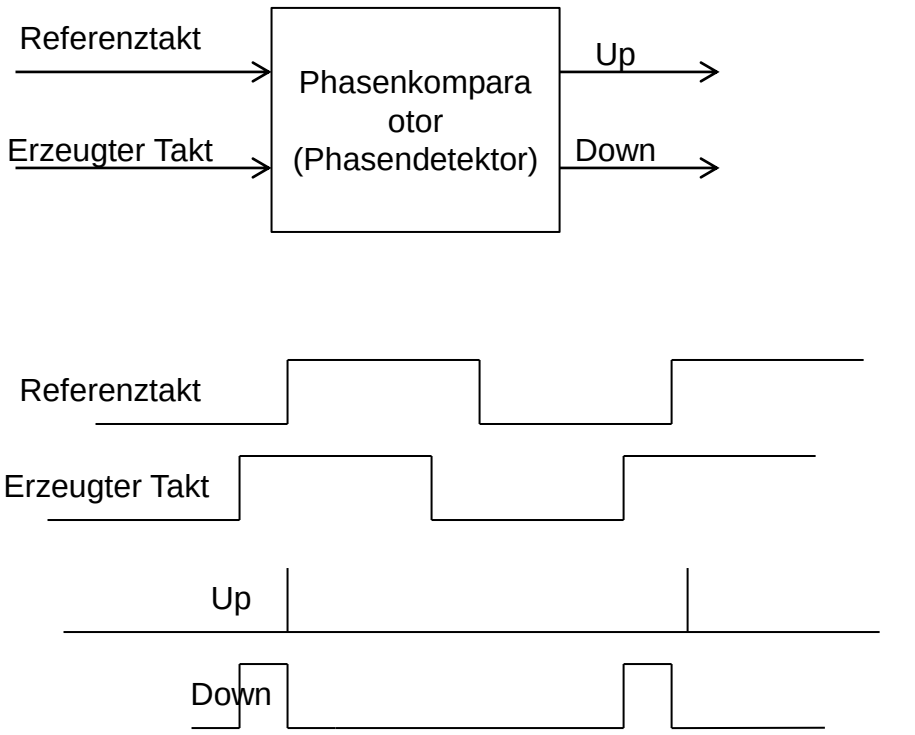

Abbildung 21: Phasenkomparator erzeugter Takt zu früh

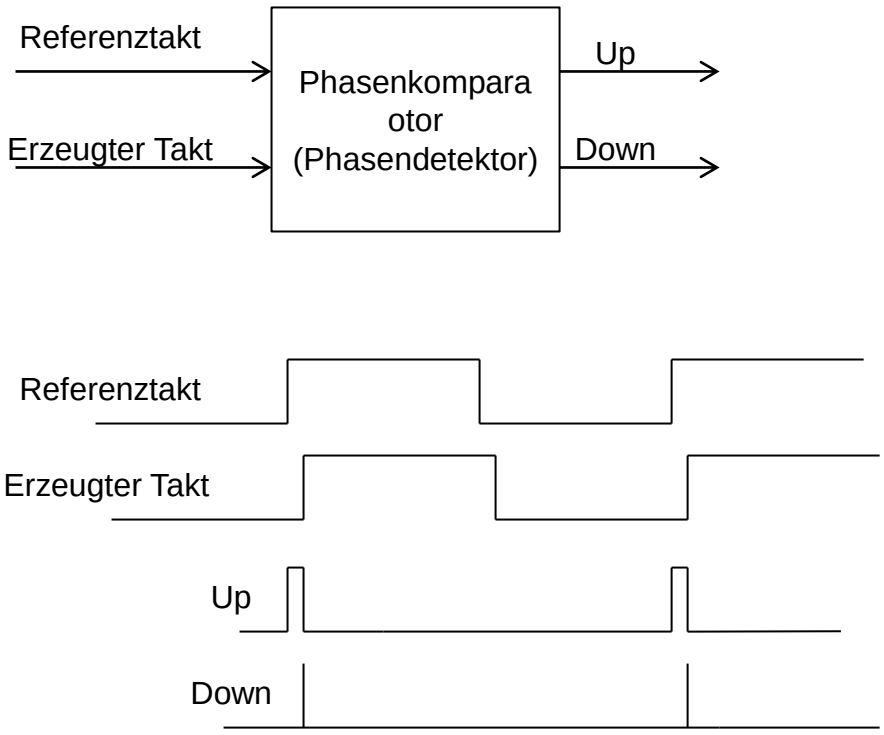

Abbildung 22: Phasenkomparator – Phasendifferenz ist kleiner

Der Hauptteil der PLL ist ein spannungsgesteuerter Oszillator – VCO. Dieser Oszillator oszilliert mit einer variablen Frequenz. Die Frequenz ist zur Eingangsspannung proportional. Der Oszillator läuft schneller, wenn die Eingangsspannung höher ist.

Eine Alternative zum spannungsgesteuerten Oszillator ist ein digitaler Oszillator. Seine Frequenz ist zur digitalen Eingangsvariable proportional.

Aus der Oszillator-Frequenz wird der erzeugte Takt hergeleitet. (In unserem Fall 800MHz) Dieser Takt wird mit einem Frequenzteiler generiert. Der geteilte Takt wird an den Phasenkomparator angeschlossen.

Die PLL ist eine Phasenregelschleife – ein System mit Gegenkopplung. Das System funktioniert wie folgend (Abbildung 23):

Wenn der geteilte Takt gegenüber der Taktreferenz verspätet ist (Abbildung 23), werden vom Phasenkomparator die Up-Signale erzeugt.

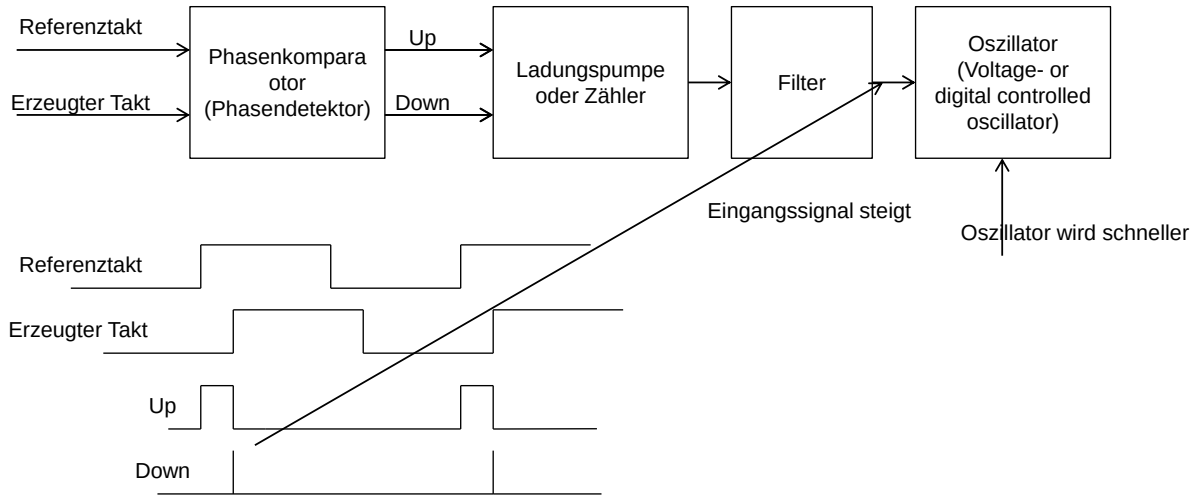

Abbildung 23: PLL Funktionsweise, erzeugter Takt zu spät

Die Up-Signale führen dazu, dass die VCO-Eingangsspannung steigt und der Oszillator schneller wird.

Es ist die Aufgabe der Ladungspumpe, aus den Up- und Down-Signalen die VCO-Eingangsspannung zu erzeugen.

Wenn ein Digitaloszillator verwendet wird, kann ein Zähler die Ladungspumpe ersetzen.

Wenn der erzeugte Takt eilt (Abbildung 24), werden vom Phasenkomparator die Down-Signale erzeugt. Die VCO-Eingangsspannung sinkt und der Oszillator wird langsamer.

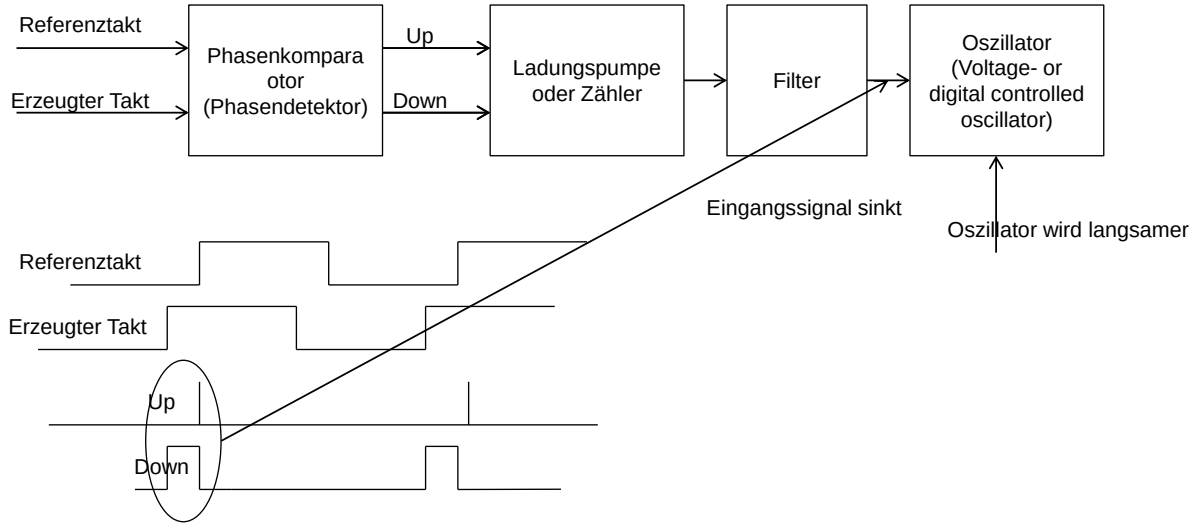

Abbildung 24: PLL Funktionsweise, erzeugter Takt zu früh

Phasenkomparator

Abbildung 25 zeigt eine Implementierung des Phasenkomparator.

