# Timing Constraints

Richten Sie die Umgebung für die statische Timing-Analyse ein. Die Angabe der richtigen Einschränkungen ist wichtig für die Analyse der STA-Ergebnisse. Die Entwurfsumgebung sollte genau angegeben werden, damit die STA-Analyse alle Zeitprobleme im Entwurf identifizieren kann. Vorbereiten der STA, Einrichten von Uhren, Festlegen von E / A-Timing-Eigenschaften sowie Angeben falscher Pfade und Mehrradpfade.

## Was ist die STA-Umgebung?

Die

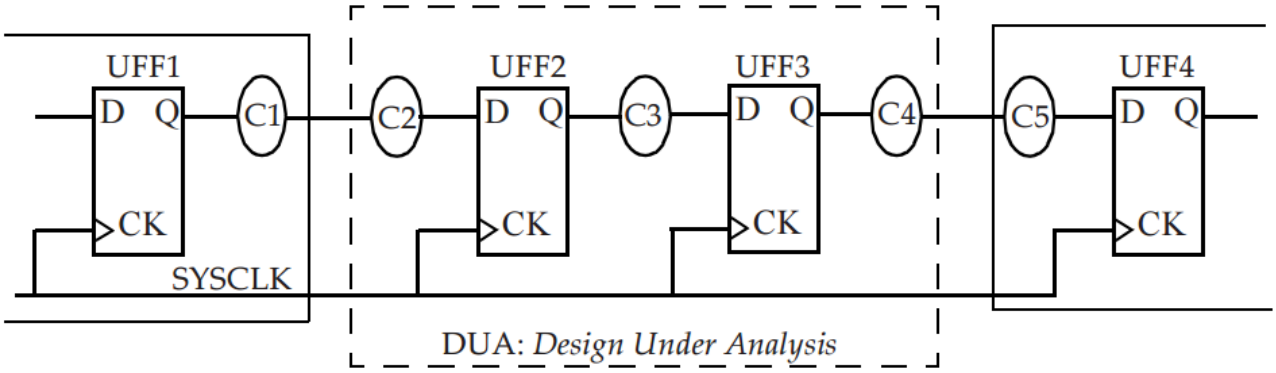



*Abbildung 1: A synchronous design*

meisten digitalen Designs sind synchron, wenn die aus dem vorherigen Taktzyklus berechneten Daten in den Flip-Flops an der aktiven Taktflanke zwischengespeichert werden. Stellen Sie sich ein typisches synchrones Design vor, das in Abbildung 1 dargestellt ist. Es wird davon ausgegangen, dass Design Under Analysis (DUA) mit anderen synchronen Designs interagiert. Dies bedeutet, dass DUA Daten von einem getakteten Flip-Flop empfängt und Daten an ein anderes getaktetes Flip-Flop außerhalb von DUA ausgibt.

Um eine STA für diesen Entwurf durchzuführen, müssen Takte für die Flip-Flops und Zeiteinschränkungen für alle Pfade angegeben werden, die in den Entwurf führen, und für alle Pfade, die den Entwurf verlassen.

Das Beispiel in 1 nimmt an, dass es nur einen Takt gibt und C1, C2, C3, C4 und C5 Kombinationsblöcke darstellen. Die Kombinationsblöcke C1 und C5 liegen außerhalb des zu analysierenden Entwurfs.

In einem typischen Entwurf kann es mehrere Takte mit vielen Pfaden von einer Taktdomäne zu einer anderen geben. In den folgenden Abschnitten wird beschrieben, wie die Umgebung in solchen Szenarien angegeben wird.

## 2. Angeben von Clocks

Um eine Uhr zu definieren, müssen wir die folgenden Informationen bereitstellen:

*i. Clock Source*: Dies kann ein Entwurfsport oder ein Pin einer Zelle innerhalb des Entwurfs sein (normalerweise ist dies Teil einer Taktgenerierungslogik).

*ii. Period*: Periodendauer der Clock.

*Iii. Duty Cycle*: high Dauer (positive Phase) und low Dauer (negative Phase).

*iv. Edge times*: Zeiten für steigende und fallende Flanke.

