

## 11 Laborübung 11:

### 11.0.1 Was sie nach dieser Laborübung können:

1. Das Konzept für einen Universal Asynchronous Receiver nachvollziehen
2. Das Konzept mit einem Zeitverlaufdiagramm überprüfen
3. Einen Moore Automaten gemäss Zustandsdiagramm in VHDL beschreiben

In diesem Projekt werden Sie das Schieberegister aus Labor 10 und den Edge Detector aus Labor 7 mit weiteren Blöcken zu einem „Universal Asynchronous Receiver“ ausbauen. An das DE2-115 Board wird ein Bluetooth Empfänger angesteckt, dessen Ausgang ein serielles Signal an den FPGA weiter gibt. Mit einer Handy App generieren Sie ASCII Zeichen, die über Bluetooth an den Bluetooth Empfänger übertragen werden.

### 11.1 Infrastrukturblock entwerfen

1. Eröffnen Sie ein neues Projektverzeichnis mit den Unterdirektorien *source* und *quartus*.
2. Laden Sie sich vom Olaf die Datei *bluetooth\_top.bdf* herunter und installieren Sie die Datei im *Source* Verzeichnis
3. Starten Sie Quartus und benutzen Sie den „New Project Wizard“, um ein neues Projekt aufzusetzen. Vergessen Sie nicht, dass die Quartus Arbeitsverzeichnisse im Unterordner *quartus* untergebracht werden sollen. Installieren Sie als erste Datei *bluetooth\_top.bdf*.
4. Entwerfen Sie eine VHDL Entity mit dem Namen „Infrastructure“ in der Datei „*infrastructure.vhd*“. Hierfür können Sie z.B. den Edge Detector als Vorlage aus dem vorherigen Lab kopieren, anpassen und dann den Dateinamen und den Entity Namen abändern. Vergessen Sie nicht, dass die *.vhd* Dateien im Ordner *source* untergebracht werden sollen.  
Die Datei Infrastruktur nimmt ein Eingangssignal *serdata\_in* auf. Signal „*serdata\_in*“ soll in diesem Block durch zwei D-FF hindurchgeführt werden und damit das Eingangssignal synchronisieren. Das synchronisierte Ausgangssignal soll *serdata\_snyc* heissen. Der Block soll auch ein Takt Signal erhalten, dass an den 50 MHz Takt angeschlossen werden soll.
5. Generieren Sie von „*infrastructure.vhd*“ ein „Board Symbol File“ (*infrastructure.bsf*). Vergessen Sie nicht, die *.bsf* Dateien im *source* Ordner unterzubringen.
6. Fügen Sie „*infrastructure.vhd*“ zum Projekt hinzu und bauen Sie den Block *infrastructure* in *bluetooth\_top* ein.
7. Setzen Sie *bluetooth\_top.bdf* als Top Level Datei. Analysieren Sie diese Datei. Überprüfen Sie die synthetisierte Struktur auf Latches mit dem RTL Viewer.

## 11.2 Baud Tick Generator

Der Baud Tick Generator erzeugt die Abtastpulse der seriellen Datenbits zum richtigen Zeitpunkt. Der richtige Zeitpunkt ist genau in der Mitte der übertragenen Datenbits (siehe Abb.11.3). Nach dem ein Start Bit detektiert wurde, vergehen eineinhalb Baud Tick Perioden, bis zur Mitte des übertragenen ersten Bits.

Bis zum nächsten übertragenen Bit wird genau eine Periode der Baud Rate vergehen. Die Taktfrequenz ist 50 MHz, die Baudrate des Signals ist 115'200 Bits pro Sekunde.