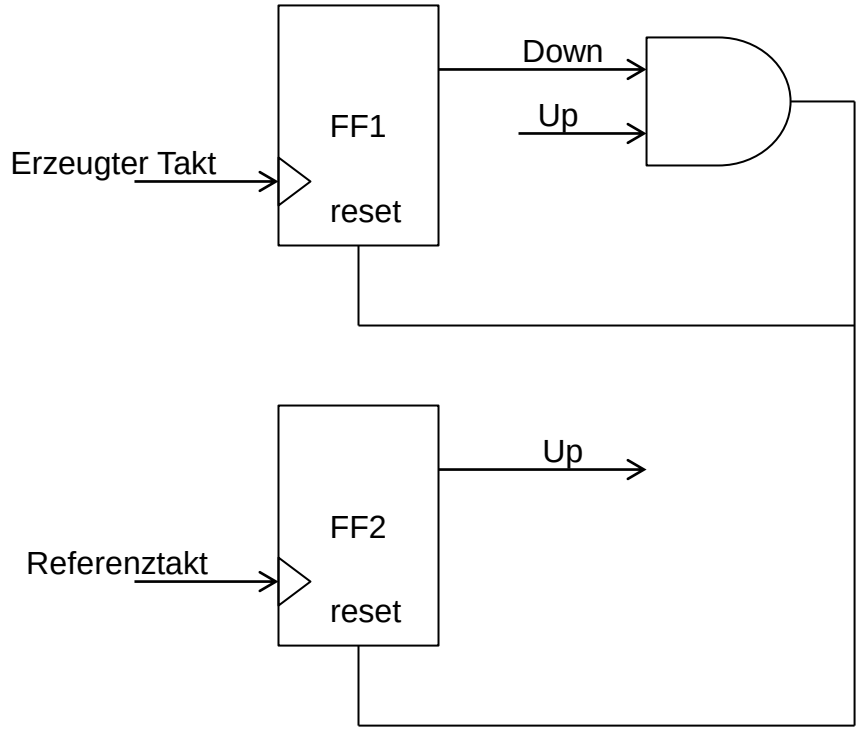

Abbildung 25: Phasenkomparator

Abbildung 26 zeigt Verlauf der Signale.

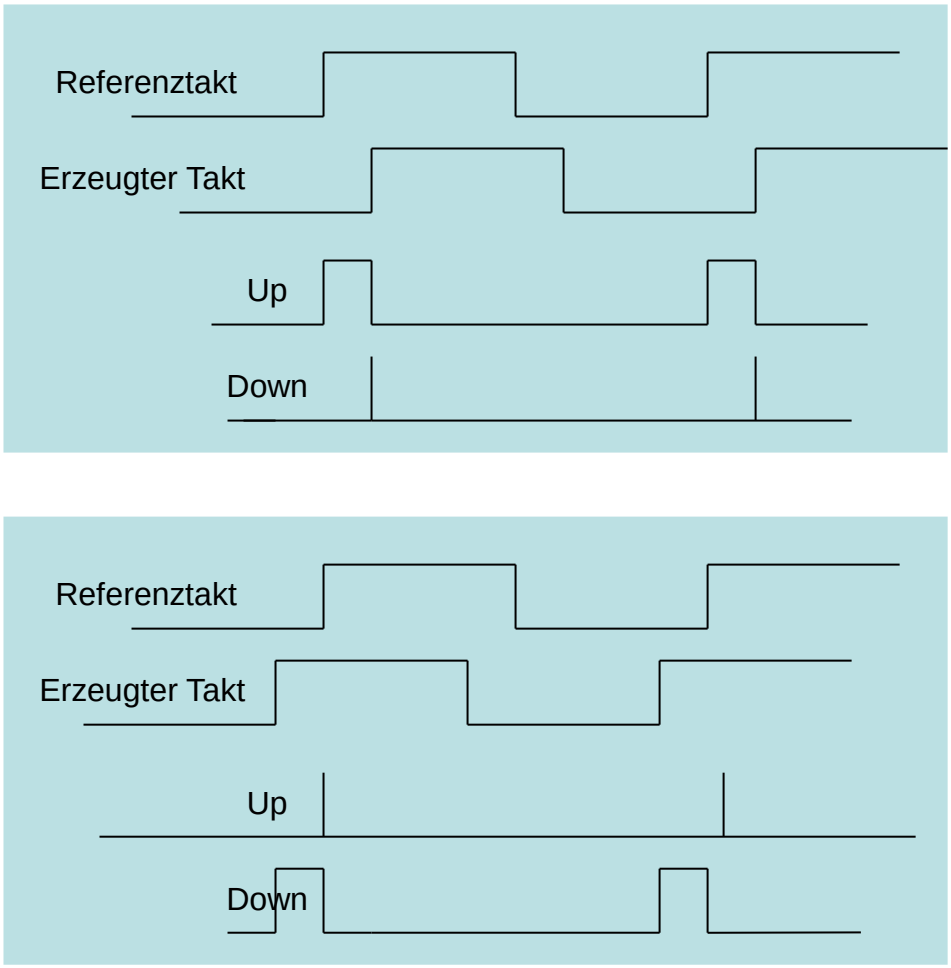

Abbildung 26: Phasenkomparator - Signale

Abbildung 26 unten zeigt den Fall wo der Referenztakt verspätet ist. Flipflop 2 kommt in den Zustand 1 auf die steigende Flanke vom erzeugten Takt. Down wird erzeugt. Flipflop 2 kommt in den Zustand 1 auf die steigende Flanke vom Referenztakt. Up wird erzeugt.

Wenn beide Flipflop-Ausgänge Eins werden, erfolgt ein Reset von beiden Flipflops. Deswegen ist das Signal Up deutlich kürzer.

Auf ähnliche Weise funktioniert die Schaltung wenn der Referenztakt früher kommt (Abbildung 26 oben).

Spannungsgesteuerter Oszillator (Voltage Controlled Oscillator - VCO)

Die zwei häufigsten Implementierungen vom spannungsgesteuerten Oszillator in CMOS Technologie sind der Ringoszillator und der LC-Oszillator.

Spannungsgesteuerter Oszillator – Wikipedia

Ringoszillator

Der Ringoszillator ist eine Kette von Buffern und Invertern (Abbildung 27). Es ist dabei wichtig, dass die Kette das Eingangssignal insgesamt negiert (es gibt n Buffers und einen Inverter).

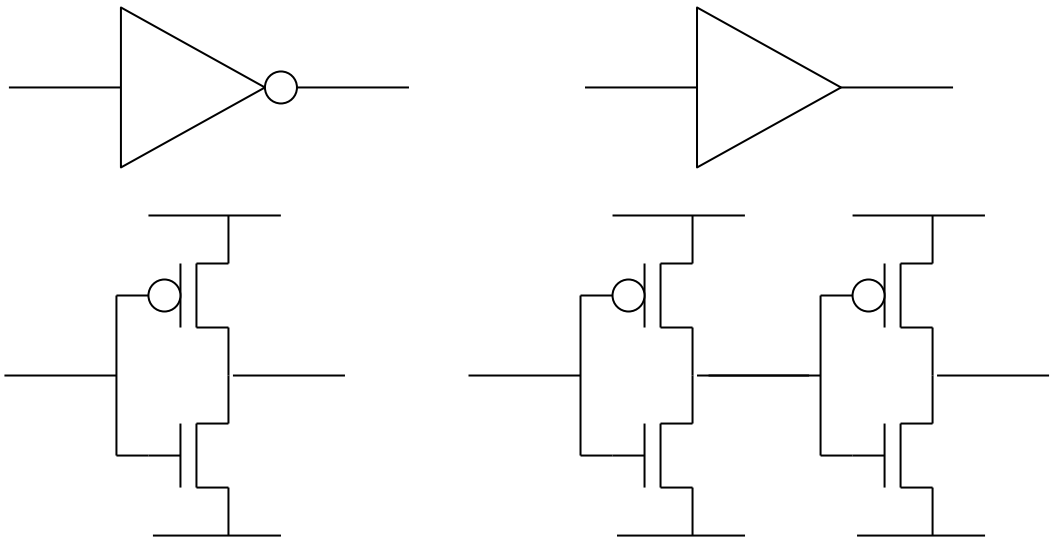

Abbildung 27: Inverter und Buffer

Betrachten wir einen Ring-Oszillator mit Start-Signal, Abbildung 28.

- …

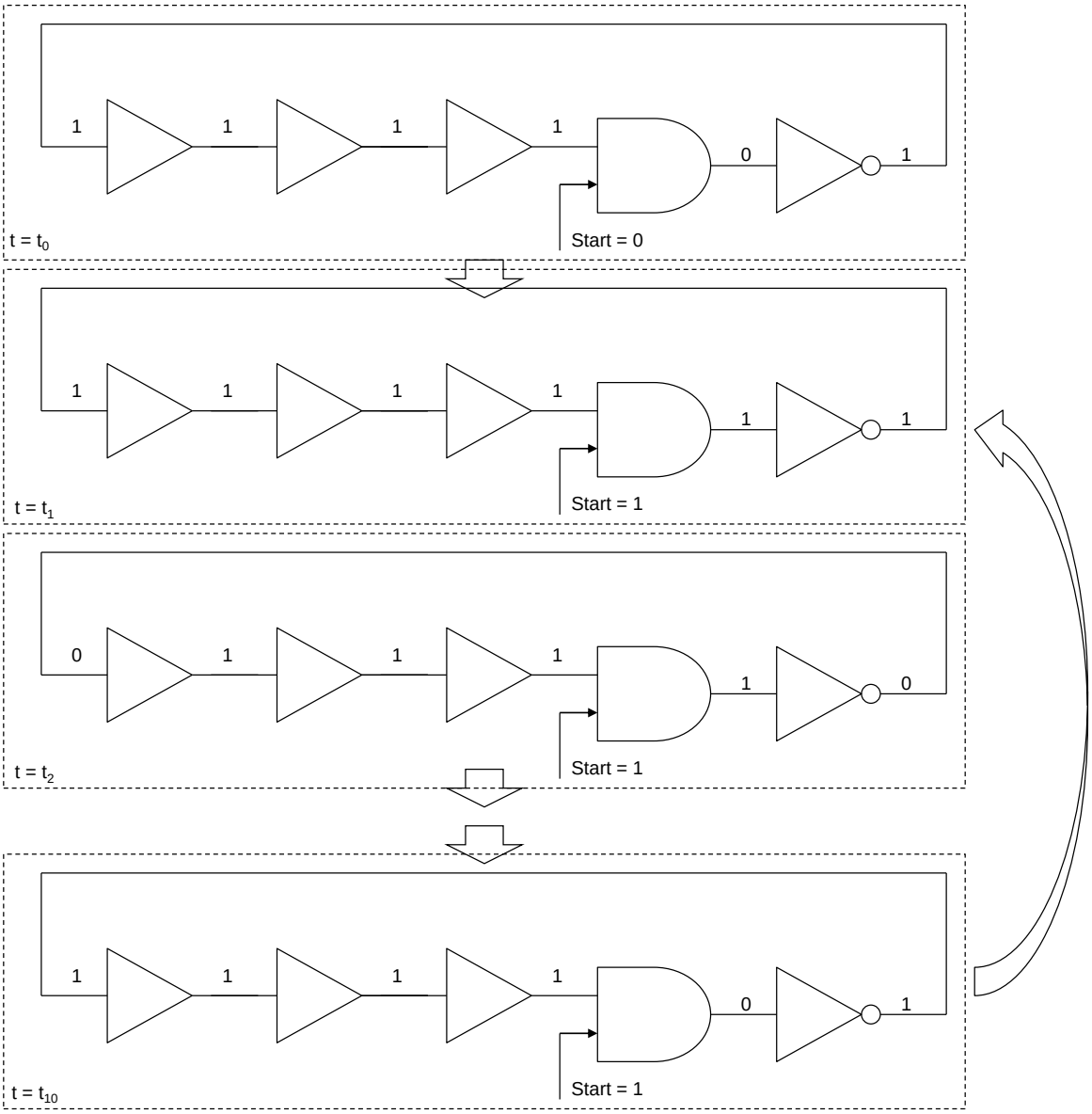

Abbildung 28: Ringoszillator mit Start-Signal

Wenn das Startsignal 0 ist ($t = t_0$), befinden sich die Komponenten des Oszillators im stabilen Zustand. Oszillator oszilliert nicht.