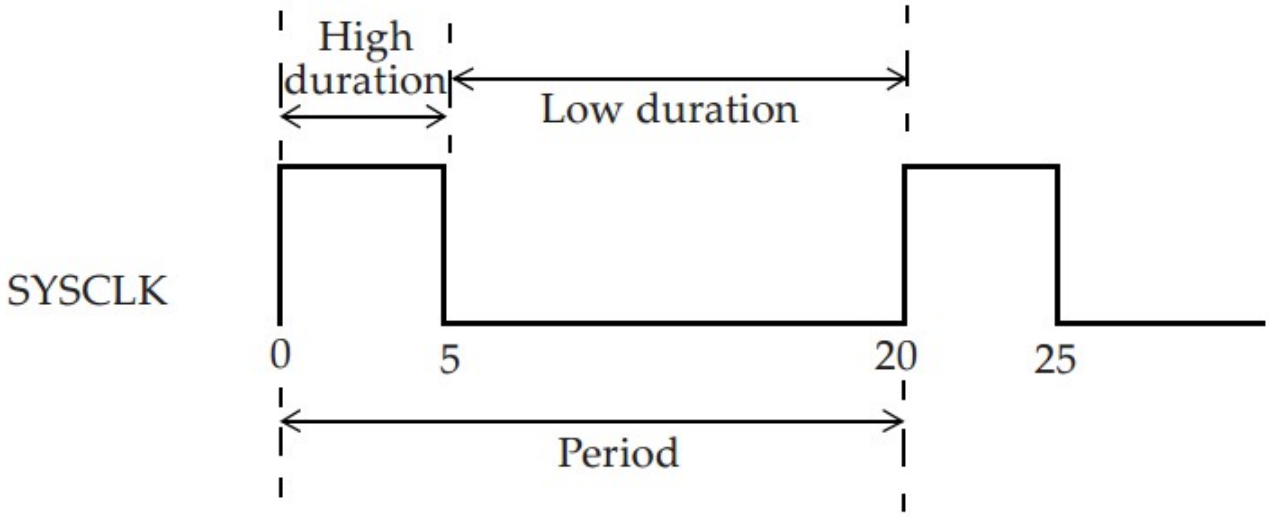



*Abbildung 2: A clock definition*

Abbildung 2 zeigt die grundlegende Definitionen. Durch das Definieren von Takten werden alle internen Zeitsteuerungspfade (alle Flip-Flop-zu-Flip-Flop-Pfade) eingeschränkt; Dies bedeutet, dass alle internen Pfade nur mit den Taktspezifikationen analysiert werden können. Die Taktspezifikation gibt an, dass ein Flip-Flop-zu-Flip-Flop-Pfad einen Zyklus dauern muss. Wir werden später beschreiben, wie diese Anforderung (von einem Zykluszeitpunkt aus) entspannt werden kann.

Hier ist eine grundlegende Clock Spezifikation:

```
create_clock \
-name SYSCLK \
-period 20 \
-waveform { 0 5 } \
[get_ports SCLK]
```

Der Name der Clock lautet **SYSCLK** und wird am Port **SCLK** definiert. Die Periodendauer von SYSCLK wird mit **20** Einheiten angegeben. Die Standardzeiteinheit ist Nanosekunden, wenn keine angegeben wurde. (Im Allgemeinen wird die Zeiteinheit als Teil der Technologiebibliothek angegeben.) Das erste Argument in der Wellenform gibt den Zeitpunkt an, zu dem eine ansteigende Flanke auftritt, und das zweite Argument gibt den Zeitpunkt an, zu dem eine abfallende Flanke auftritt.

In der Wellenformoption kann eine beliebige Anzahl von Flanken angegeben werden. Alle Flanken müssen jedoch innerhalb einer Periode liegen. Die Flankenzeiten wechseln sich ab der ersten ansteigenden Flanke nach dem Zeitpunkt Null ab, dann von einer abfallenden Flanke, dann von einer ansteigenden Flanke und so weiter. Dies bedeutet, dass alle Zeitwerte in der Flankenliste monoton ansteigen müssen.

```
-waveform {time_rise time_fall time_rise time_fall ... }
```

Außerdem muss eine gerade Anzahl von Flanken angegeben werden. Die Wellenformoption gibt die Wellenform innerhalb einer Taktperiode an, die sich dann wiederholt.

Wenn keine Wellenformoption angegeben ist, lautet die Standardeinstellung:

```
-waveform { 0 , period/2 }
```

Hier ist ein Beispiel für eine Taktspezifikation ohne Wellenformspezifikation:

```
create_clock -period 5 [ get_ports SCAN_CLK ]
```

In dieser Spezifikation ist der Name der Clock mit dem Namen des Ports identisch, welcher dann auch **SCAN_CLK** lautet, da keine Option -name angegeben ist.

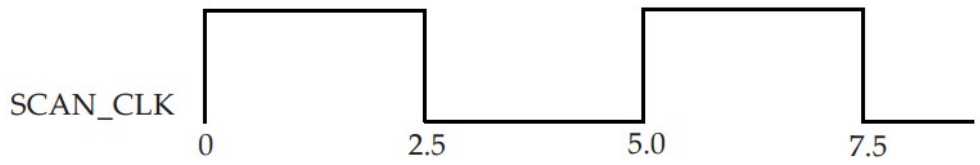



*Abbildung 3: Clock specification example*

Hier ist ein weiteres Beispiel für eine Taktspezifikation, bei der sich die Flanken der Wellenform in der Mitte einer Periode befinden.

```
create_clock -name BDYCLK -period 15 \
-waveform { 5 12 } [get_ports GBLCLK]
```

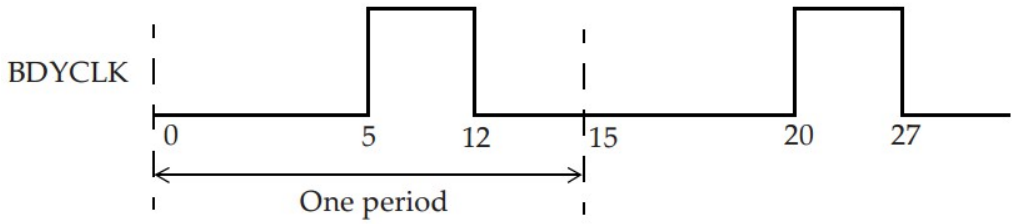



*Abbildung 4: Clock specification with arbitrary edges*

Der Name des Takts lautet **BDYCLK** und wird am Port **GBLCLK** definiert. In der Praxis ist es eine gute Idee, den Taktnamen mit dem Portnamen gleich zu halten.