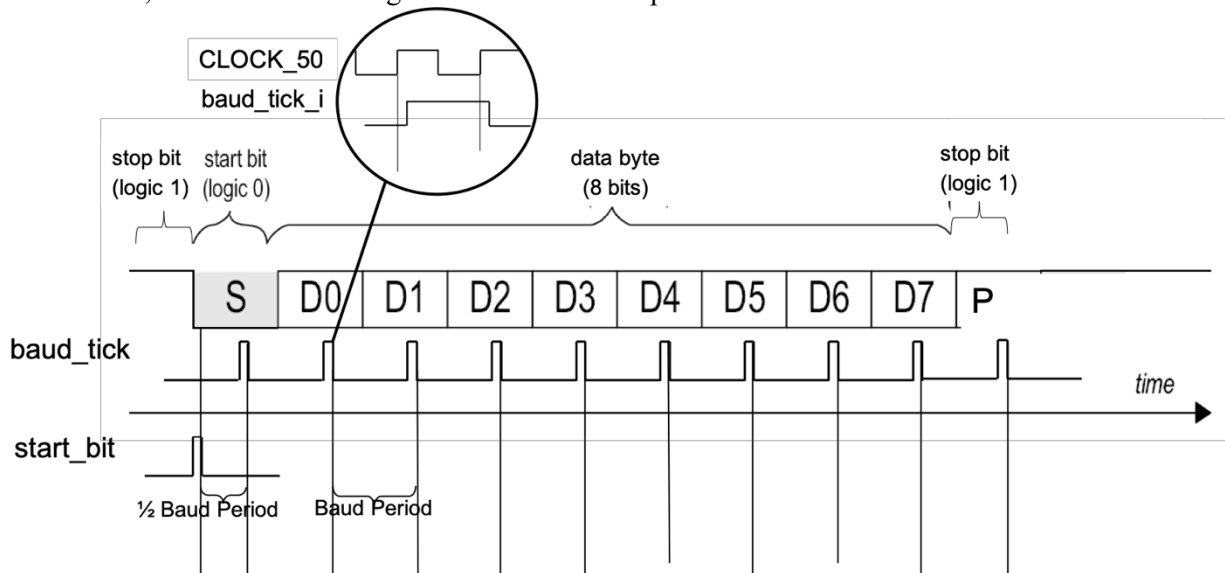


Abbildung 11.1: Baud Tick Generierung

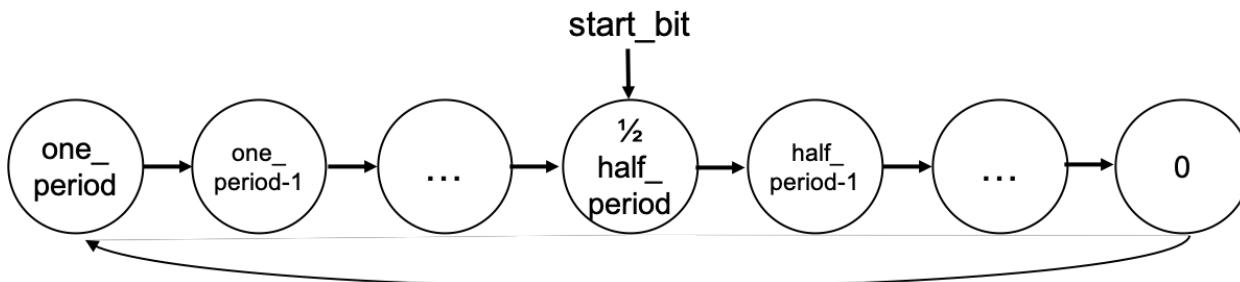


Abbildung 11.2: Baud Tick Counter

1. Berechnen Sie den Wert *half\_period* des Baud Tick Counters, der in den Zähler geladen werden muss, nach dem ein Start Bit detektiert wurde und den Wert *one\_period* des Zählers, der nach Ablauf des Baud Tick Counters geladen werden soll.
2. Schreiben Sie den VHDL Code *baud\_tick.vhd* des Baud Tick Generators.
3. Setzen Sie *baud\_tick.vhd* als Top Level, analysieren Sie den Code mit Quartus und erstellen sie *baud\_tick.bsf*. Prüfen Sie nach, dass keine unerwünschten Latches entstanden sind.

## 11.3 Überprüfung der Eingangsstufen

1. Kopieren Sie sich den Edge Detector von Labor 10 in das *Source* Verzeichnis. Bauen Sie Edge Detector, Start Bit Detector und Baud Tick Generator wie in Abbildung 11.6 in *bluetooth\_top*

zusammen. Shift Register und Display Output FSM wird erst in der nächsten Stufe erstellt.

2. Setzen Sie *bluetooth\_top* als Toplevel.
3. Weisen sie die DE2-115 pins zu (Import assignments).
4. Synthetisieren Sie im Quartus den Entwurf und prüfen Sie mit dem RTL Viewer, ob die Schaltung plausibel aussieht.
5. Prüfen Sie im Schaltplan oder im Pin Planer, ob alle DE2 Pins richtig angeschlossen wurden.
6. Installieren Sie auf Ihrem Mobiltelefon die App „bt terminal V4.3“ aus dem App Shop.
7. Stecken Sie den Bluetooth Empfänger auf das DE2 Board, schalten Sie Ihr Entwicklungsboard ein und laden Sie die *.sof* Datei auf das Board.
8. Drücken Sie am Board *Key(0)* um die Schaltung zurückzusetzen.
9. Gehen Sie mit Ihrem Mobiltelefon auf „Einstellungen“ und suchen Sie nach dem Bluetooth Modul. Die Bluetooth Module beginnen alle mit RNBT-xxxx. xxxx steht für die letzten 4 Zeichen der MAC Adresse des Empfängers.
10. Koppeln Sie Ihr Telefon mit dem Bluetooth Modul. Nach erfolgreicher Verbindung muss die LEDG 0 auf Ihrem DE2 Board leuchten.
11. Gehen Sie in der Bluetooth Terminal App auf „Settings“ und setzen Sie „Send hexadezimal“.
12. Starten Sie *Signal Tap* im Quartus und fügen Sie *serdata\_sync*, *edge*, *start\_bit*, *baud\_tick*, die Zähler Bits des Start Bit Detektors und die Zähler Bits des Baud Tick Generators als Beobachtungspunkte ein. Wählen Sie die fallende Flanke des Signals *serdata\_sync* als Trigger für Signal Tap.
13. Laden Sie den Entwurf auf das DE2-115 board. Senden Sie vom Bluetooth Terminal Hex-Zahlen an das Modul Verifizieren Sie die Funktionalität Ihres Entwurfs mit Signal Tap.

## 11.4 Shift Register

Erweitern Sie das Schieberegister aus Lab10 auf 10-bits. Die Taktleitungen der Schieberegister Flip-Flops müssen direkt an den 50 MHz Takt angeschlossen werden.

1. Erstellen Sie das Schieberegister *shiftreg.vhd* in VHDL.
2. Setzen Sie *shiftreg.vhd* als Top Level, analysieren Sie den Schieberegister Code mit Quartus und prüfen Sie nach unbeabsichtigten Latches. Erstellen Sie ein Board Symbol *shiftreg.bsf*
3. Bauen sie *shiftreg.bsf* in den Top Level ein. Verbinden Sie die Parallel Ausgänge des Schieberegisters temporär mit zwei 7-seg Decodern und verbinden Sie die Sieben Segment Decoder mit den 7-Segment LEDs auf dem Board. Dies ist nur zum Testen des Schieberegisters. Später bauen wir noch die Ausgangsregister ein.
4. Benutzen Sie eventuell Signal Tap um die Funktionen Ihres Schieberegisters zu testen.

## 11.5 Installieren des Bluetooth Terminals auf dem Smartphone

1. Installieren Sie auf Ihrem Mobiltelefon die App „bt terminal V4.3“ aus dem App Shop.
2. Stecken Sie den Bluetooth Empfänger auf das DE2 Board, schalten Sie Ihr Entwicklungsboard ein und laden Sie die *.sof* Datei auf das Board.
3. Drücken Sie am Board *Key(0)* um die Schaltung zurückzusetzen.
4. Gehen Sie mit Ihrem Mobiltelefon auf „Einstellungen“ und suchen Sie nach dem Bluetooth Modul. Die Bluetooth Module beginnen alle mit RNBT-xxxx. xxxx steht für die letzten 4 Zeichen der MAC Adresse des Empfängers.
5. Koppeln Sie Ihr Telefon mit dem Bluetooth Modul. Nach erfolgreicher Verbindung muss die LEDG 0 auf Ihrem DE2 Board leuchten.

6. Gehen Sie in der Bluetooth Terminal App auf „Settings“ und setzen Sie „Send hexadezimal“.
7. Starten Sie *Signal Tap* im Quartus und fügen Sie *serdata\_sync*, *edge*, *start\_bit*, *baud\_tick*, die Zähler Bits des Start Bit Detektors und die Zähler Bits des Baud Tick Generators als Beobachtungspunkte ein. Wählen Sie die fallende Flanke des Signals *serdata\_sync* als Trigger für Signal Tap.
8. Laden Sie den Entwurf auf das DE2-115 board. Senden Sie vom Bluetooth Terminal Hex-Zahlen an das Modul Verifizieren Sie die Funktionalität Ihres Entwurfs mit Signal Tap.