Fig Abbildung 28 zeigt wie sich die Zustände der Schaltung verändern ($t = t_1$ bis $t = t_{10}$) wenn das Startsignal auf 1 gesetzt wird. Oszillator oszilliert und hat 10 Zuständen. Die Zahl von Zuständen entspricht der Zahl von Komponenten (3 Buffers, ein AND und ein Inverter) multipliziert mit 2.

Oft wird der Ringoszillator ohne Startsignal implementiert (Abbildung 29).

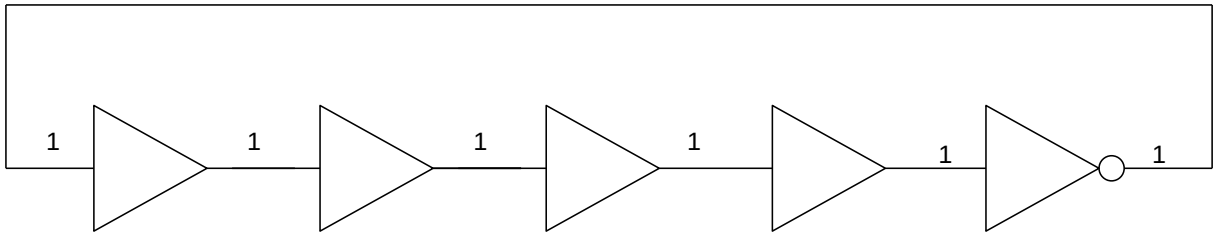

Abbildung 29: Einfacher Ringoszillator

Manchmal wird der Ringoszillator als Kette von Invertern gezeichnet (Abbildung 30). Beide Darstellungen in Abbildung 30 zeigen dieselbe Schaltung. Ein Buffer ist die Serienschaltung von zwei Invertern.

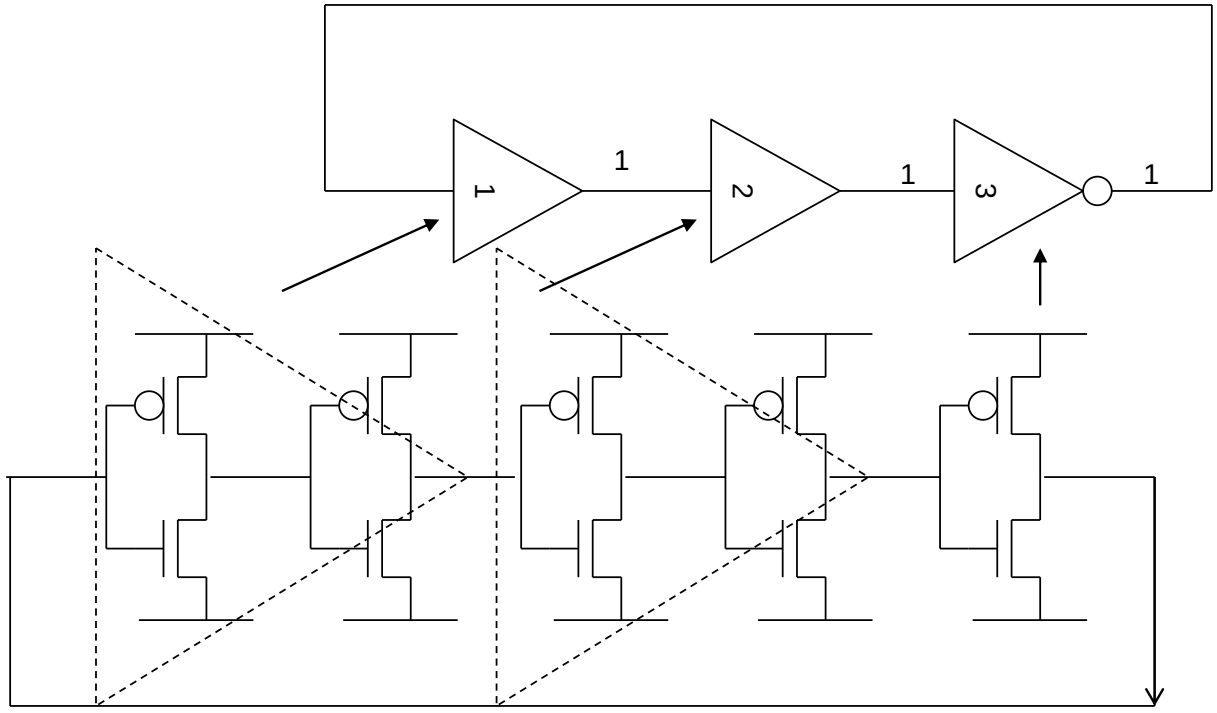

Abbildung 30: Einfacher Ringoszillator - Transistorschaltplan

Die Periode des Oszillators ist die Summe von Eingang-Ausgang Signalverzögerungen (delay) von allen Komponenten für die Änderungen von 0 auf 1 und von 1 auf 0.

$T = T_{del1,buffer}(0->1) + T_{del2,buffer}(0->1) + T_{del3,inv}(0->1) + T_{del1,buffer}(1->0) + T_{del2,buffer}(1->0) + T_{del3,inv}(1->0)$

Im Fall vom Buffer gilt:

$T_{del,buffer}(0->1) = T_{del,buffer}(1->0) = T_{del,buffer}$

Die Verzögerung vom Buffer ist die Summe von Signalverzögerungen in zwei Invertern (Abbildung 31).

$T_{del,buffer} = T_{del,inv}(0->1) + T_{del,inv}(1->0)$

Die Verzögerungen vom Inverter wurden in Vorlesung 3 hergeleitet:

$$T_{del,inv}(0 \rightarrow 1) = \frac{4C}{\mu_p C'_{ox} \frac{W_p}{L_p} V_{gst}}$$

$$T_{del,inv}(1 \rightarrow 0) = \frac{4C}{\mu_n C'_{ox} \frac{W_n}{L_n} V_{gst}}$$

C ist die Lastkapazität ist, μ_n und μ_p sind die Beweglichkeiten von Elektronen und Löchern, W_n, L_n und W_p, L_p sind die Transistorgate-Dimensionen vom NMOS und PMOS Transistor.

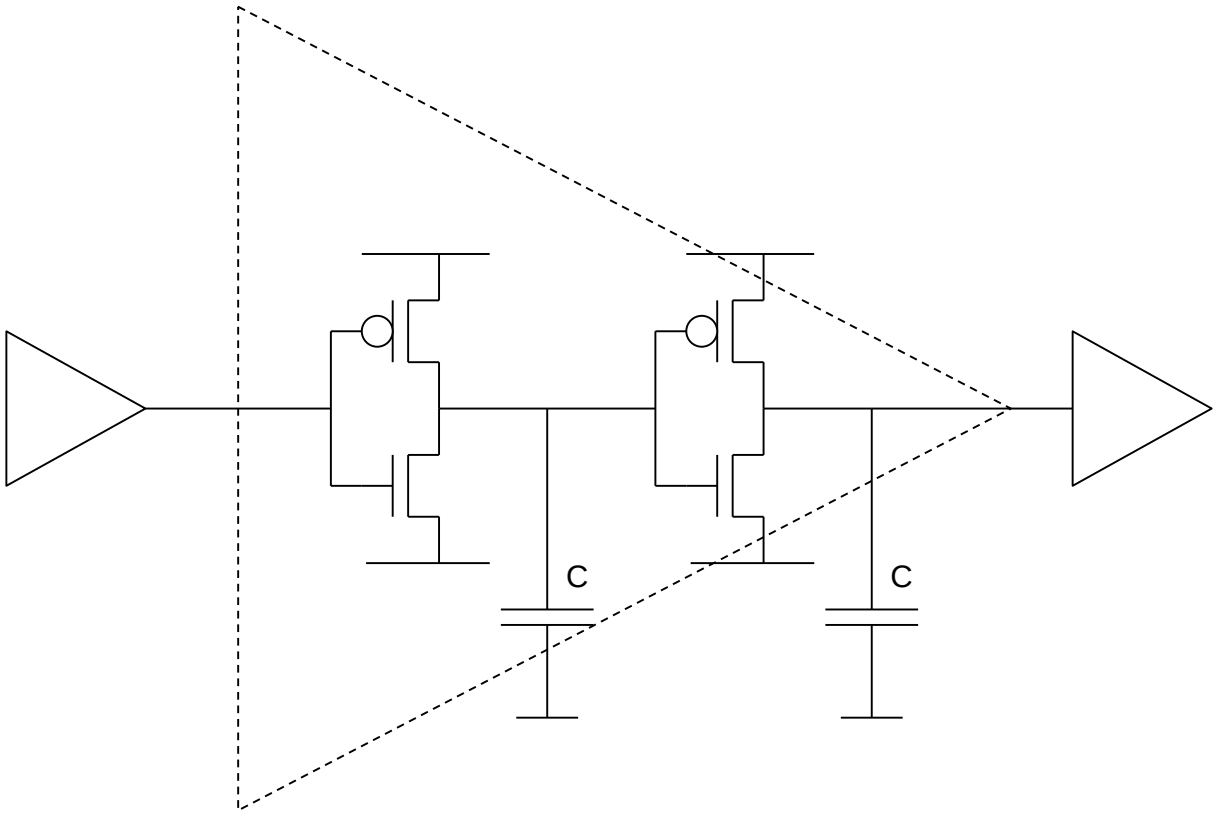

Abbildung 31: Signalverzögerung im Buffer

Wie können wir die Frequenz verändern?

Eine Möglichkeit ist es, variable Kondensatoren zu verwenden. Abbildung 32 zeigt eine Realisierung des variablen Kondensators.

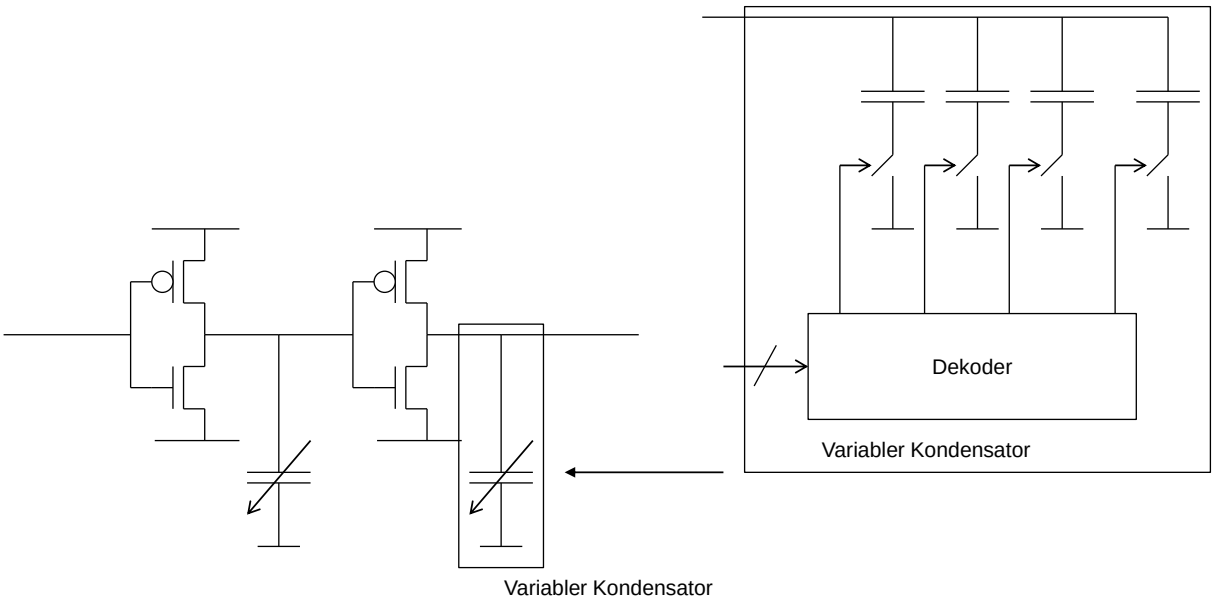

Abbildung 32: Programmierbare Kapazität

Zweite Möglichkeit ist es, regelbare Widerstände zu verwenden, Abbildung 33.

