GH09.B.4.compress - Data Compression

General Description

This design is for data compression based on the Lempel-Ziv-Welch (LZW) lossless data compression algorithm. The design processes a block of bits and outputs a compressed version of that block.

Features

- Original Lempel-Ziv-Welch compression algorithm
- Processes arbitrary sized block of data
- 4kilo of symbols in the code book
- 12 bit static symbol output

Block Diagram

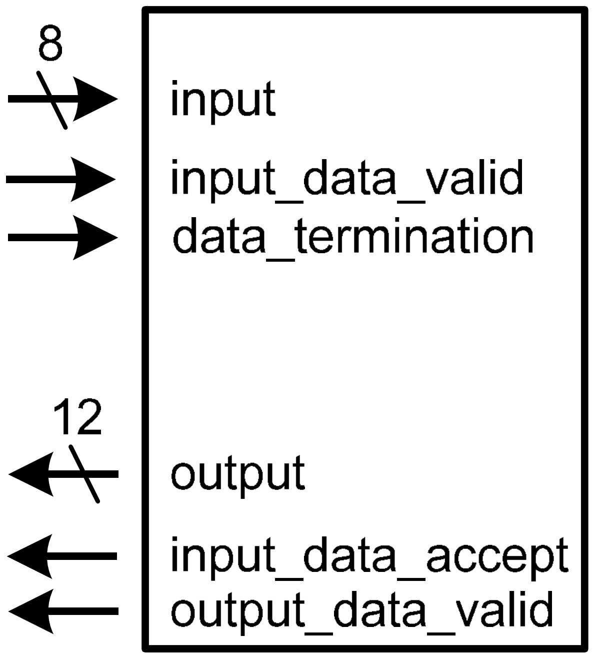

Figure 1 – Block diagram of the LZW compression device

Details

The Data Compression design will implement the LZW compression algorithm. We will define this based on a pseudo code of the algorithm.

Algorithm

The LZW compression algorithm is as follows:

```
INPUT: character i
OUTPUT: symbol o
VARIABLE: character string t
t = NULL;
while (characters to compress)
{
        i = next character
        if {t+i} exists in the code dictionary
                t = t+i
        else
        {
                add {t+i} to code dictionary
                o = t
                t = i
        }
}
```

In addition to the pseudo code, the operation of the design is defined in terms of how the data is received. This is defined by the input and output protocol description as seen below in the Chip Description section. The next section describes pins not described in the Chip Description section.

Pins

The input and output protocol defines many of the pins below. The remaining pins are further defined as follows:

- input – this is a simple 8 bit port to pass in the data to be compressed in 8 bit chunks.
- data_termination – this signal indicates when the bits to be compressed are all sent. The next bits of data will be compressed using a new codebook.
- Output – this is the output 12 bit port that sends out the compressed bits in 12bit chunks.

Chip Description

Input Protocol

Inputs are received using a simple handshaking protocol. The pins of importance are the input (parallel port), the input_data_ready, the input_data_valid, the reset, and the clock. Figure 2 shows the basic communication setup between receiver (this design) and sender (external environment), and Figure 3 shows a waveform for a simple transfer between sender and receiver assuming the sender has one 8 bit packet to send (also note that the input data width in the examples is 8 bits as opposed to 128 bits in the actual design).

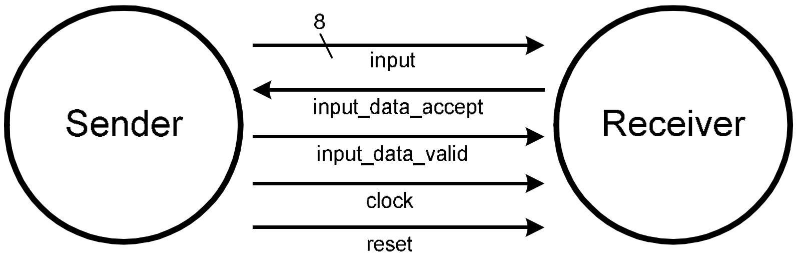

Figure 2 – The interface signals between the sender (the external environment) and the receiver (this design).

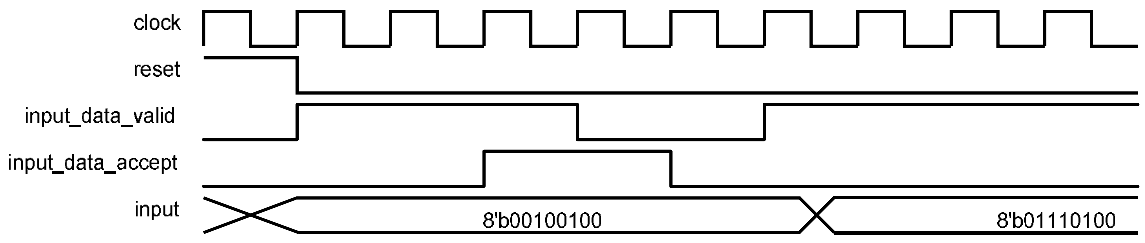

Figure 3 – a waveform for the communication between sender and receiver for one packet.