Hier sind einige weitere Taktspezifikationen:

```
# Siehe Abbildung 5a:
create_clock -period 10 -waveform { 5 10 } [get_ports FCLK]
# Creates a clock with the rising edge at 5ns and the falling edge at 10ns.


# Siehe Abbildung 5b:
create_clock -period 125 \
-waveform { 100 150 } [get_ports ARMCLK]
# Since the first edge has to be rising edge,
# the edge at 100ns is specified first and then the falling
# edge at 150ns is specified. The falling edge at 25ns is
# automatically inferred.
```

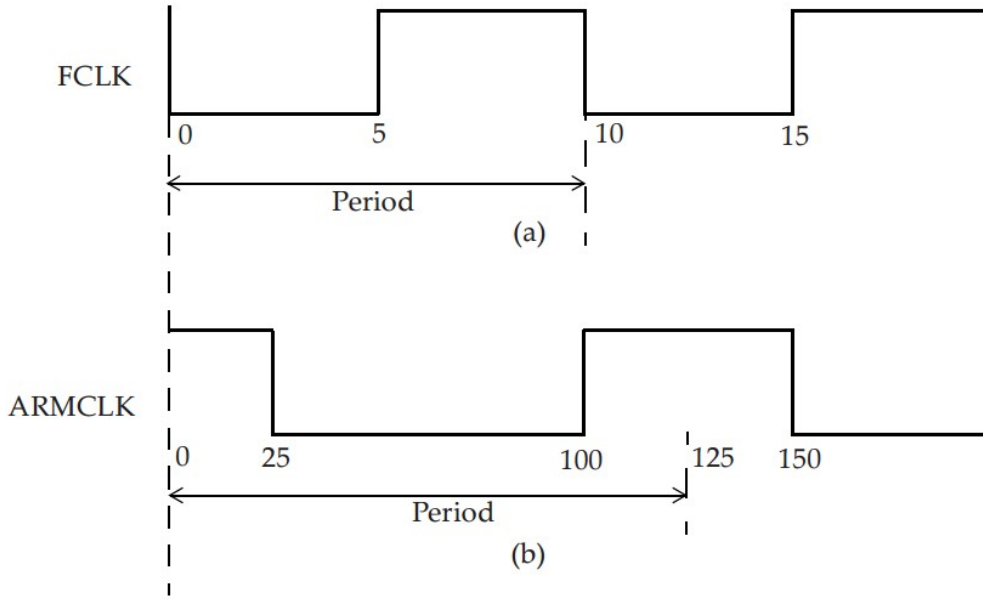



*Abbildung 5: Example clock waveform*

```
# Siehe Abbildung 6a:
create_clock -period 1.0 -waveform { 0.5 1.375 } MAIN_CLK
# The first rising edge and the next falling edge is
# specified. Falling edge at 0.375ns is inferred
# automatically.


# Siehe Abbildung 6b:
create_clock -period 1.2 -waveform { 0.3 0.4 0.8 1.0 } JTAG_CLK
# Indicates a rising edge at 300ps, a falling edge at 400ps
# a rising edge at 800ps and a falling edge at 1ns, this
# pattern is repeated every 1.2ns
```