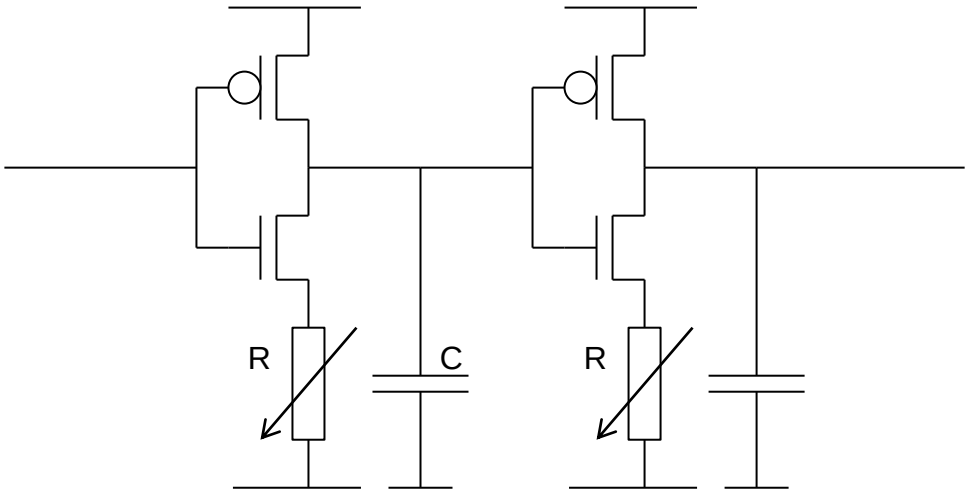

Abbildung 33: Variabler Widerstand

Wenn der Widerstand R viel größer als der Widerstand des eingeschalteten Transistors ist, kann man den letzteren in der Formel für Verzögerung vernachlässigen. Die Gesamtverzögerung des Buffers ist in dem Fall etwa:

$T_{del,buffer} \sim T_{del,inv}(1->0) \sim 3RC$

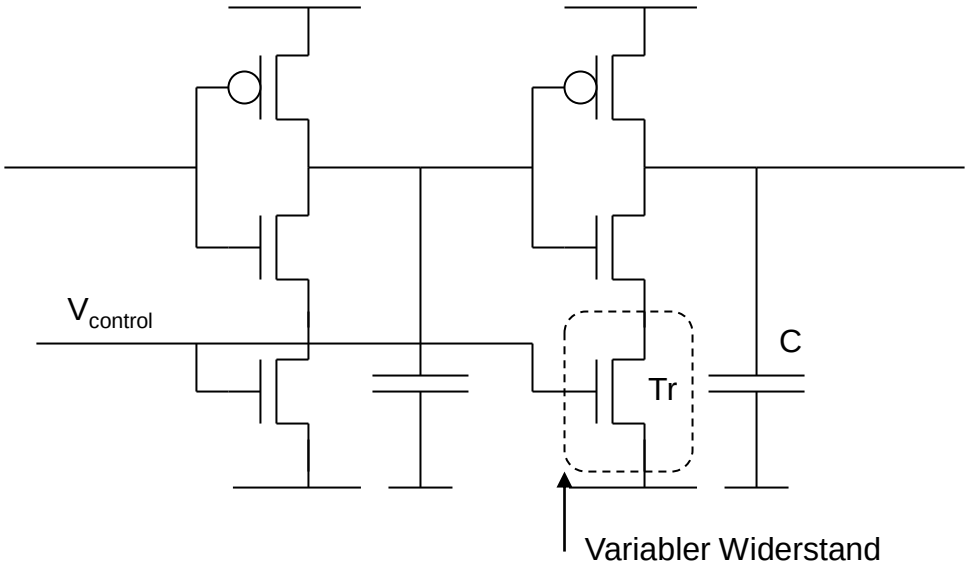

Abbildung 34: Variabler Widerstand - Implementierung

Widerstand R wird normalerweise als Transistor implementiert (Tr in Abbildung 34).

Der On-Widerstand von Tr wird mithilfe seiner Gate Spannung variiert. Wenn der Widerstand niedriger ist, dauert die Umladung der Kapazitäten kürzer und Verzögerung einer Stufe ist kleiner. Frequenz wird höher.

Es gilt

$$R_{on} = \frac{1}{\mu_p C'_{ox} \frac{W_p}{L_p}(V_{control} - V_{th})}$$

Und

$T_{del,buffer} \sim T_{del,inv}(1->0) \sim 3R_{on}C$

Die Frequenz eines Ringoszillators kann im großen Bereich verändert werden. Der Nachteil der Ringoszillators ist Taktzittern (jitter). Auch wenn die Eingangsspannung konstant ist, schwankt die Frequenz: Der Grund ist das Rauschen in den Transistoren und die Spannungsschwankungen an Versorgungslinien.

LC-Oszillator

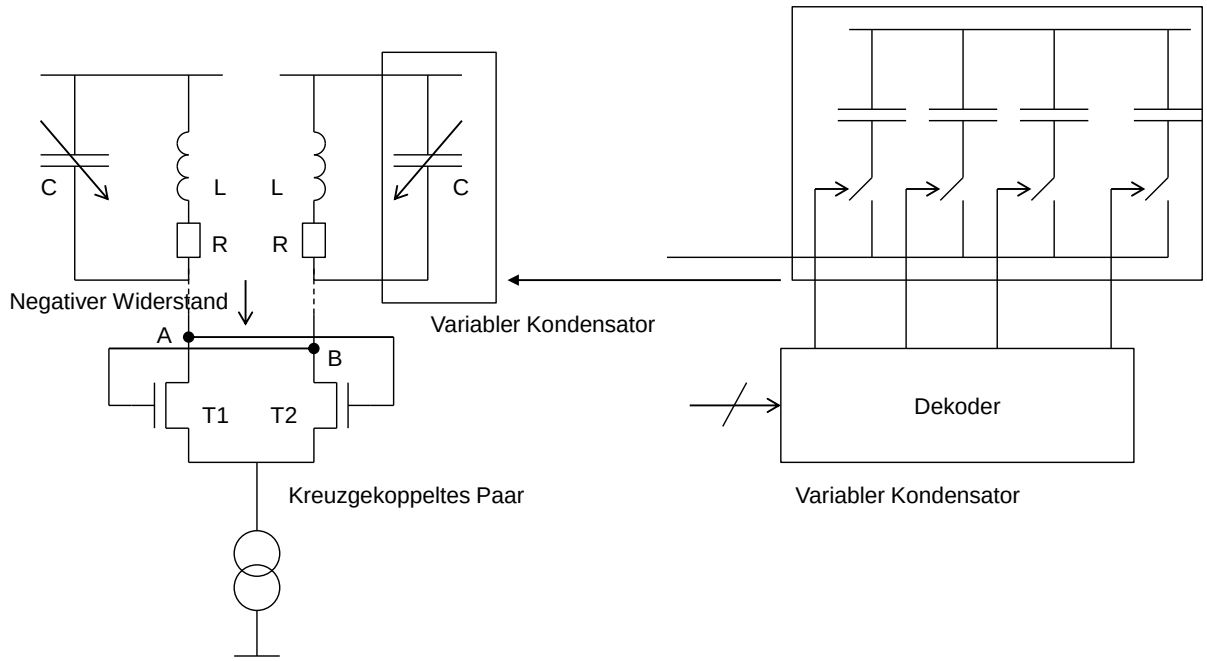

Abbildung 35: LC - Oszillator

Ein LC Oszillator hat weniger jitter. Die Spulen (Induktoren) und die Kapazitäten können auf dem Chip implementiert werden. Die Frequenz kann z.B., durch schalten von zusätzlichen Kapazitäten verändert werden.

Die Oszillationen einer realen LC-Schaltung klingen wegen dem Innenwiderstand der Spule nach einer Weile ab. Um Oszillationen aufrecht zu erhalten, brauchen wir eine aktive Schaltung, die den Innenwiderstand kompensiert. Eine Solche Schaltung ist das kreuzgekoppelte Transistorpaar T1 und T2 (Abbildung 35). Die Schaltung erzeugt einen negativen Widerstand zwischen den Punkten A und B. Der Widerstand des Paars hebt den Innenwiderstand der Spule auf. Das illustriert Abbildung 36.

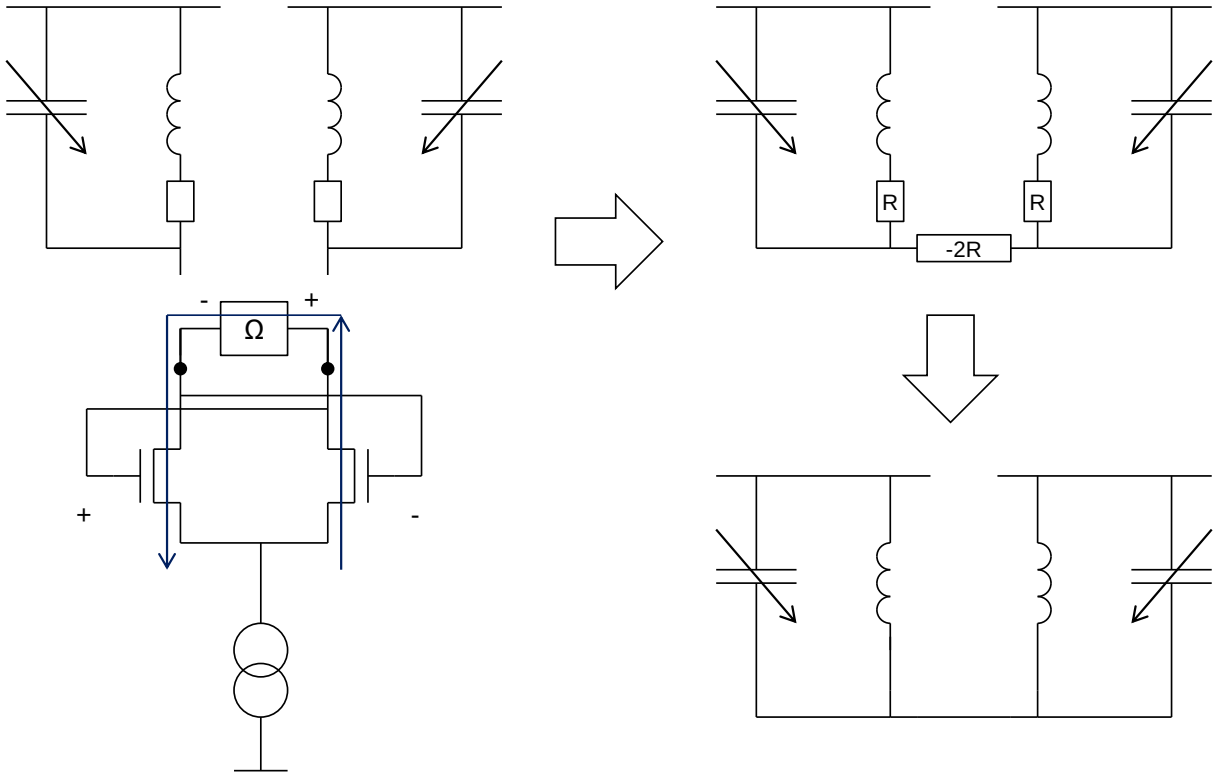

Abbildung 36: LC- Oszillator – negativer Widerstand

Ladungspumpe und Filter

Am Ende beschreiben wir die Schaltung, welche aus Up- und Down- Signalen die Eingangsspannung des Oszillators erzeugt.