The assumption in this simple send receive protocol is that the shared clock between sender and receiver are synchronized.

Figure 4 and 5 show the finite state diagrams of both the sender and receiver.

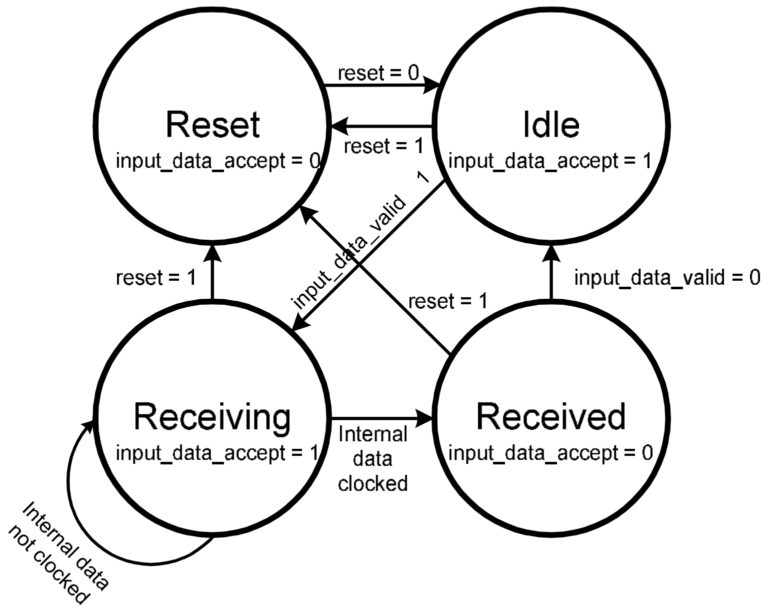

Figure 4 – Finite state machine for the receiver.

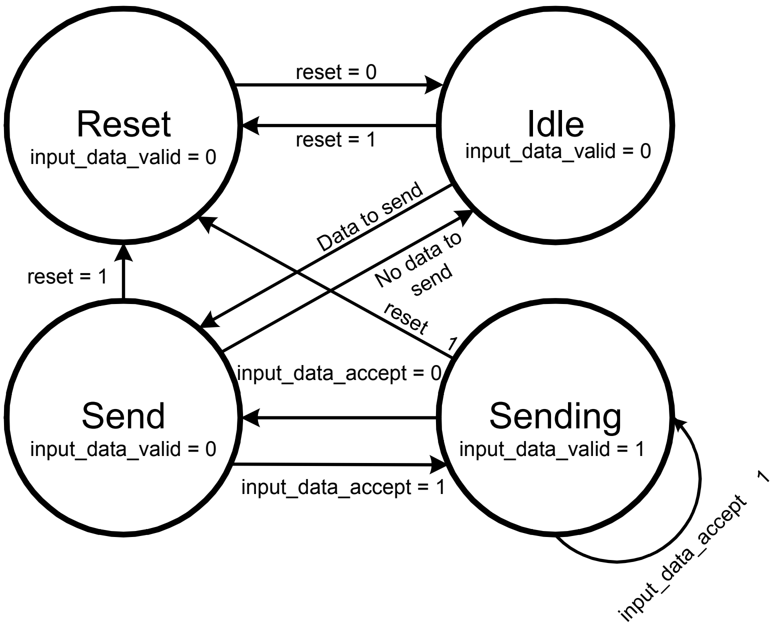

Figure 5 – Finite state machine for the sender.

Output Protocol

The output protocol is a very simple protocol. When the ouput_data_valid signal is high for one clock cycle, then the output bus contains a value. Figure 6 shows a waveform where three output values (val0, val1, and val2) are sent out of from the design.

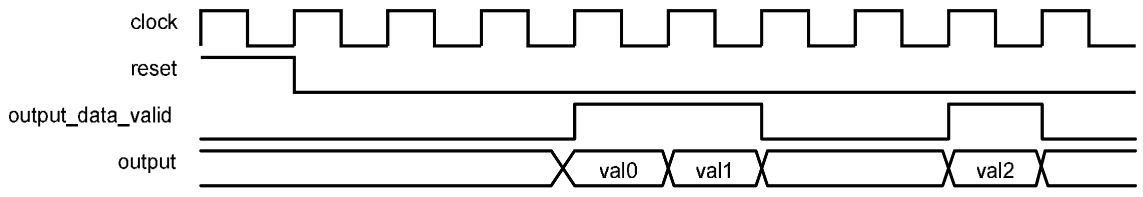

Figure 6 – A waveform showing the output being sent and the corresponding output_data_valid signal.