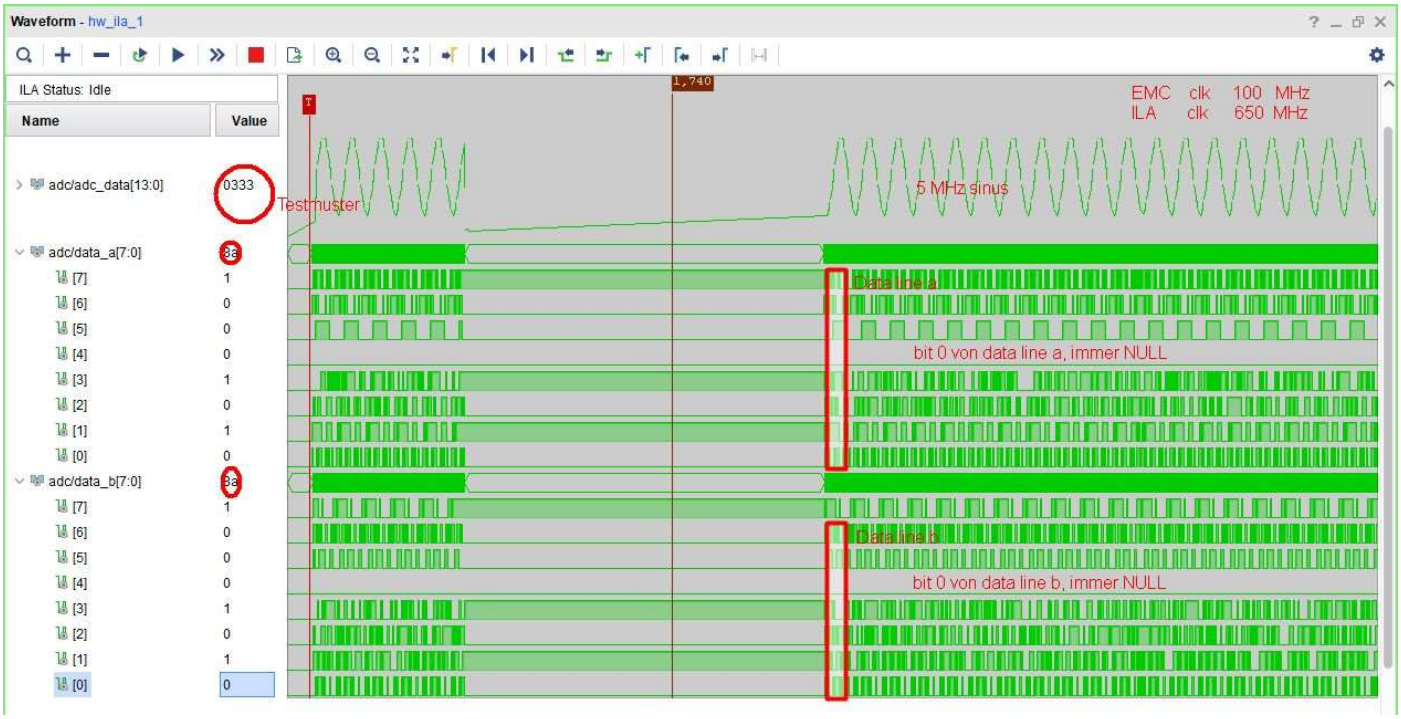
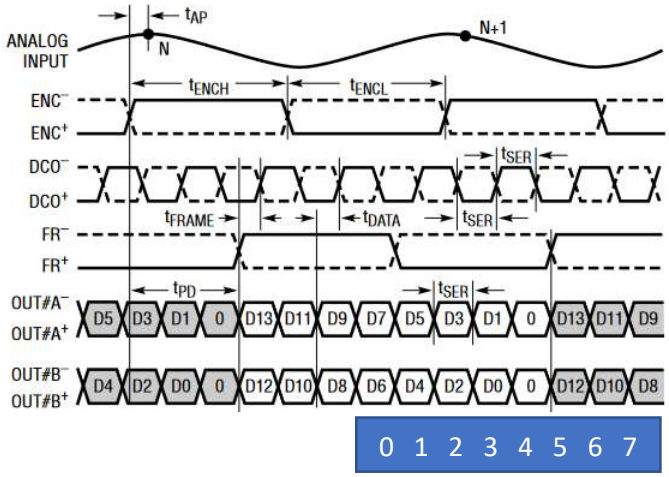


Delay DCO clk (400 MHz): 0 counts, Data (400 MHz): 256 counts, Frame clk (100 MHz): 511 counts

2-Lane Output Mode, 16-Bit Serialization*



0 1 2 3 4 5 6 7

```
reg [8-1: 0]    current_a;
reg [8-1: 0]    past_a;
always  @(posedge  clk_adc_a_pll) begin

    current_a    <= data_a;          // frische Daten aus serdes
    past_a    <= current_a;

end

reg [8-1: 0]    current_b;
reg [8-1: 0]    past_b;
always  @(posedge  clk_adc_a_pll) begin

    current_b    <= data_b;          // frische Daten aus serdes
    past_b    <= current_b;

end
```

So kommen die Daten aus dem Serdes, ohne bit slip!

Es wird immer ein Wert gespeichert und dann aus dem alten und aktuellen Wert die 14 bit ADC Werte wie folgt zusammen gebaut …..

```
reg [14-1: 0]  adc_data;
always  @(posedge  clk_adc_a_pll) begin

// eins nach links
//    adc_data <= {current_a[6], current_b[6], current_a[7], current_b[7],
//                 past_a[0],    past_b[0],    past_a[1],    past_b[1],  past_a[2], past_b[2], past_a[3], past_b[3], past_a[4], past_b[4]};

// richtige position: (so geht das richtig...)
    adc_data <= {current_a[5], current_b[5], current_a[6], current_b[6], current_a[7], current_b[7],
                 past_a[0],    past_b[0],    past_a[1],    past_b[1],    past_a[2],    past_b[2],    past_a[3], past_b[3]};

// eins nach rechts
//    adc_data <= {current_a[4], current_b[4], current_a[5], current_b[5], current_a[6], current_b[6], current_a[7], current_b[7],
//                 past_a[0],    past_b[0],    past_a[1],    past_b[1],    past_a[2],    past_b[2]};

    adc_data_1  <= adc_data + 14'h2000;
end
```