Eine einfache Schalung ist in Abbildung 37 gezeigt. Die Up- und Down-Signale schalten NMOS- und PMOS-Stromquellen. Die Stromquellen laden oder entladen den Filter-Kondensator C.

Es gilt folgendes:

Durchschnittlicher Strom (innerhalb mehrerer Taktperioden) der Ladungspumpe I_{out} ist zum Mittelwert von Up- und Down-Signalen proportional. Dieser Mittelwert hängt von Phasendifferenz zwischen Taktsignalen.

Deshalb gilt:

$$I_{out}(t) = a\phi$$

A ist Konstante und φ Phasendifferenz.

Spannung am Kondensator ist durch die folgende Formel gegeben:

$$v_{out}(t) = \int \frac{i(t)}{C} dt$$

Das Mittelwert von $v_{out}(t)$ für mehrere Taktperioden ist:

$$V_{out}(t) = \int \frac{I_{out}(t)}{C} dt = \int \frac{a\phi(t)}{C} dt$$

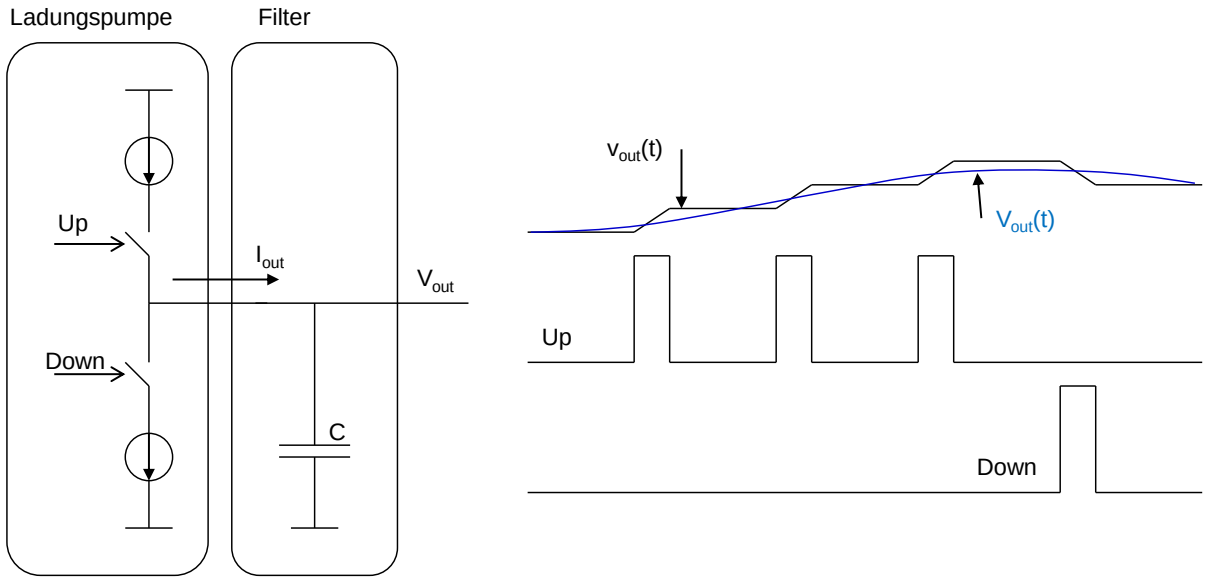

Abbildung 37: Ladungspumpe und Filter

Stabilität (Optional)

PLL ist ein System mit Rückkopplung, es ist deshalb wichtig Stabilität des Systems zu prüfen. Dafür müssen wir zuerst die Schleifenverstärkung berechnen. (Wir haben die Methode in der Vorlesung DAS gezeigt.)

Wir trennen die Rückkopplung zwischen dem Oszillator und dem Frequenzteiler, Abbildung 38.

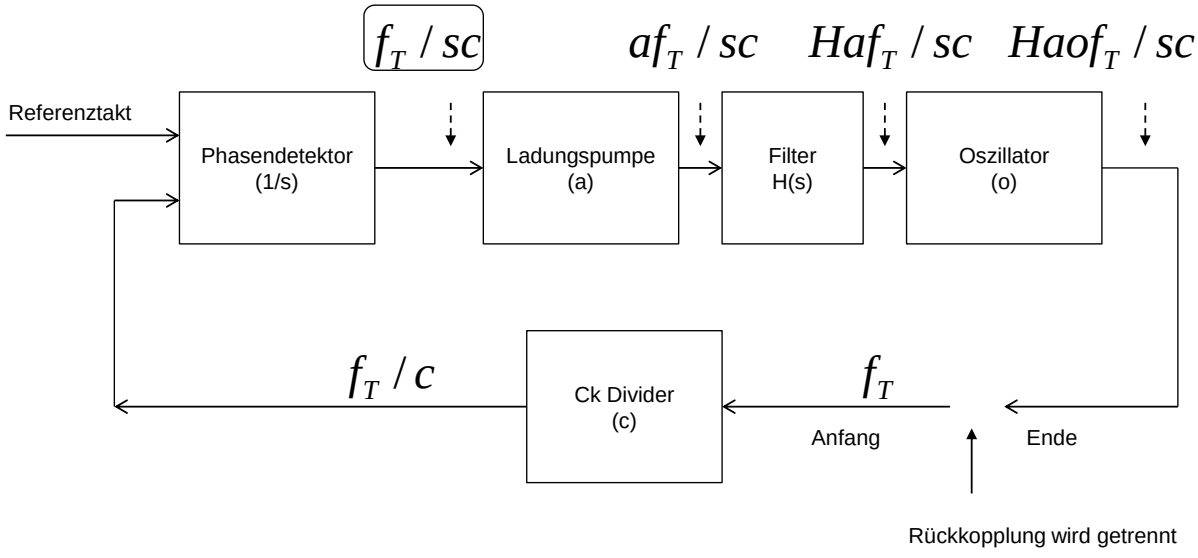

Abbildung 38: Berechnung von Schleifenverstärkung

Am Anfang nach dem Trennpunkt haben wir ein Taktsignal mit der Frequenz f_T. Wir berechnen nun wie sich die Taktfrequenz vom Anfang bis zum Ende der Schleife verändert.

Frequenzteiler verkleinert die Frequenz um Konstante c. Deswegen ist die Frequenz am Ausgang des Frequenzteilers f_T/c.

Phasenkomparator erzeugt Signale Up und Down deren Mittelwert zur Phase proportional ist.

Phase verändert sich als Zeitintergral von Taktfrequenz. Deshalb gilt für die Laplace-Transform von Phase am Eingang des Phasenkomparators.

$$\phi(s) = \frac{f_T}{sc}$$

Ladungspumpe erzeugt einen Strom der zur Phase proportional ist.

$$I(s) = \frac{\alpha f_T}{sc}$$

Filter wird durch Funktion H(s) beschrieben und erzeugt Spannung. Oszillator erzeugt ein Taktsignal der von Filterspannung anhängig ist. Die Taktfrequenz am Ende der Schleife ist

$$f_{out} = \frac{H(s)\alpha o f_T}{sc}$$

Wobei α, o und c Konstanten sind.

Die Schleifenverstärung ist

$$\beta A = \frac{f_{out}}{f_T} = \frac{H(s)\alpha o}{sc}$$

Es gilt folgende Formel, wenn wir Filter mit einem Kondesnator realisieren

$$H(s) = \frac{1}{sC}$$

In dem Fall ist die Schleifenverstärkung:

$$\beta A = \frac{ao}{cs^2 C}$$

Systeme mit solcher Schleifenverstärkung sind instabil. Das kann, z.B., mithilfe vom Nyquist-Kriterium bewiesen werden.

Wenn wir als Filter eine Reihenschaltung vom Widerstand und Kondensator verwenden, bekommen wir günstigere Schleifenverstärkung und bessere Stabilität.

Es gilt

$$H(s) = R + \frac{1}{sC} = \frac{1+sRC}{sC}$$

und

$$\beta A = \frac{ao(1+sRC)}{cs^2C}$$

Datenübertragung ohne Taktleitung

Takt- und Phasenbestimmung aus den Daten

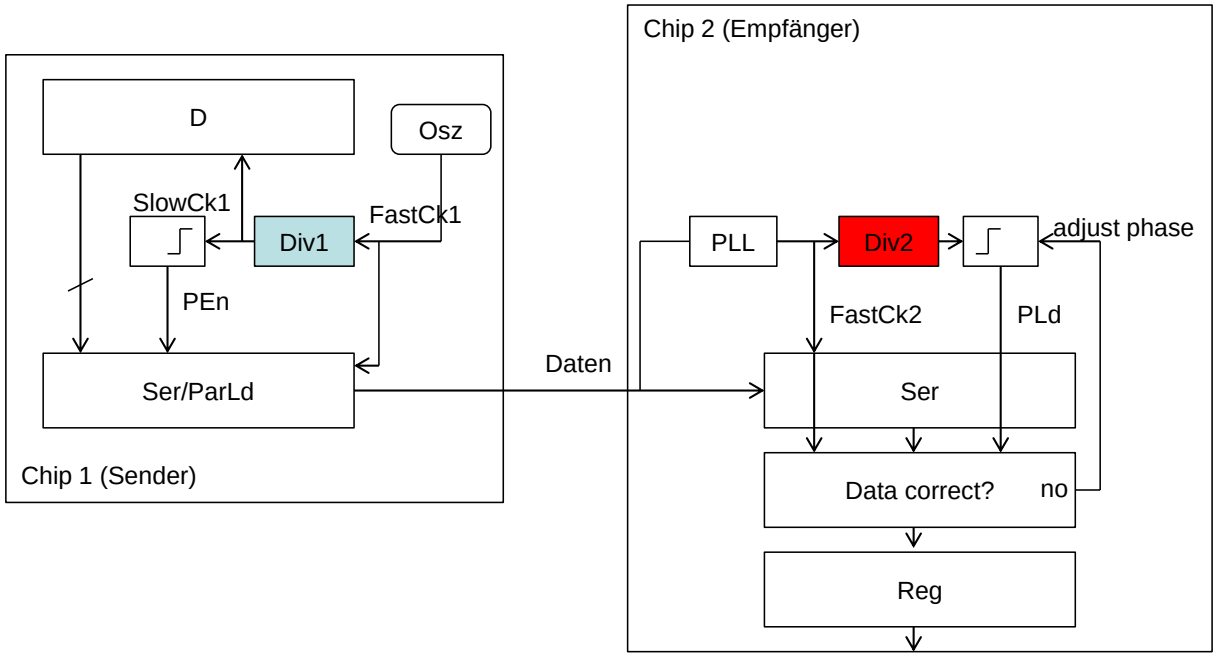

Abbildung 39: Datenübertragung ohne Taktleitung

Bei der Übertragung auf große Entfernungen ist es Sinnvoll Takt (schnellen und langsamen) aus dem empfangenen Signal (Bit-Serie) zu bestimmen. Dieser Takt wird für Abtasten des Empfangssignals benutzt.