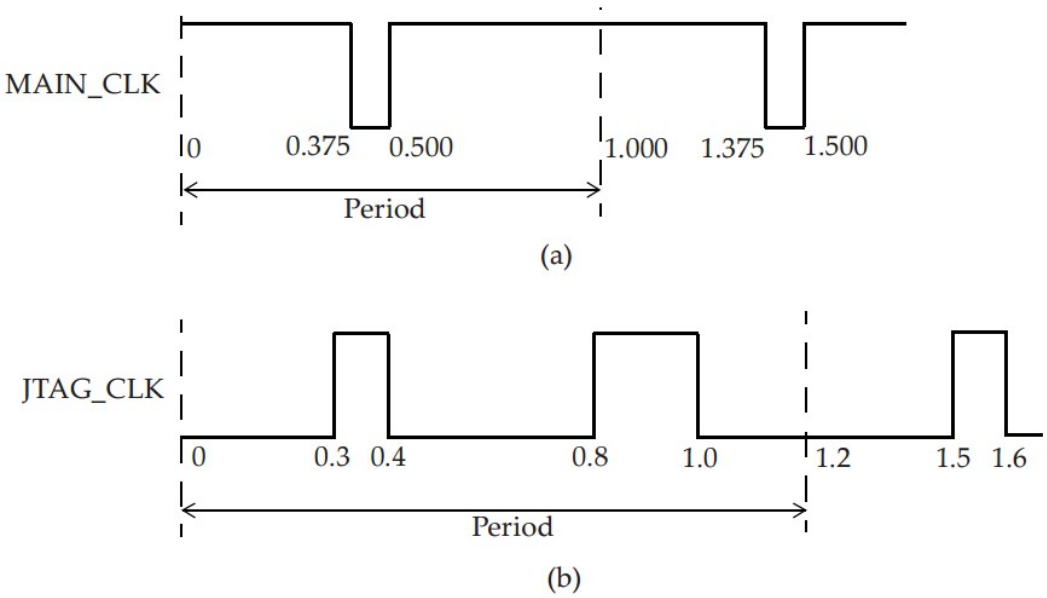


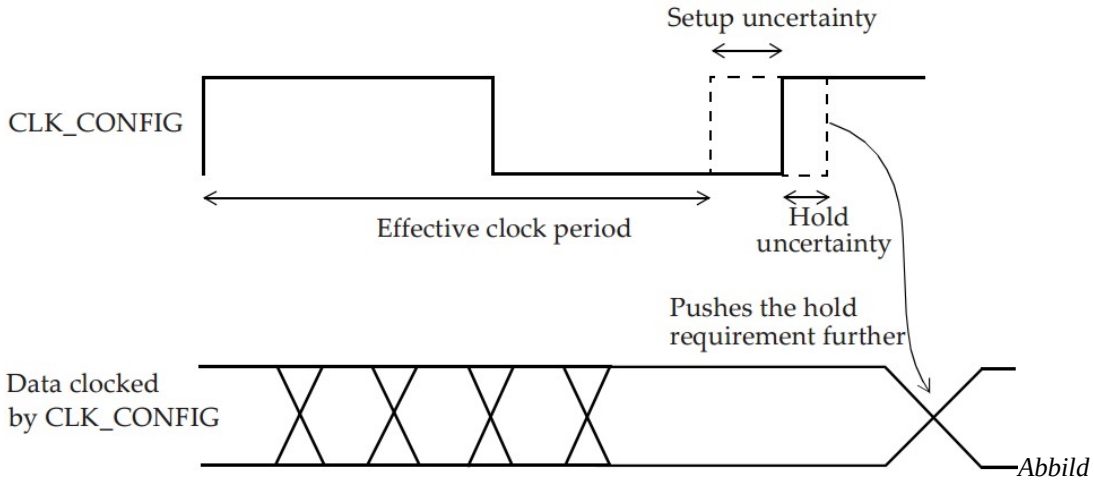

*Abbildung 6: Example with general clock waveform*

## 2.1 Unklarheit über einen Takt

Besteht Unklarheit über eine Taktperiode kann dies unter Verwendung der **set_clock**_uncertainty-Spezifikation angegeben werden. Die Spezifikation kann verwendet werden, um verschiedene Faktoren zu modellieren, die eine effektive Taktperiode verringern können. Diese Faktoren können der Taktjitter und jeder andere Unsicherheit sein, den man für die Timing-Analyse berücksichtigen möchte.

```
set_clock_uncertainty -setup 0.2 [get_clocks CLK_CONFIG]
set_clock_uncertainty -hold 0.05 [get_clocks CLK_CONFIG]
```

Beachten Sie, dass die Taktunsicherheit für „ Setup“ die verfügbare Taktperiode effektiv um den angegebenen Betrag reduziert, wie in Abbildung 7 dargestellt. Bei „Hold“ Prüfungen wird die Taktunsicherheit für „Hold“ als zusätzliche Zeitspanne verwendet, die erfüllt werden muss.




*una 7: Specifying clock uncertainty*

Die folgenden Befehle geben die Unsicherheit an, die auf Pfaden verwendet werden soll, die bestimmte Taktgrenzen überschreiten, die als Inter-Takt-Unsicherheit bezeichnet werden.

```
set_clock_uncertainty -from VIRTUAL_SYS_CLK -to SYS_CLK \
-hold 0.05
set_clock_uncertainty -from VIRTUAL_SYS_CLK -to SYS_CLK \
-setup 0.3
set_clock_uncertainty -from SYS_CLK -to CFG_CLK -hold 0.05
set_clock_uncertainty -from SYS_CLK -to CFG_CLK -setup 0.1
```

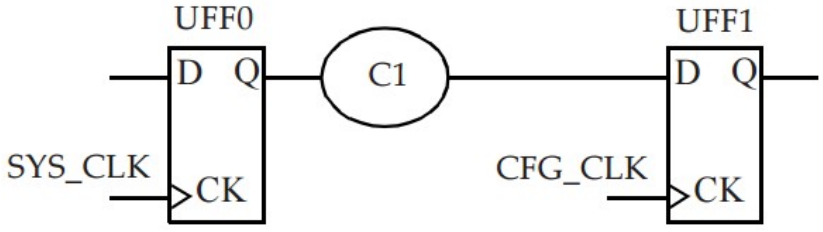



*Abbildung 8: Inter-clock paths*

Abbildung 8 zeigt einen Pfad zwischen zwei verschiedenen Taktdomänen, SYS_CLK und CFG_CLK. Basierend auf den obigen Inter-Clock-Unsicherheitsspezifikationen werden 100 ps als Unsicherheit für Einrichtungsprüfungen und 50 ps als Unsicherheit für Halteprüfungen verwendet.

