Project 1 Sample waveform generator

Sometimes there is a need to generate a waveform to order, perhaps to test a product on an assembly line. An oscillator could be used for this purpose, but it can be tedious to build an oscillator to do this if the waveform is not a pure sine wave, square wave, ramp, or triangular. One way of generating a complex waveform would be to use a microcontroller with a digital-to-analogue converter (DAC). The complex waveform could be stored into read only memory (ROM) and accessed via the microcontroller. However, this seems overkill. There are also potential sampling frequency limitations with the microcontroller. An alternative way would be to use a clocked FSM. The sampling rate could then be controlled by the clock rate, which would be limited by that of a PLD or FPGA. The complex waveform is still stored in a ROM but the ROM is controlled by the FSM.

Consider the block diagram of Figure 4.7. In this system, raising the st input starts the waveform generator. Each memory location is accessed in sequence and its content, a digitized sample of the waveform, is sent to the DAC to be converted to an analogue form. When the end of memory is reached, the address counter simply runs over to the zero location and starts again.

Setting the st input low stops the system. The actual sampling rate and, hence, the period of the waveform can be calculated once the state diagram is completed. The output of the DAC will need to be filtered to remove the sampling frequency component – this can be accomplished using a simple first-order low-pass filter section if the sampling frequency is much higher than the highest synthesized waveform frequency. (Usually, it is to satisfy Shannon's sampling theory.)

The state diagram now needs to be developed. A little thought reveals that the block diagram itself provides an indication of the sequence required.

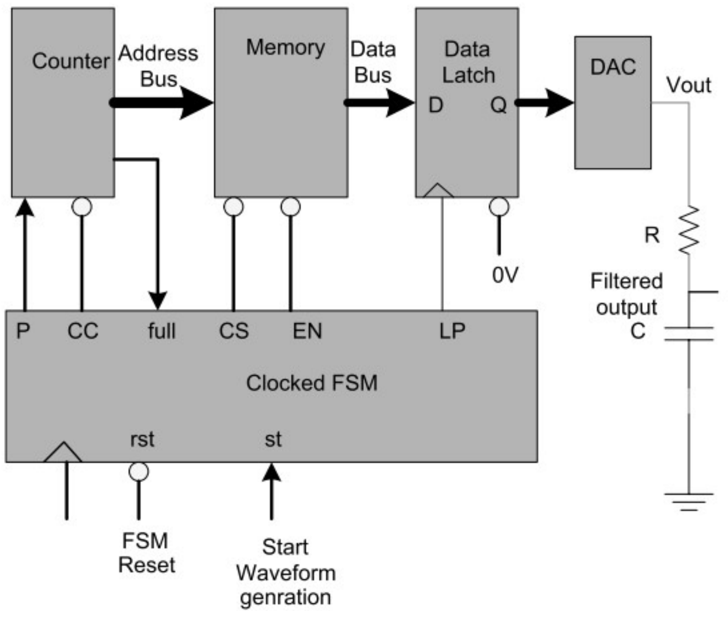

Figure 4.7. Block diagram of simple waveform generator. Please note that additional inputs are missing including the type of the waveform, period of the waveform and the number of data points in one period.

1. Initially, the address counter needs to be cleared to provide the necessary zero address for the first location of the memory. The system should remain in state s0 until the start input st is asserted (high).
2. The memory then needs to be enabled, selected, and allowed to settle, after which the data in the memory location will be available at the data latch inputs. Then the data need to be latched into the data latch to be available at the input of the DAC.
3. At this stage, the address counter needs to be incremented so as to point to the next memory location and the sequence in 2 repeated again as long as the start input is still asserted (high).

You will need to design a system where the user will select the type of the waveform (sine wave, square wave or triangle, or one special waveform of your choice), the sampling speed and number of data points. For example, sine wave with the period of 1 ms with at least 100 points per period. The supported periods should cover three orders of magnitudes at least (for example between 1ms and 1s). Please do research on ways to implement waveform generators.

The final report should include the description of the design, FSM as well as VHDL design and testbench.

Project 2 DEVELOPMENT OF A SERIAL ASYNCHRONOUS RECEIVER

Often, there is a requirement to use serial transmission and receiving of data in a digital system. Although there are lots of serial devices on the market, it is useful to be able to implement one's own design directly to incorporate into an FPGA device. The advantage of this approach is that the baud rate and protocols can be dictated by the designer, as can how the device will be controlled.

In this example, the serial data input is encapsulated into an asynchronous data packet with start (st) and stop (sp) protocol bits that have been added to the serial transmission packet. These

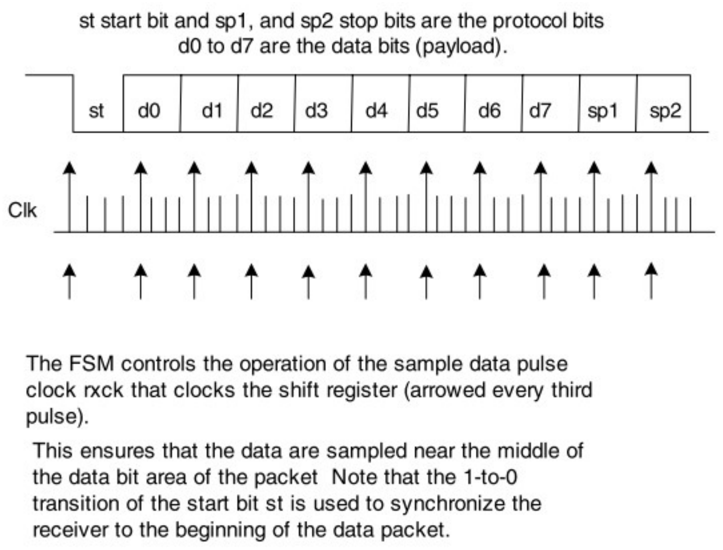
st start bit and sp1, and sp2 stop bits are the protocol bits
d0 to d7 are the data bits (payload).
st
d0
d1
d2
d3
d4
d5
d6
d7
sp1
sp2
Clk
The FSM controls the operation of the sample data pulse clock rxck that clocks the shift register (arrowed every third pulse).
This ensures that the data are sampled near the middle of the data bit area of the packet Note that the 1-to-0 transition of the start bit st is used to synchronize the receiver to the beginning of the data packet.