## 11.6 Ausgangs-Register

In den Ausgangs-Registern werden die vier, über das BT Modul übertragenen, Bytes gespeichert. Das Ausgangs-Register besteht aus den vier Byte großen Speichern und einer FSM, die vor jedem empfangenen Byte weiter schaltet und so bestimmt, welches Byte in welchem Register abgespeichert werden soll (siehe Abb. 11.5)

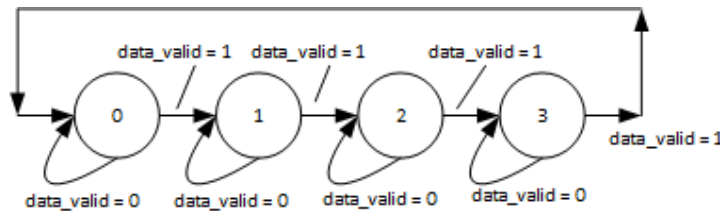
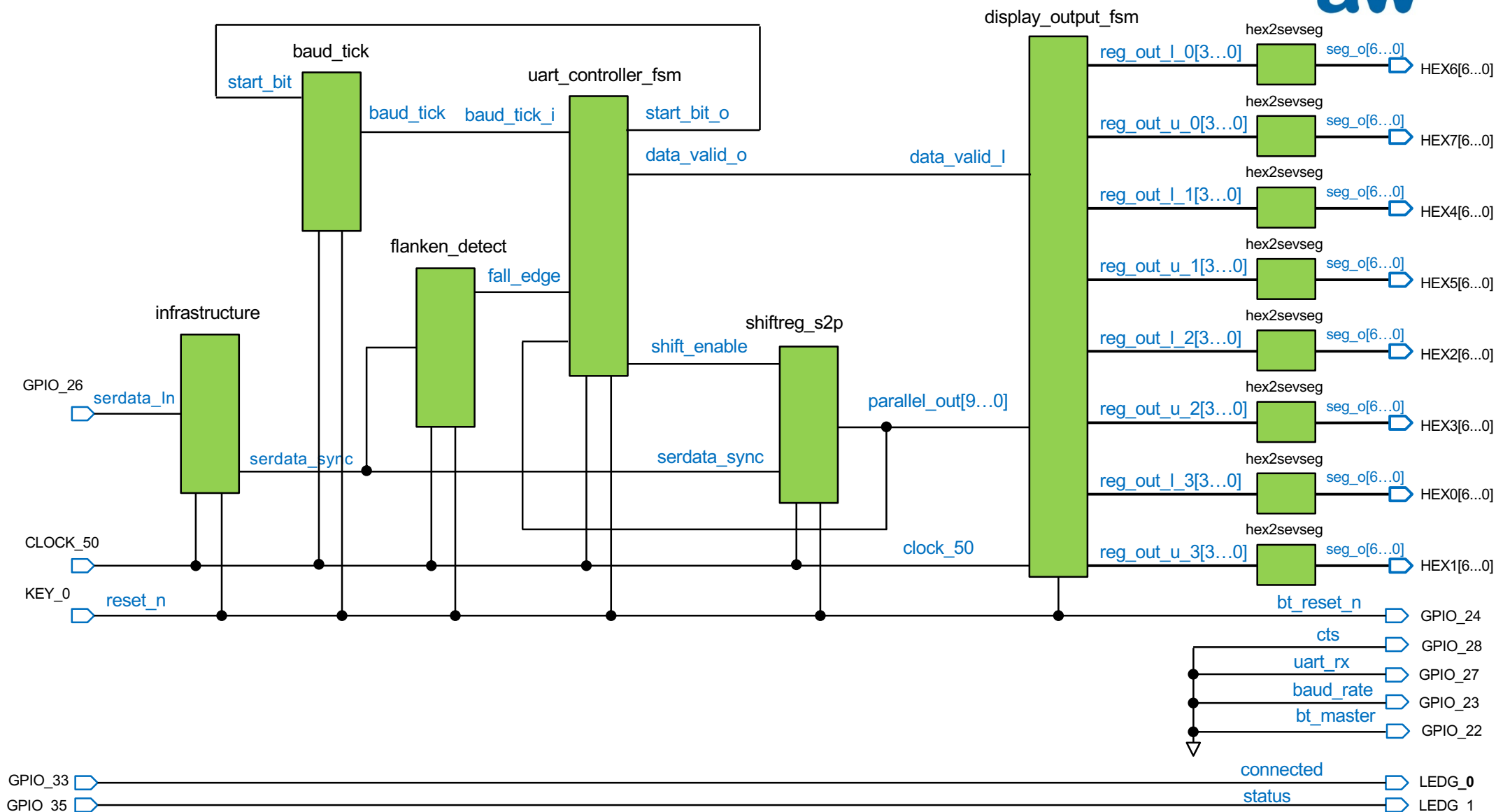
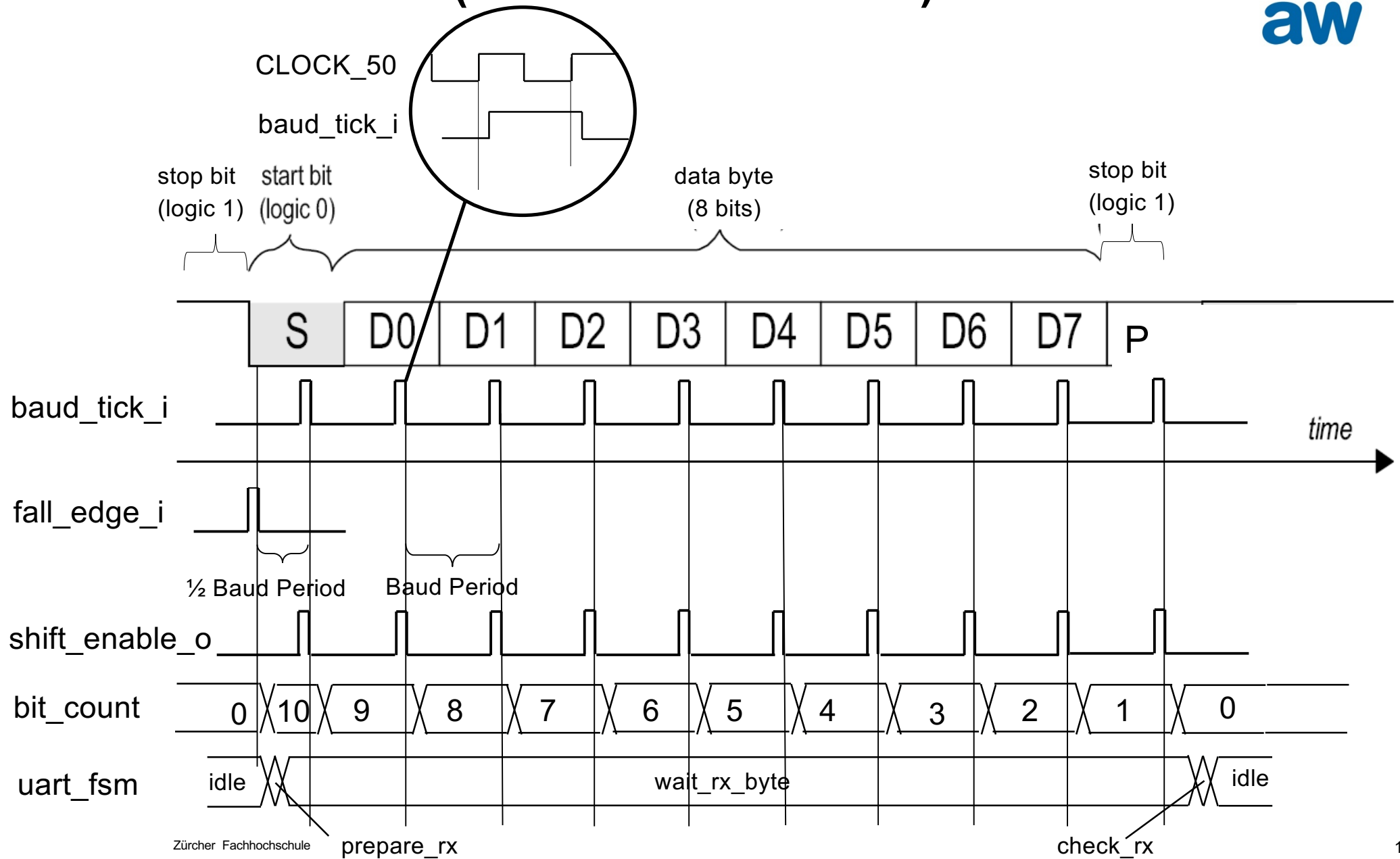