## 2.2 Takt Latenz

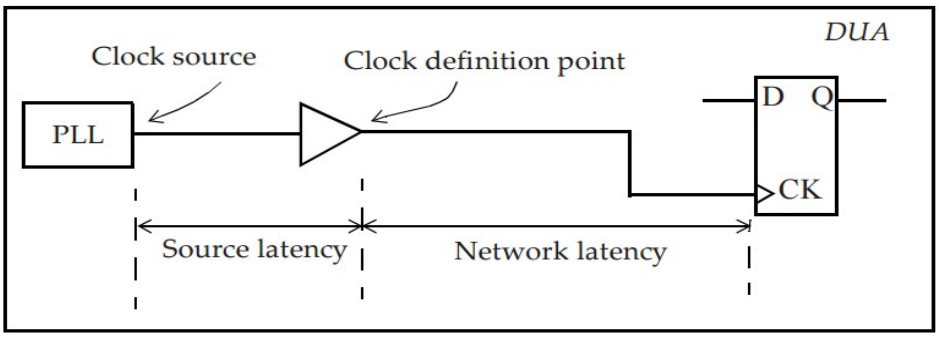
Die Latenz eines Taktes kann mit dem Befehl **set_clock_latency** angegeben werden.

```
# Rise clock latency on MAIN_CLK is 1.8ns:
set_clock_latency 1.8 -rise [get_clocks MAIN_CLK]
# Fall clock latency on all clocks is 2.1ns:
set_clock_latency 2.1 -fall [all_clocks]
# The -rise, -fall refer to the edge at the clock pin of a # flip-flop.
```

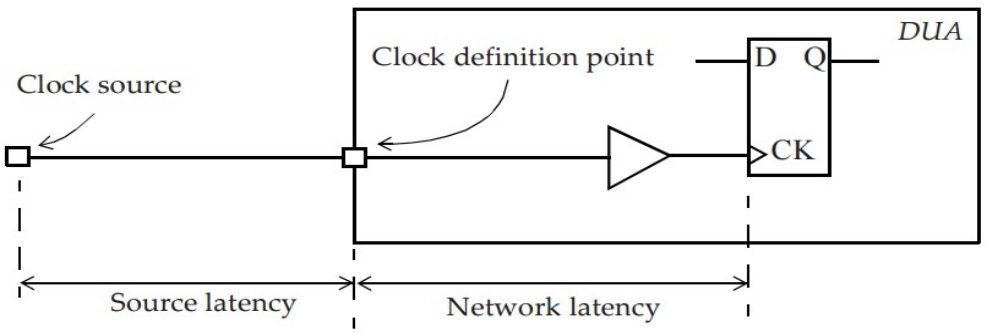
Es gibt zwei Arten von Taktlatenz: Netzwerklatenz und Quelllatenz. Die Netzwerklatenz ist die Verzögerung vom Taktdefinitionspunkt (create_clock) zum Taktstift eines Flipflops. Die Quellenlatenz, auch Einfügungsverzögerung genannt, ist die Verzögerung von der Taktquelle zum Taktdefinitionspunkt. Die Quellenlatenz kann entweder die On-Chip- oder die Off-Chip-Latenz darstellen. Abbildung 9 zeigt beide Szenarien. Die Gesamttaktlatenz am Taktpin eines Flipflops ist die Summe aus Quell- und Netzwerklatenz.

Hier sind einige Beispielbefehle, die die Quell- und Netzwerklatenz angeben.

```
# Specify a network latency (no -source option) of 0.8ns
# for rise, fall, max and min:
set_clock_latency 0.8 [get_clocks CLK_CONFIG]
# Specify a source latency:
set_clock_latency 1.9 -source [get_clocks SYS_CLK]
# Specify a min source latency:
set_clock_latency 0.851 -source -min [get_clocks CFG_CLK]
# Specify a max source latency:
set_clock_latency 1.322 -source -max [get_clocks CFG_CLK]
```




(a) On-chip clock source.




(b) Off-chip clock source.

*Abbildung 9: Clock latency*

## 3. Generierte Takte

Ein erzeugter Takt ist ein vom einer Haupttakt abgeleiteter Takt. Ein **Haupttakt** ist ein Takt, der anhand der **create_clock-Spezifikation** definiert wird.

Wenn ein neuer Takt in einem Entwurf generiert wird, der auf einem Haupttakt basiert, kann der neue Takt als generierter Takt definiert werden. Wenn es beispielsweise eine durch 3 dividierende Schaltung für einen Takt gibt, würde man eine generierte Taktdefinition am Ausgang dieser Schaltung definieren. Diese Definition ist erforderlich, da STA nicht weiß, dass sich die Taktperiode am Ausgang der Divide-by-Logik geändert hat, und was noch wichtiger ist, wie die neue Taktperiode ist. Fig. 10 zeigt ein Beispiel eines erzeugten Takts, der eine Division durch 2 des Haupttakts **CLKP** ist.

```
create_clock -name CLKP 10 [get_pins UPLL0/CLKOUT]
# Create a master clock with name CLKP of period 10ns
# with 50% duty cycle at the CLKOUT pin of the PLL.
create_generated_clock -name CLKPDIV2 -source UPLL0/CLKOUT -divide_by 2 [get_pins
UFF0/Q]
# Creates a generated clock with name CLKPDIV2 at the Q
# pin of flip-flop UFF0. The master clock is at the CLKOUT
# pin of PLL. Period of generated clock is double that of
```