Figure 4.20 Protocol of the serial asynchronous receiver.

are used to provide a means of identifying the data packets as they arrive. This allows the data packets to arrive at any time and at any selected rate (dictated by the baud rate).

The problem with receiving data is that it is necessary to ensure that the shift register is clocked with correct data bits. To do this the FSM clock is used to drive an FSM to create a shift register clock RXCK in the middle of the data bit time period. This RXCK clock pulse can be seen in Figure 4.20 as the arrowed pulses occurring every third clk pulse. Thus, the clk signal runs four times faster than the RXCK signal generated by the FSM. Note, the FSM needs to detect the start of the data packet by looking for the 1-to-0 transition on the receiver input.

The block diagram for the serial asynchronous receiver is illustrated in Figure 4.21. The FSM is used to create the shift register clock, and to control the operation of the serial asynchronous receiver. The Divide by 11 Counter is used to count out the 11 bits that make up the protocol packet. This provides a shift register full signal rxf to indicate to the FSM that a complete data packet has arrived. The Data Latch is used for collecting the received data from the shift register to send to the outside world device controlling the asynchronous receiver.

The FSM must wait for start (by monitoring for the st bit change 1 to 0); this is just the first receive bit coming into the shift register. When detected, shift the data into the shift register. If the stop bit is not correct, then the FSM can issue an error via signal ERR. Note, in this version the start bit is tested along with the two stop bits via an AND gate (error detection signal ed) to ensure packet alignment after the complete packet is received, the receiver rx input is held at logic 1 by a pull-up resistor so that the start bit (active low) can be detected. The ack signal is available so that the outside world device using the system can respond to an error condition (no error means successful packet received). Healthy data packets will be latched into the data latch ready to be read by the controlling device.

The signal CDC is used to clear the shift register and set st to logic 1, i.e. the flip-flop representing the start bit of the shift register needs to be pre-set so that it can be cleared by the incoming start bit from the serial line.

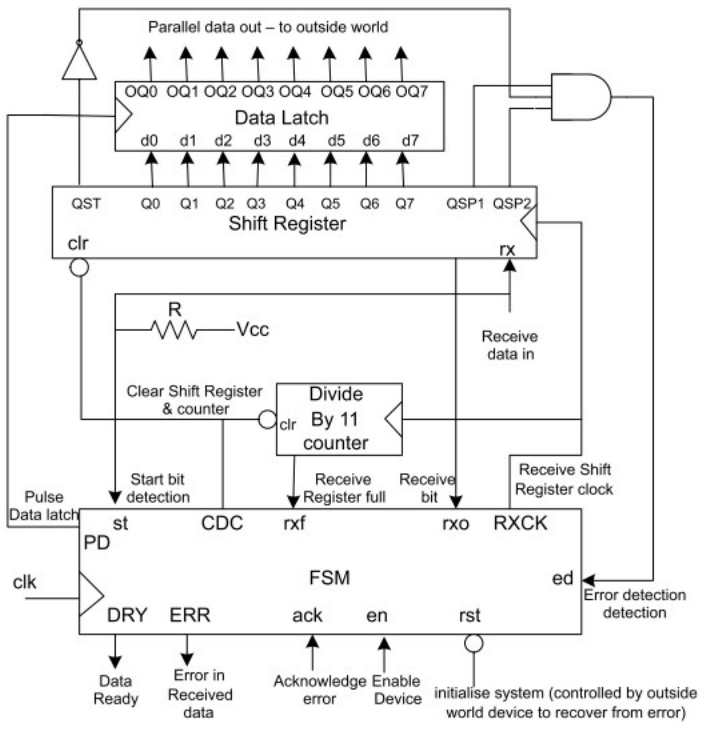
Parallel data out – to outside world
OQ0 OQ1 OQ2 OQ3 OQ4 OQ5 OQ6 OQ7
Data Latch
d0 d1 d2 d3 d4 d5 d6 d7
QST Q0 Q1 Q2 Q3 Q4 Q5 Q6 Q7 QSP1 QSP2
Shift Register
clr
rx
R
Vcc
Receive data in
Clear Shift Register & counter
Divide By 11 counter
clr
Start bit detection
Receive Register full
Receive bit
Receive Shift Register clock
Pulse Data latch
st
CDC
rxf
rxo
RXCK
PD
clk
FSM
ed
Error detection detection
DRY
ERR
ack
en
rst
Data Ready
Error in Received data
Acknowledge error
Enable Device
initialise system (controlled by outside world device to recover from error)

Figure 4.21 Block diagram of the serial asynchronous receiver.

The en signal is used to enable and start the asynchronous receiver. This is necessary to ensure that the system starts monitoring the clock so as to issue the shift register clock pulse (RXCK) at the right time (in the middle of the data bit period).

In this project, you will send data between two FPGA boards (or from the same FPGA board you will send data to the output pin and then connect this output pin to the input pin to receive the data). You will need to implement serial transmitter as well to support this protocol, then to receive the data on the receiver and to present what you received.

The final report should include the description of the design, FSM as well as VHDL design and testbench.