Abbildung 11.3: RX State Machine

1. Erstellen Sie Display Output FSM und die Register in **einer** VHDL Datei *rxreg.vhd*.
2. Setzen Sie *rxreg.vhd* als Top Level, analysieren Sie den *rxreg.vhd* mit Quartus und prüfen Sie nach unbeabsichtigten Latches. Erstellen Sie ein Board Symbol *rxreg.bsf*
3. Bauen sie *rxreg.bsf* in den Top Level ein. Verbinden Sie die 7-Segment LEDs auf dem Board.
4. Benutzen Sie Signal Tap um evtl. Fehler zu finden.

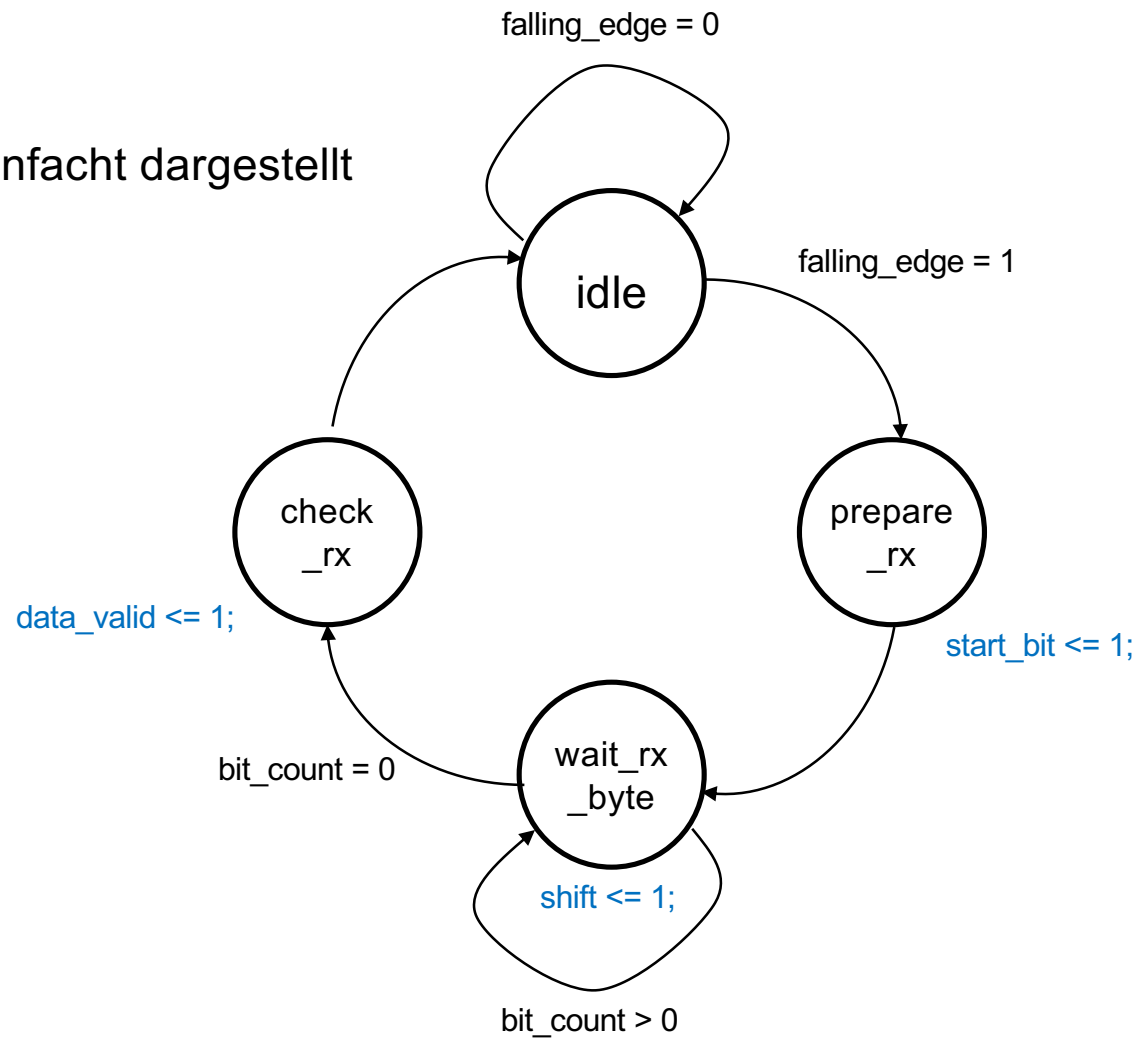


# Bit Counter (im UART Controller)

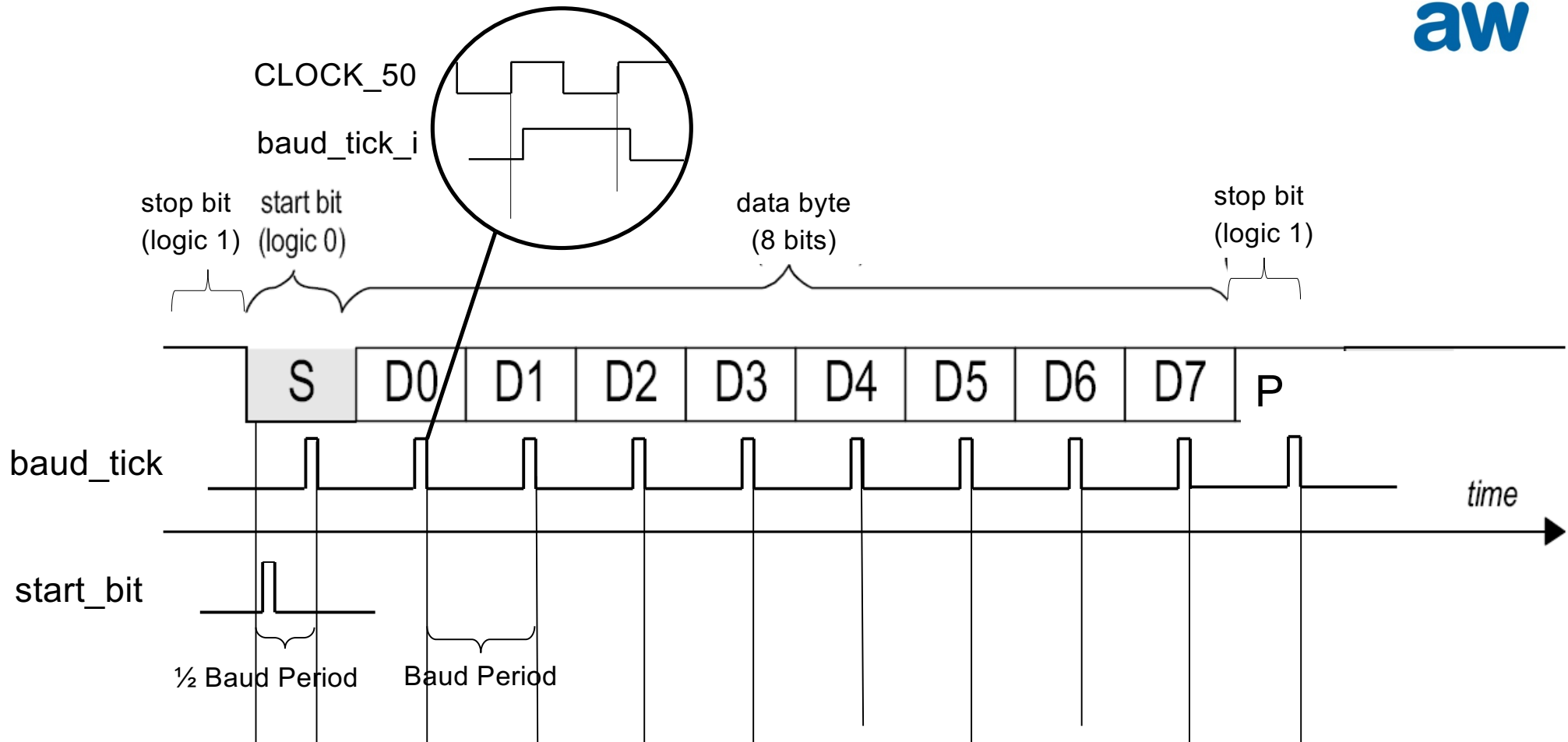


# UART FSM

Vereinfacht dargestellt

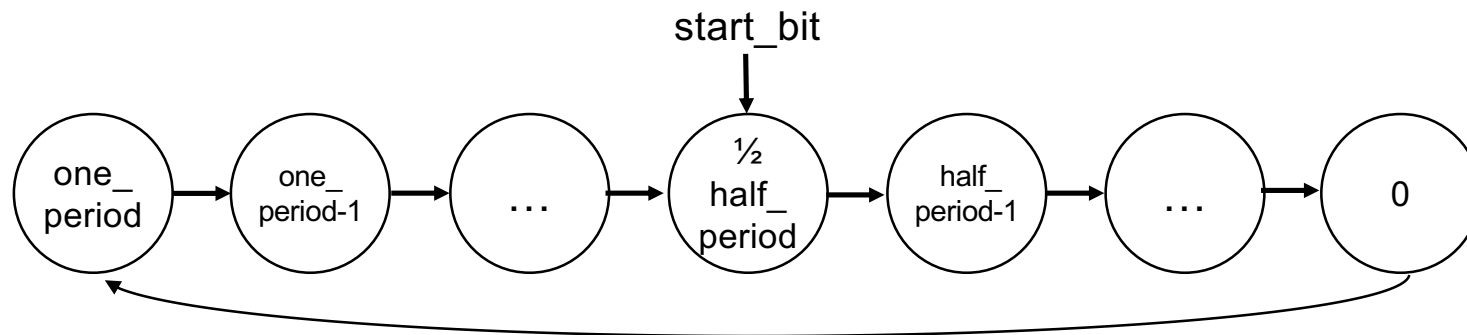
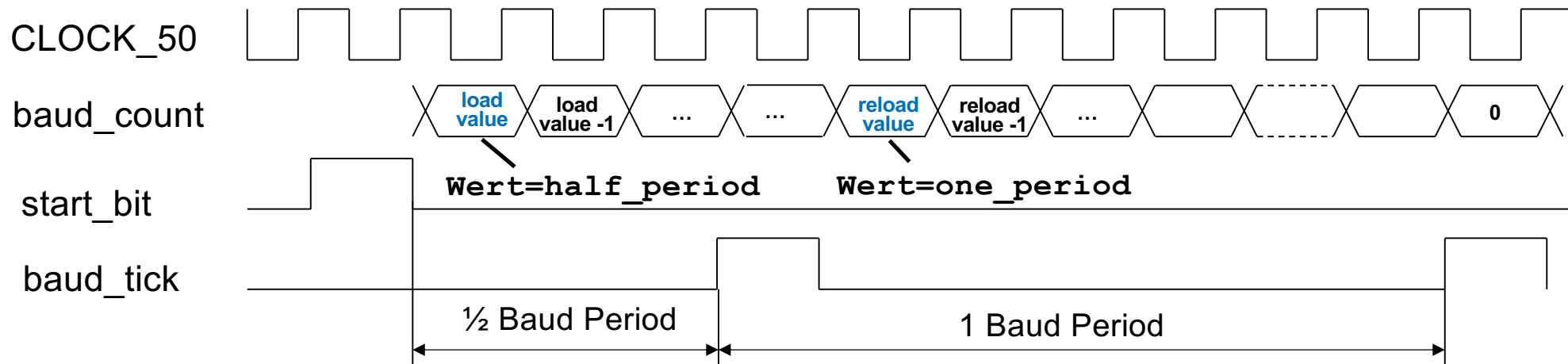


# Baud Tick Generator





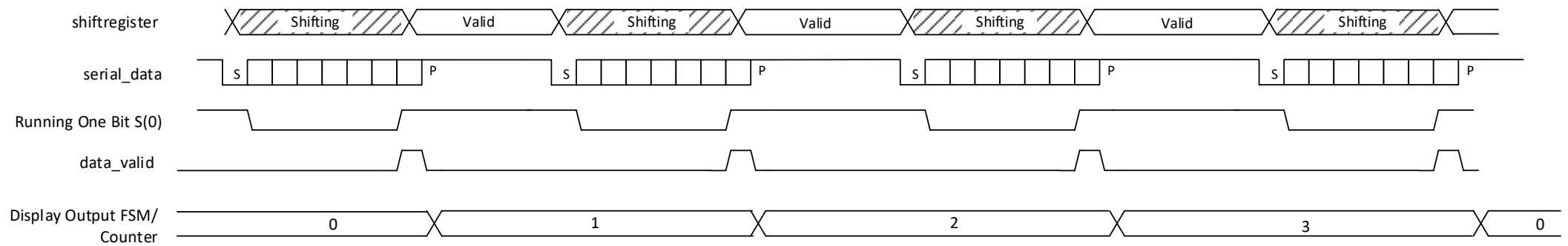
# Baud Tick Generator



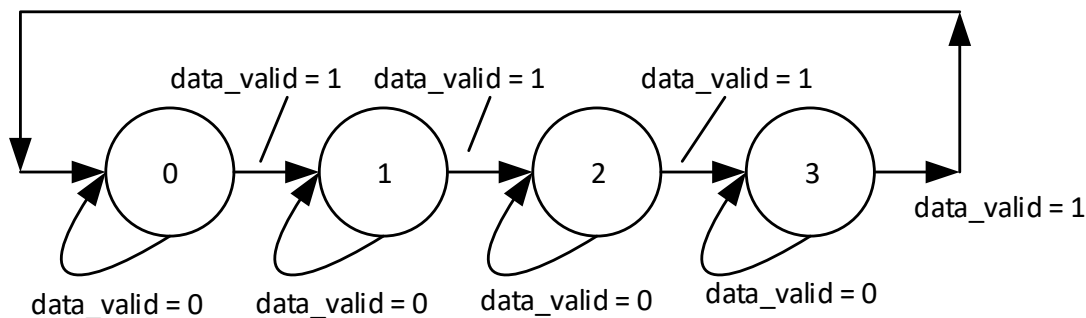
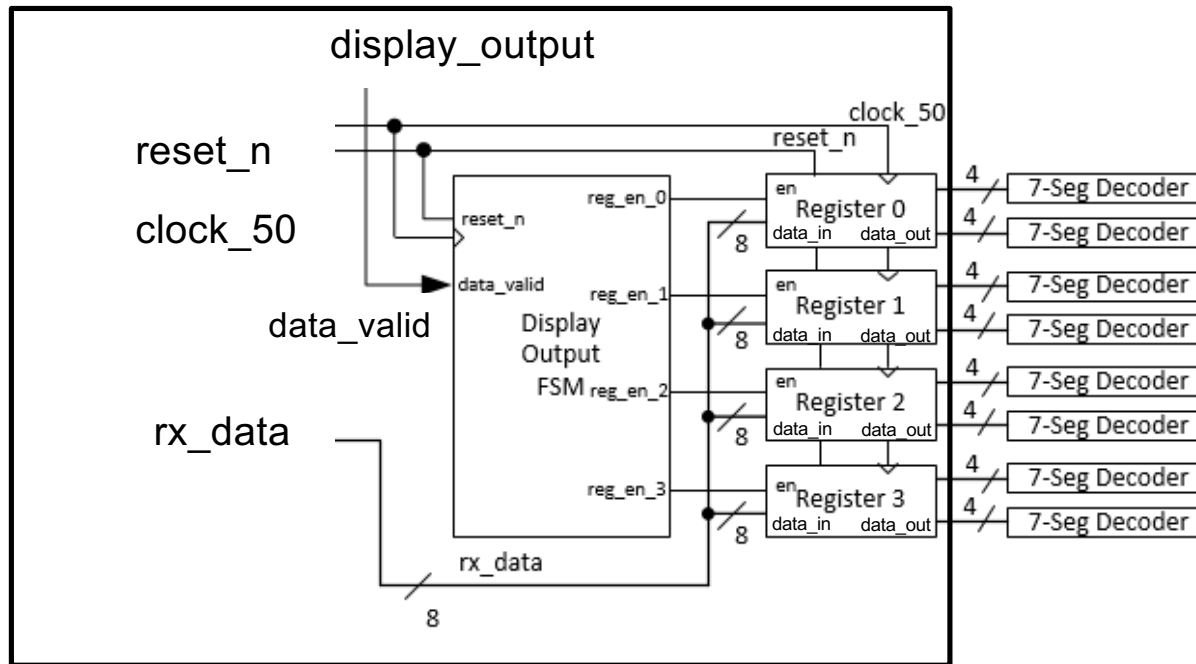
```

CONSTANT clock_freq      : natural := 50_000_000; -- Clock/Hz
CONSTANT baud_rate       : natural := 115_200;    -- Baude Rate/Hz
CONSTANT count_width     : natural := 10;        -- FreqClock/FreqBaudRate=50000000/115200 =434 so need 10bits
CONSTANT one_period      : unsigned(count_width - 1 downto 0) := to_unsigned(clock_freq / baud_rate , count_width);
CONSTANT half_period     : unsigned(count_width - 1 downto 0) := to_unsigned(clock_freq / baud_rate / 2, count_width);
  