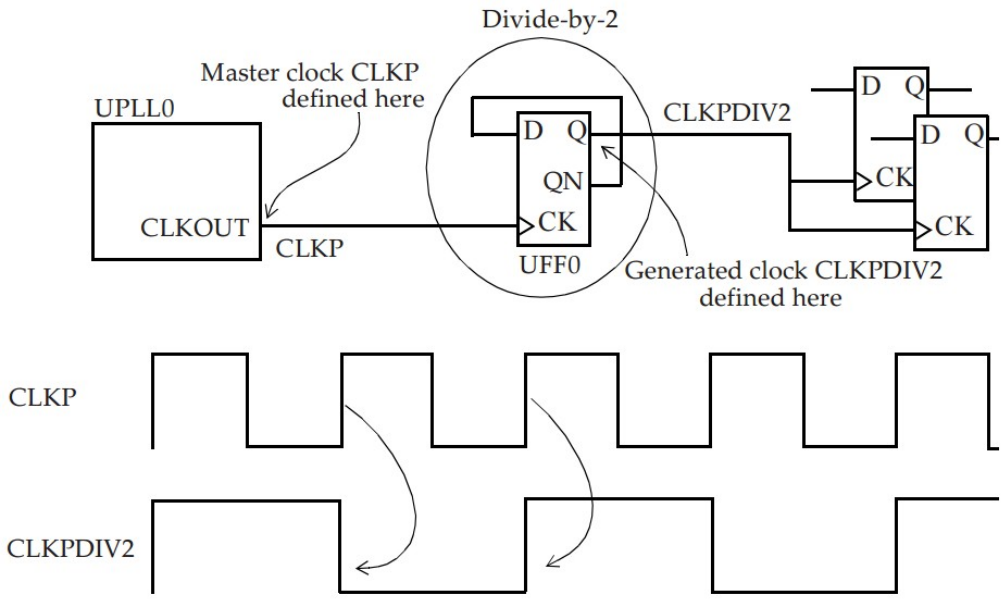



*Abbildung 10: Generated clock at output of divider*

Kann ein neuer Takt (ein Haupttakt) am Ausgang des Flip-Flops anstelle eines erzeugten Taktes definiert werden? Die Antwort lautet ja, es gibt jedoch einige Nachteile. Durch das Definieren eines Haupttaktes anstelle eines generierten Taktes wird eine neue Taktdomäne erstellt. Dies ist im Allgemeinen kein Problem, außer dass beim Einrichten der Einschränkungen für STA mehr Taktdomänen zu behandeln sind. Das Definieren des neuen Taktes als generierten Takt erzeugt keine neue Taktdomäne, und der erzeugte Takt wird als in Phase mit seinem Haupttakt betrachtet. Für den erzeugten Takt müssen keine zusätzlichen Einschränkungen entwickelt werden. Daher muss versucht werden, einen neuen intern generierten Takt als generierten Takt zu definieren, anstatt ihn als einen anderen Haupttakt festzulegen.

Ein weiterer wichtiger Unterschied zwischen eines Haupttaktes und eines erzeugten Taktes ist der Begriff des Taktursprungs. Bei einem Haupttakt liegt der Ursprung des Taktes am Definitionspunkt des Haupttaktes. In einem erzeugten Takt ist der Taktursprung der des Haupttaktes und nicht der des erzeugten Taktes. Dies

impliziert, dass in einem Taktpfadbericht der Startpunkt eines Taktpfads immer der Haupttaktdefinitionspunkt ist. Dies ist ein großer Vorteil eines generierten Taktes gegenüber der Definition eines neuen Haupttaktes, da die Quellenlatenz für den Fall eines neuen Haupttaktes nicht automatisch berücksichtigt wird.

Abb. 11 zeigt ein Beispiel, bei dem der Takt **SYS_CLK** durch den Ausgang eines Flipflops gesteuert wird. Da der Ausgang des Flip-Flops möglicherweise keine Konstante ist, besteht eine Möglichkeit mit dieser Situation umzugehen darin, einen erzeugten Takt am Ausgang der UND-Zelle zu definieren, der mit dem Eingangstakt identisch ist.