Abbildung 39 zeigt Blockschaltplan vom Sender- und Empfänger-Chip.

Sender Chip enthält Digitalteil (D), der die Daten generiert, Oszillator, Frequenzteiler (Div1), Flankendetektor und Serialisierer.

Taktrückgewinnung

Empfänger enthält PLL, die den schnellen Takt aus der empfangenen Bit-Serie bestimmt, Frequenzteiler (Div2), für die Erzeugung vom langsamen Takt, Flankendetektor, Komparator und Register für die Daten. Der Komparator vergleicht empfangene Bitsequenz mit einem sync-Wort (Kontrollwort, Testmuster, training pattern, spacing word, usw.). Die Phase des langsamen Takts wir so lange verschoben, bis das sync-Wort erkannt wird.

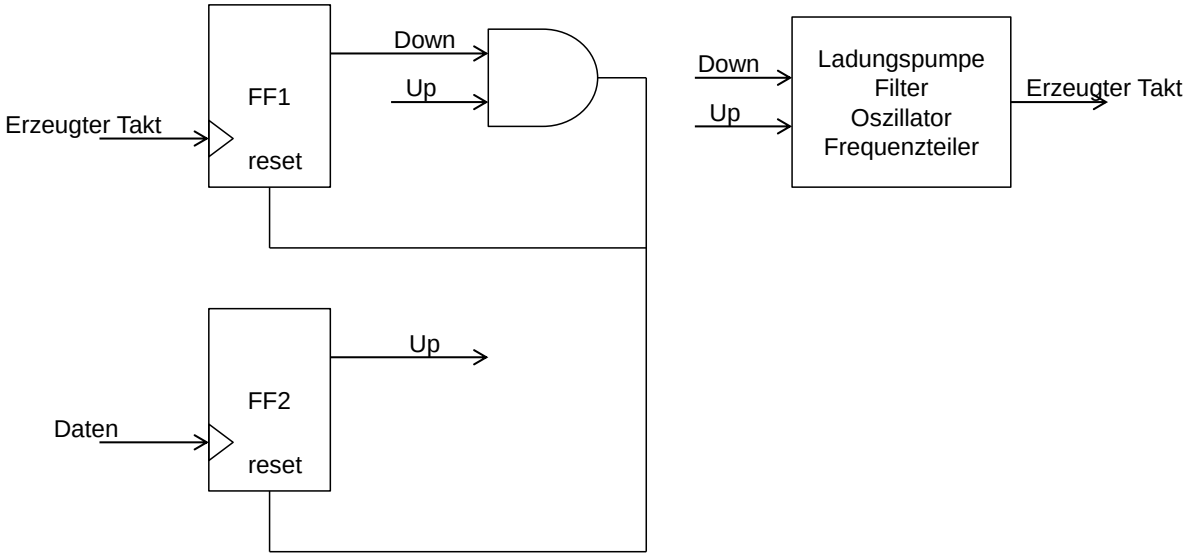

Abbildung 40: Normaler Phasenkomparator und PLL

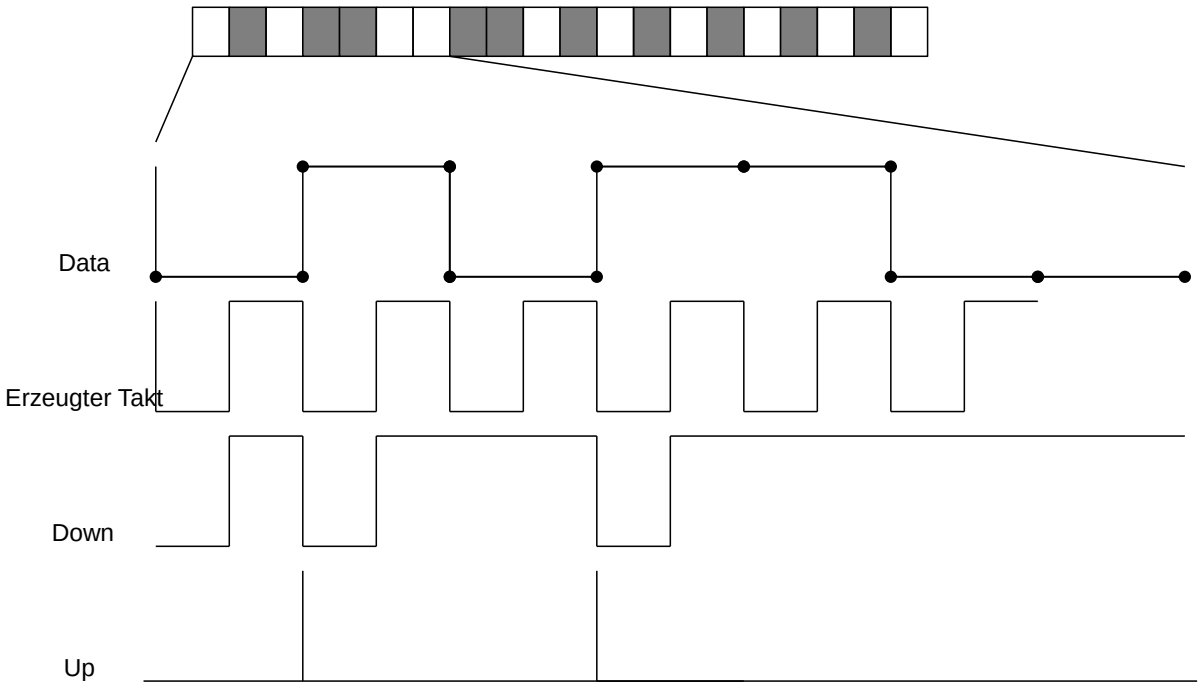

Abbildung 41: Normaler kann das Taktsignal nicht richtig bestimmen

Die PLL für die Taktrückgewinnung (clock data recovery) basiert auf der beschriebenen Phasenregelschleife (Abbildung 40). Die Taktrückgewinnung funktioniert nur wenn das Empfangssignal genug viele Signalflanken aufweist. Das kann z.B. durch den Einsatz eines Leitungscodes erreicht werden.

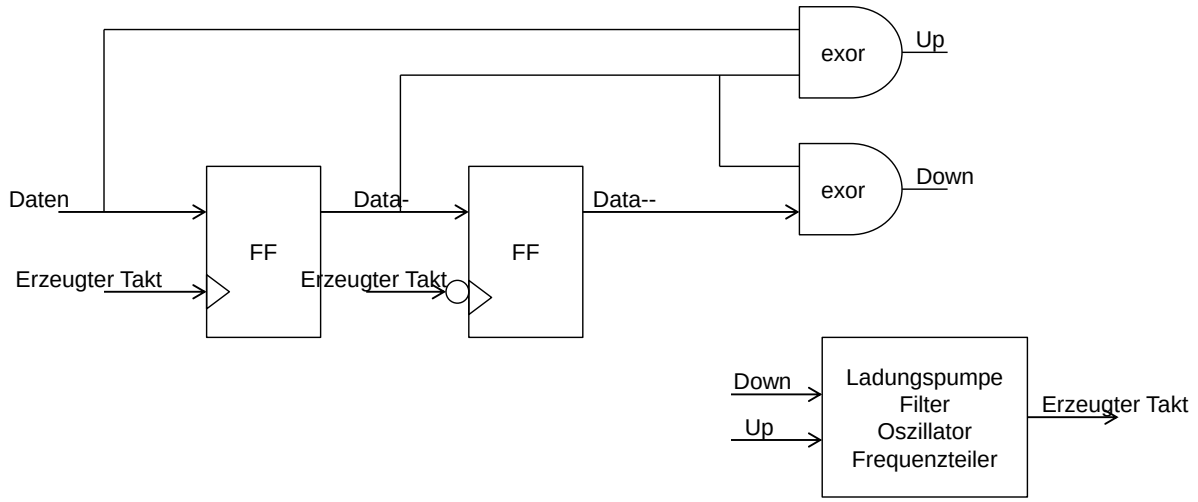

Abbildung 42: Linearer Phasendetektor

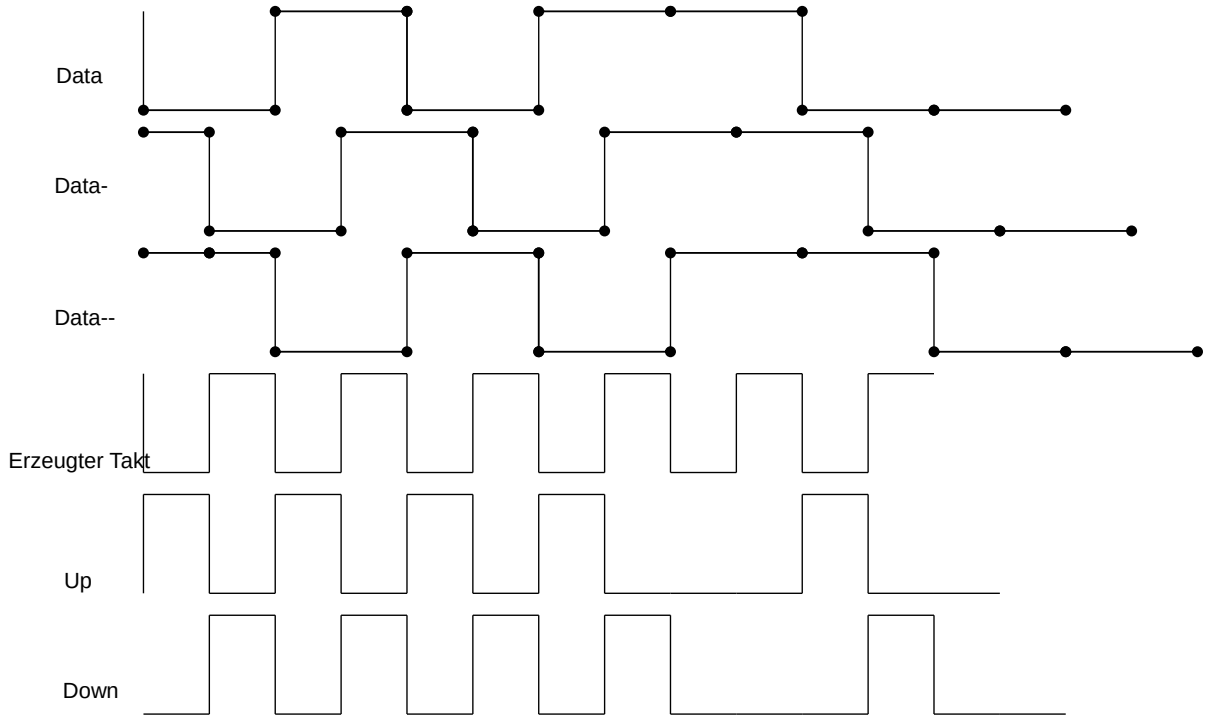

Abbildung 43: Takt wird richtig erzeugt

Der einfache Phasenkomparator würde nicht richtig funktionieren. Abbildung 41 zeigt das empfangene Signal und einen perfekt alignierten Takt. Down Signal ist zu lang.