```

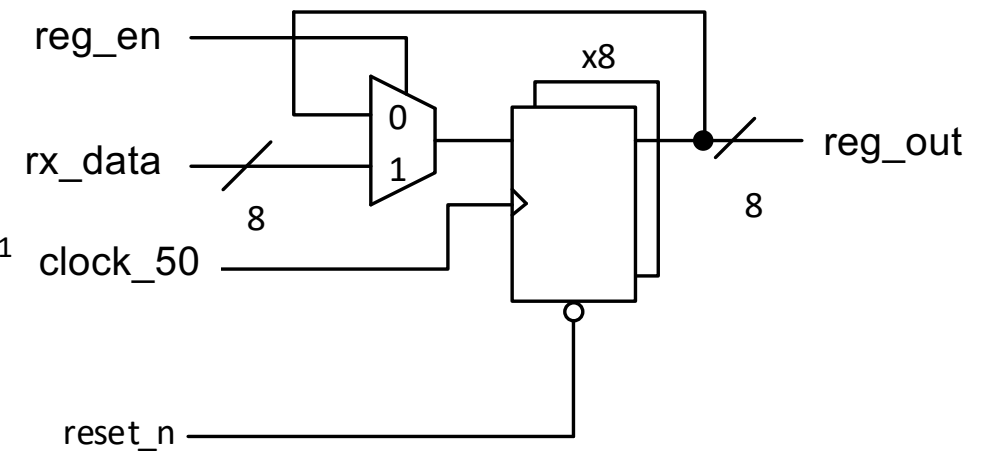
# Empfang von vier Bytes



# Display Output



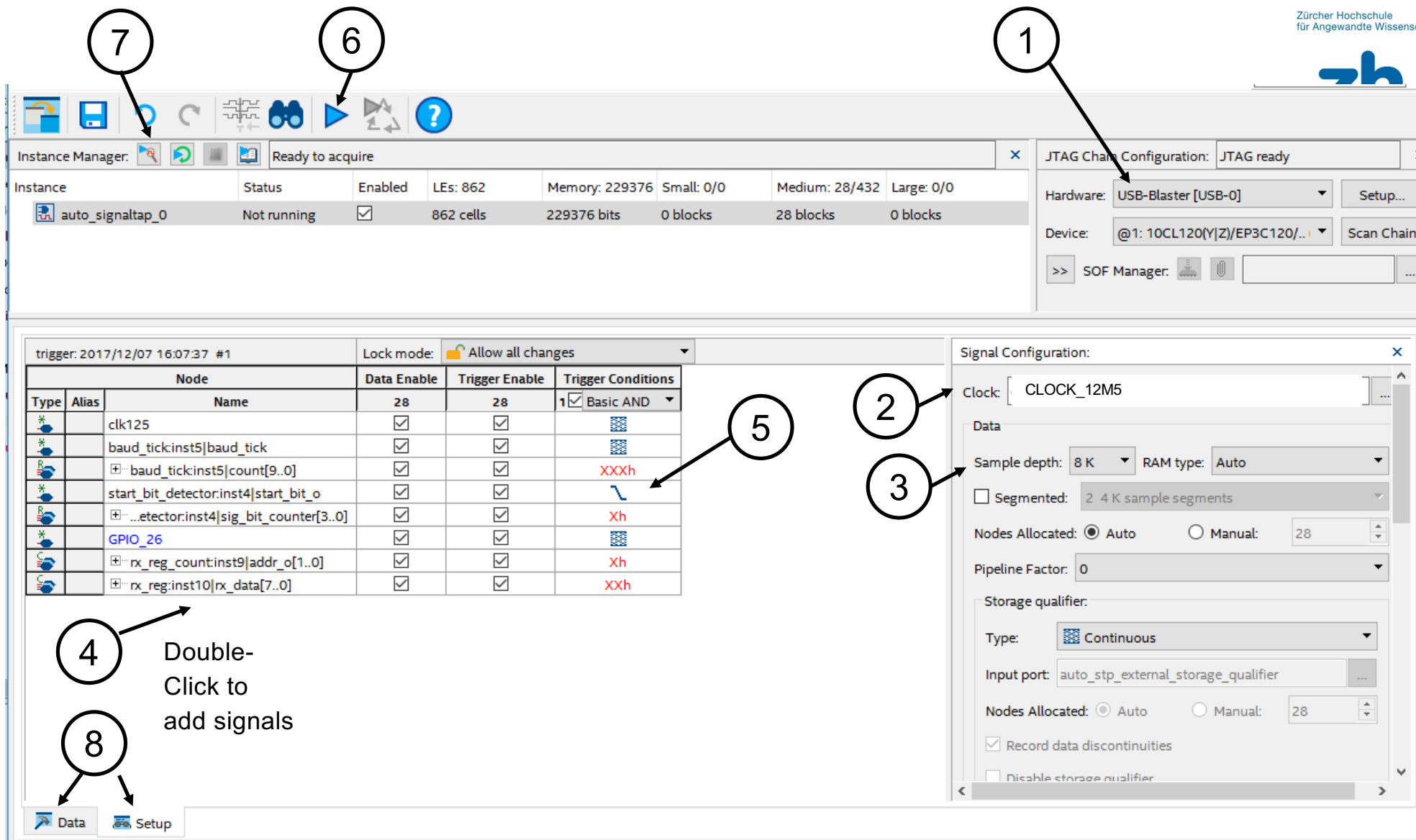
Display Output FSM



Registers 3..0

**Link zu Terminal Programm:** <https://play.google.com/store/apps/details?id=Qwerty.BluetoothTerminal>





The screenshot shows the logic analyser software interface. The top toolbar contains icons for file operations, analysis, and help. The 'Instance Manager' shows a single instance 'auto\_signaltap\_0' in a 'Not running' state. The 'JTAG Chain Configuration' panel shows the hardware as 'USB-Blaster [USB-0]' and the device as '@1: 10CL120[Y(Z)/EP3C120/...'. The main signal configuration table lists various signals with their aliases and trigger conditions. The 'Signal Configuration' panel on the right shows settings for the 'CLOCK\_12M5' clock, including sample depth (8 K), RAM type (Auto), and storage qualifier (Continuous). The bottom of the interface has tabs for 'Data' and 'Setup'.

Type	Alias	Name	Data Enable	Trigger Enable	Trigger Conditions
*		clk125	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
*		baud_tickinst5 baud_tick	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
B		baud_tickinst5 count[9..0]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	XXXh
*		start_bit_detector:inst4 start_bit_o	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