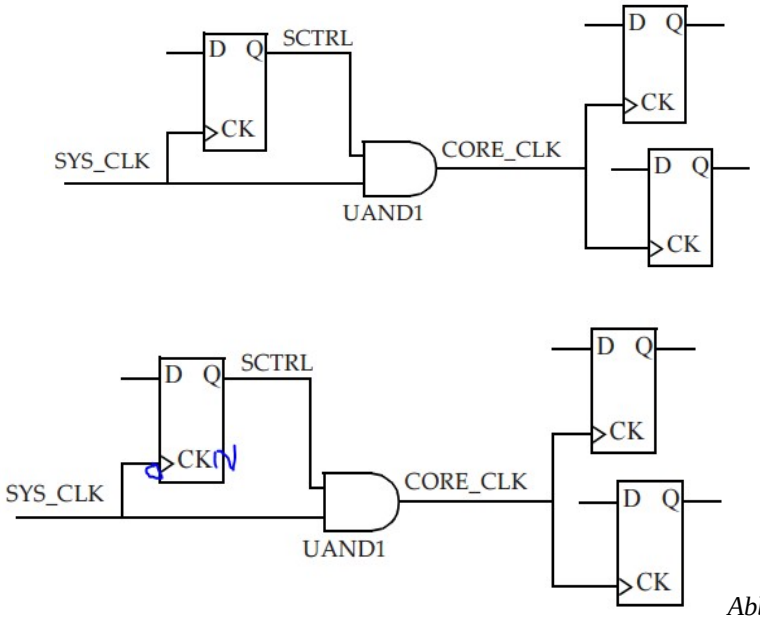



*Abbildung 11: Clock gated by a flip-flop**

* Es sollte sich um CKN im linken FF handeln, oder es würde nicht die Anforderungen an das Hold des Clock Gating erfüllen, Details und Erläuterungen unter Überprüfen des Clock Gating

```
create_clock 0.1 [get_ports SYS_CLK]
# Create a master clock of period 100ps with 50% duty
# cycle.
create_generated_clock -name CORE_CLK -divide_by 1 \
-source SYS_CLK [get_pins UAND1/Z]
# Create a generated clock called CORE_CLK at the output of
# the AND cell and the clock waveform is the same as that
# of the master clock
```

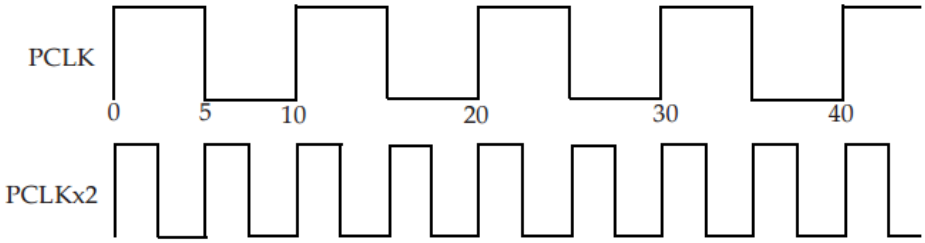



*Abbildung 12: Master clock and multiply-by-2 generated clock*

```
create_clock -period 10 -waveform { 0 5 } [get_ports PCLK]
# Create a master clock with name PCLK of period 10ns
# with rise edge at 0ns and fall edge at 5ns.
create_generated_clock -name PCLKx2 \
-source [get_ports PCLK] \
-multiply_by 2 [get_pins UCLKMULTREG/Q]
# Creates a generated clock called PCLKx2 from the master
# clock PCLK and the frequency is double that of the master
# clock. The generated clock is defined at the output of
# the flip-flop UCLKMULTREG.
```

Beachten Sie, dass sich die Optionen -multiply_by und -divide_by auf die Taktfrequenz beziehen, obwohl in einer Haupttaktdefinition eine Taktperiode angegeben ist.