Abbildung 42 zeigt den Hogge-Phasenkomparator (linearen Phasenkomparator). Diese Schaltung kann für die Taktrückgewinnung verwendet werden.

Taktrückgewinnung – Wikipedia

Bestimmung der Phase

Wir werden nun das Prinzip eines Leitungscodes besprechen.

Erste Aufgabe des Leitungscodes ist es das zu übertragende Signal spektral zu formen. Es wird z.B. DC Anteil unterdruckt.

Beispiel ist die Manchester-Kodierung.

Bei der Manchesterkodierung entspricht eine Null-Eins-Folge einer logischen Null (steigende Flanke), eine Eins-Null-Folge (fallende Flanke) einer logischen Eins.

Ein weiteres Beispiel sind Block-Codes.

P Bits der Eingangssequenz werden zusammengefasst und zu einem Block der Länge q > p abgebildet. Die Abbildung wird so gewählt, dass die Ausgangssequenz etwa gleiche Zahl von Einsen und Nullen hat, unabhängig von Eingangsdaten. Manche q-Bit Symbole werden als Kontrollsymbole (Kommawort) benutzt. Diese Symbole sind keine Abbildung eines Eingangssymbols. Die Kontrollsymbole sollen auch nicht zwischen zwei Datensymbolen zufällig entstehen. Die Kontrollsymbole können für die Bestimmung von Phase benutzt werden.

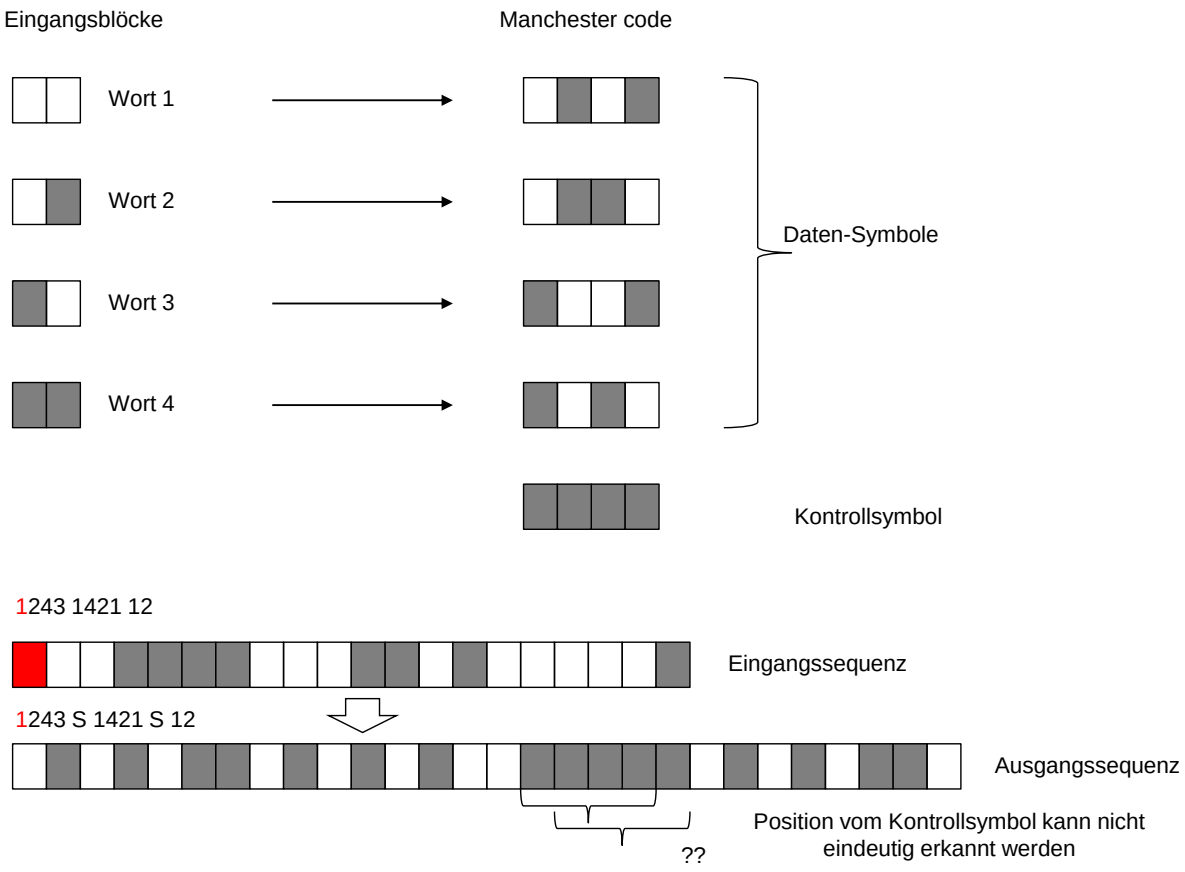

Abbildung 44: Herleitung eines Leitungscodes – erster Versuch. Das Kontrollsymbol kann nicht erkannt werden.

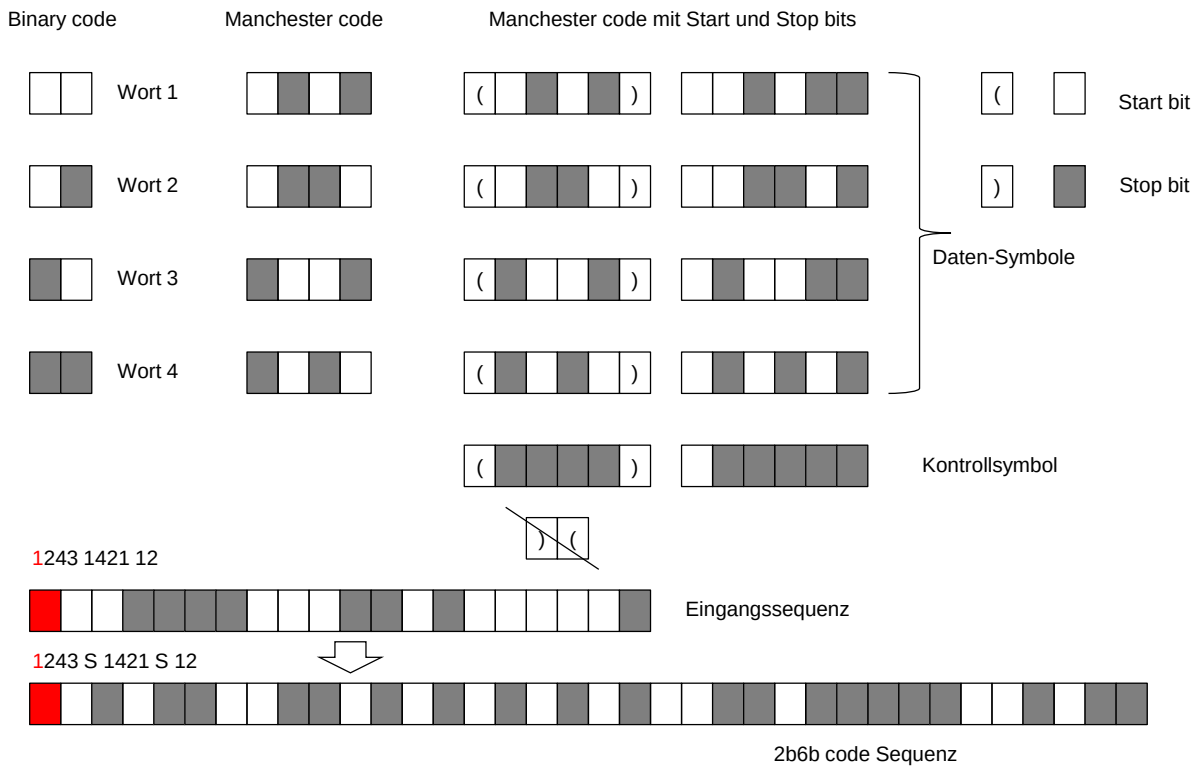

Abbildung 45: Ein einfacher Leitungscode

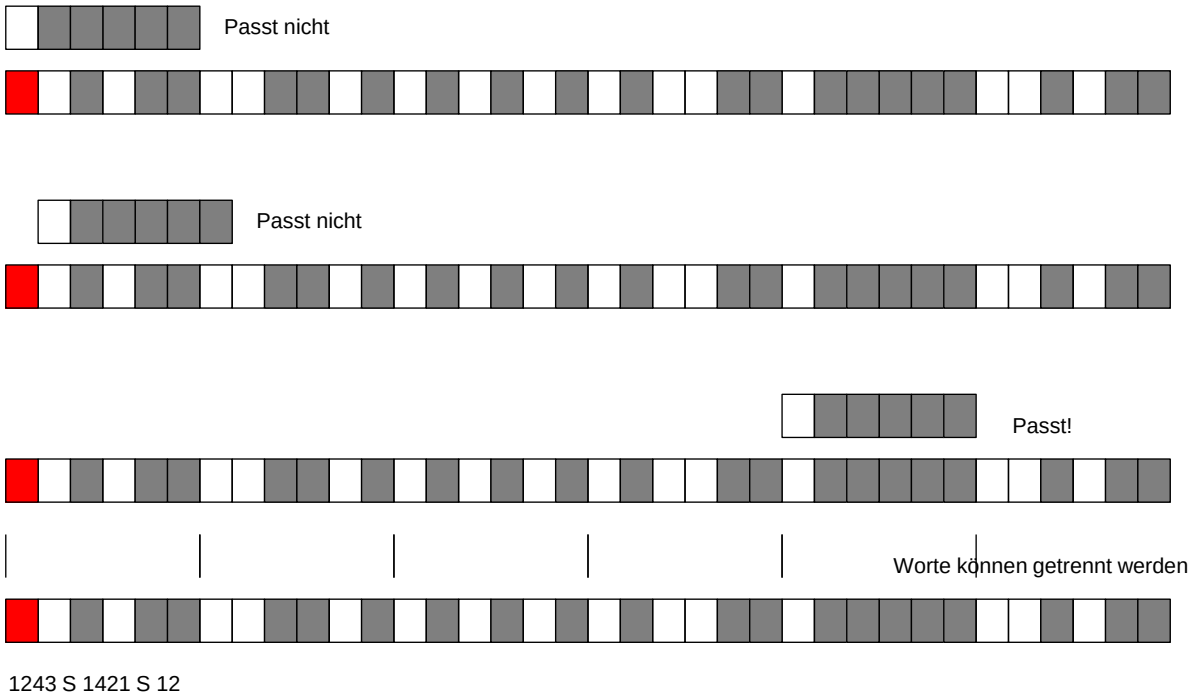

Abbildung 46: Kontrollsymbol kann richtig erkannt werden, die Worte können erkannt werden

Abbildung 44 zeigt wie man einen einfachen Leitungscode mit einem Kontrollsymbol herleiten könnte. Wir benutzen für 2-bit Daten 4-bit Manchester-Code. Wir fügen noch ein Kontrollsymbol hinzu. Wenn wir ein Kontrollsymbol zwischen den 4-bit Datenworten senden, können wir die Position des Kontrollsymbols nicht richtig erkennen.