B		start_bit_detector:inst4 sig_bit_counter[3..0]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Xh
*		GPIO_26	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
C		rx_reg_countinst9 addr_o[1..0]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Xh
C		rx_reg:inst10 rx_data[7..0]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	XXh

Callout 1 points to the JTAG Chain Configuration panel. Callout 2 points to the Clock selection in the Signal Configuration panel. Callout 3 points to the Sample depth selection in the Signal Configuration panel. Callout 4 points to the 'Data' tab at the bottom. Callout 5 points to the Trigger Conditions column in the signal table. Callout 6 points to the 'Ready to acquire' button in the Instance Manager. Callout 7 points to the 'Arm' button in the toolbar. Callout 8 points to the 'Setup' tab at the bottom.

1. Define connection to Board

2. Set sample clock

3. Set sample depth

4. Add signals to watch

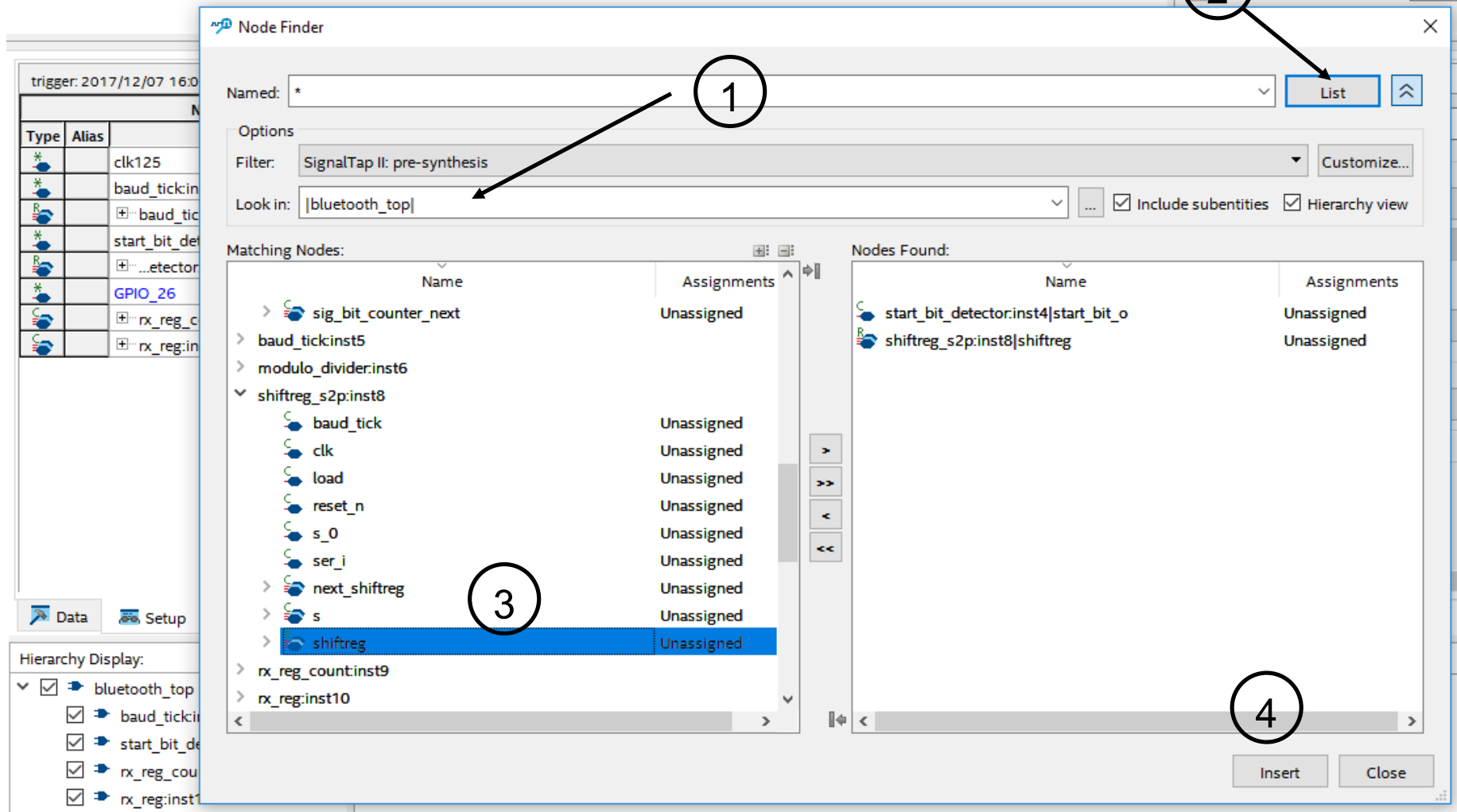
5. Set trigger

6. Synthesize &amp; download to board

7. Arm Analyser

8. Switch between logic analyser and setup view

# Node Finder im Signaltap



1. Set design hierarchy, often best «pre-synthesis»
2. List the filtered signals
3. Move signal to the right window
4. Insert to list of signals to watch