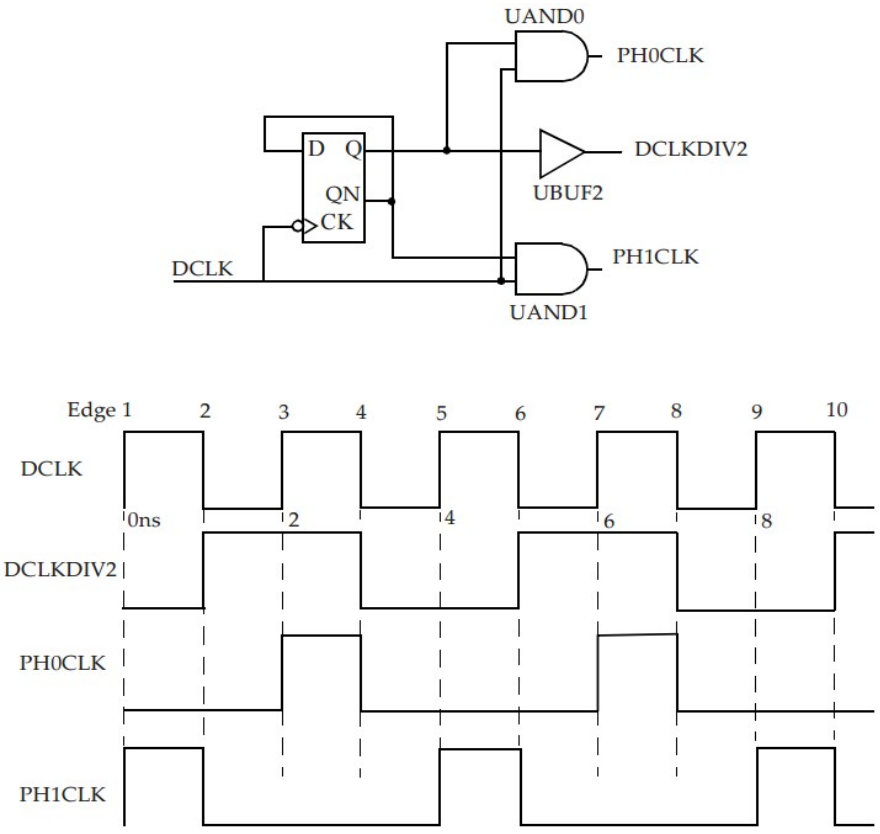



*Abbildung 13: Clock generation*

Abbildung 13 zeigt ein Beispiel generierter Takte. Zusätzlich zu den phasenverschobenen Takten wird ein Divisionstakt durch 2 erzeugt. Die Wellenform der Takte ist ebenfalls in der Abbildung dargestellt.

```
create_clock 2 [get_ports DCLK]
# Name of clock is DCLK, has period of 2ns with a rise edge
# at 0ns and a fall edge at 1ns.
create_generated_clock -name DCLKDIV2 -edges {2 4 6}\
-source DCLK [get_pins UBUF2/Z]
create_generated_clock -name PH0CLK -edges {3 4 7} \
-source DCLK [get_pins UAND0/Z]
create_generated_clock -name PH1CLK -edges {1 2 5} \
-source DCLK [get_pins UAND1/Z]
```

**Taktlatenz für generierte Takte**

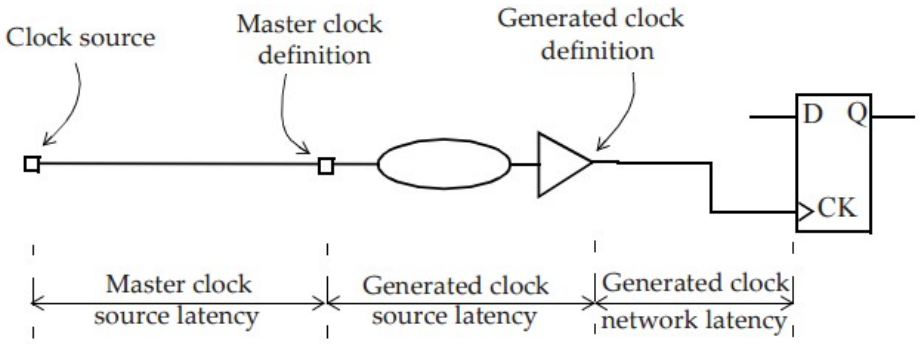



*Abbildung 14: Latency on generated clock*

Ein erzeugter Takt kann einen anderen erzeugten Takt als Quelle haben, dh man kann Takte vom einem erzeugten Takt erzeugen, und so weiter, jedoch kann ein erzeugter Takt nur einen Haupttakt haben.

**Typisches Taktgenerierungsszenario**

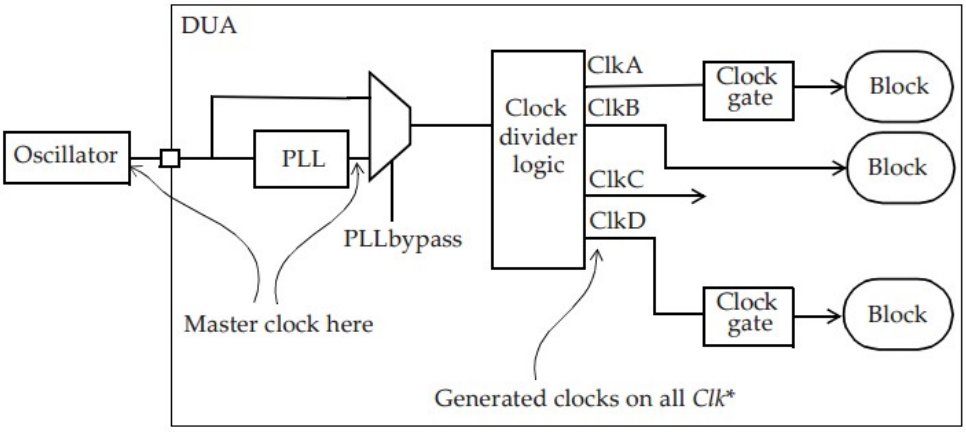



*Abbildung 15: Clock distribution in a typical ASIC*

Abbildung 15 zeigt ein Szenario, wie eine Taktverteilung in einem typischen ASIC aussehen kann. Der Oszillator befindet sich außerhalb des Chips und erzeugt einen Niederfrequenztakt (typisch 10-50 MHz), der von der On-Chip-PLL als Referenztakt verwendet wird, um einen Hochfrequenztakt mit niedrigem Jitter (typisch 200-800 MHz) zu erzeugen. Dieser PLL-Takt wird dann einer Taktteilerlogik zugeführt, die die erforderlichen Takte für den ASIC erzeugt.

In einigen Zweigen der Taktverteilung kann es Taktgatter geben, die verwendet werden, um den Takt auf einen inaktiven Teil des Entwurfs auszuschalten, um bei Bedarf Energie zu sparen. Die PLL kann auch einen Multiplexer am Ausgang haben, so dass die PLL bei Bedarf umgangen werden kann. Ein Haupttakt wird für den Referenztakt am Eingangspin des Chips definiert, an dem er in das Design eintritt, und ein zweiter Haupttakt wird am Ausgang von PLL definiert. Der PLL-Ausgangstakt hat keine Phasenbeziehung zum Referenztakt. Daher sollte der Ausgangstakt kein erzeugter Takt des Referenztakts sein. Höchstwahrscheinlich werden alle von der Taktteilerlogik erzeugten Takte als erzeugte Takte des Haupttakts am PLL-Ausgang angegeben.