Eine bessere Lösung ist in Abbildung 45. Wir verwenden noch die Start- und Stop-Bits. Jetzt ist es möglich die Position des Kontrollsymbols in der Bitsequenz richtig zu erkennen. Wichtig ist es dabei dass die Kombination Stop-Start-bit nicht im Kontrollcode vorkommt.

Leitungscode – Wikipedia

Beispiel eines Leitungscodes ist 8b10b Code, der in Gigabit Ethernet verwendet wird. Abbildung 47 zeigt Sender und Empfänger für 8b10b kodierte Bitsequenz. Ein Kommawort (Kontrollcode) wird für dir Bestimmung der Phase verwendet.

Einige Links:

8b/10b encoding – Wikipedia

Chuck Benz's ASIC/FPGA pages

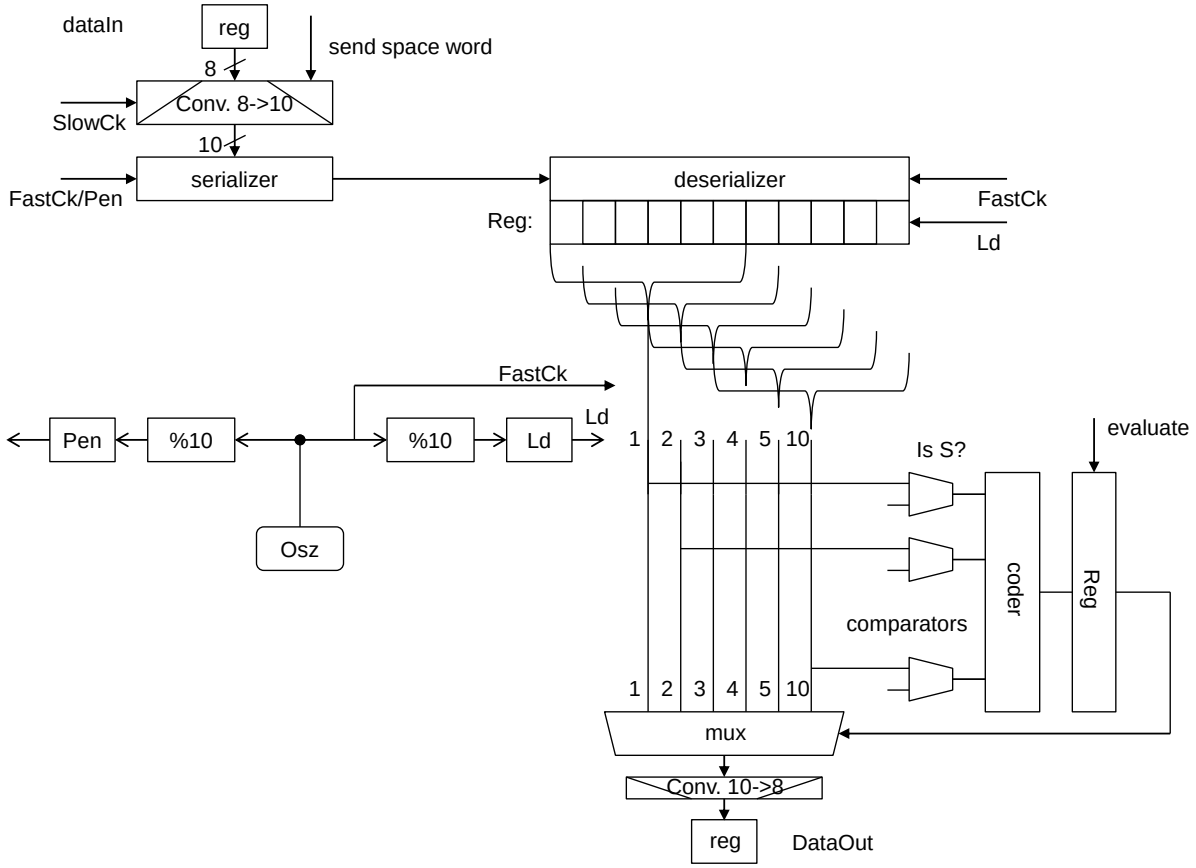

Abbildung 47: Sender und Empfänger für 8b10b kodierte Daten

Eine weitere Methode für CDR und die Trennung von Blöcken basiert auf Scrambling. Durch die Verwendung von PSRG wird das Sendesignal pseudozufällig umformt und damit im Mittel hinreichend viele Signalflanken für die Taktrückgewinnung zur Verfügung gestellt.

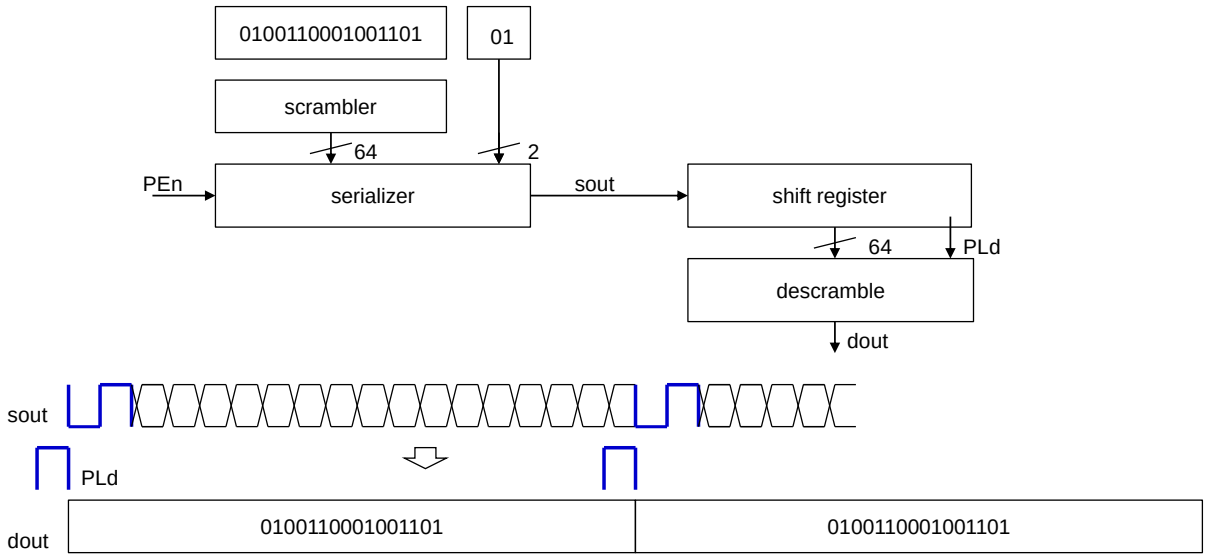

Abbildung 48: 64b66b Code

Ein Beispiel ist die 64b66b Code der in 10-Gbit Ethernet verwendet wird.

[Xilinx SP011 Aurora 64B/66B Protocol Specification, standards specification]

64 Daten-Bits werden mit einem multiplikativen LFSR basierten Scrambler verwürfelt. An die verwürfelten Bits werden zwei feste Kontrollbits angehängt (Abbildung 48). Die 66 Bit werden serialisiert und gesendet. Die zwei Kontrollbits können im Sender erkannt werden da das die einzigen Bits, sind die sich nicht ständig verändern. Auf diese Weise kann die richtige Phase des PLd Signals im Empfänger bestimmt werden.

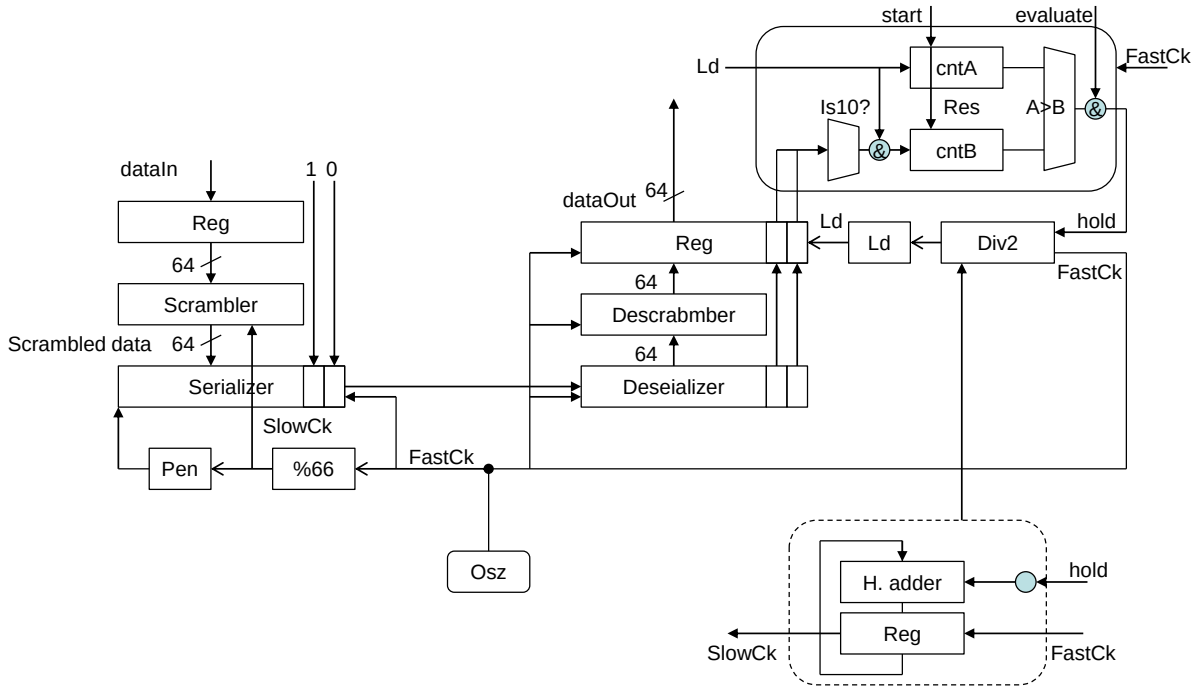

Abbildung 49: Sender und Empfänger für 64b66b kodierte Daten

Abbildung 49 zeigt Sender und Empfänger für 64b66b kodierte Bitsequenz.