NTNU
Kunnskap for en bedre verden

Institutt for elektroniske systemer

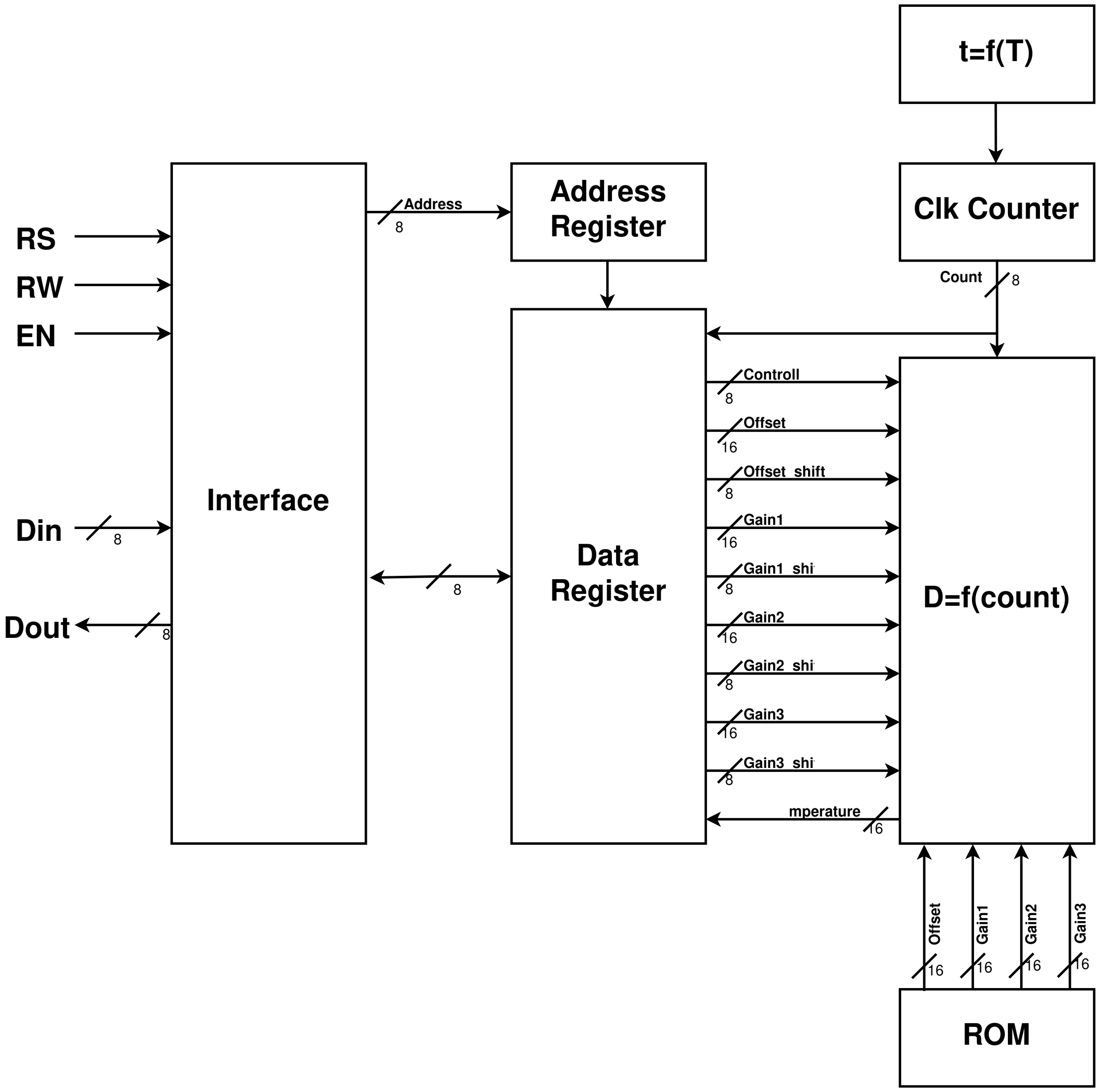
TFE4188 Avanserte Integrerte Kretser

Temperature interface

Skrevet av:
Trond F. Christiansen

12. mai 2024

1 Block diagram



Figur 1: Interface

Pin	**Function**	**Note**
Vdd	Power supply	
GND	Power supply	
$\overline{\text{RESET}}$	Sensor reset	
CLK	Clock signal	40MHz external clock signal
SIGNAL	Counter trigger from analog design	
EN	Interface enable	
RS	Selects registers. 0: Address register 1: Data register	
RW	Selects read or write. 0: Write 1: Read	
Dout	Data output	Data split into Dout and Din due to limitations of OpenLane
Din	Data input	Data split into Dout and Din due to limitations of OpenLane

Tabell 1: Interface ports

Address	**Name**	**R/W**	**Function**
0x00	-	-	-
0x01	TMPH	R	Temperature high
0x02	TMPL	R	Temperature low
0x03	RAWH	R	Raw count high
0x04	RAWL	R	Raw count low
0x05	CTL	R/W	Controll
0x06	-	-	-
0x07	-	-	-
0x08	OFS1	R/W	Ofset
0x09	OFS0	R/W	Ofset
0x0A	OFS_SHA	R/W	Offset shift amount
0x0B	1ORD1	R/W	1st order gain high
0x0C	1ORD0	R/W	1st order gain low
0x0D	1ORD_SHA	R/W	1order shift amount
0x0E	2ORD1	R/W	2nd order gain high
0x0F	2ORD0	R/W	2nd order gain low
0x10	2ORD_SHA	R/W	2order shift amount
0x11	3ORD1	R/W	3rd order gain high
0x12	3ORD0	R/W	3rd order gain low
0x13	3ORD_SHA	R/W	3order shift amount

Tabell 2: Register overview

CTL	Controll register of the temperature unit							
Bit	7	6	5	4	3	2	1	0
Name	CALEN	CALSELH	CALSELL	-	-	-	-	-

Tabell 3: Control register bits

- **7 CALEN :** Custom calibration enable bit.
  - **0:** Use pre-programed 3rd polynomial order calibration.
  - **1:** Use custom calibration based on CALSEL bits.

- **6:5 CALSEL :** Calibration selection bits. Chooses calibration metod when custom calibration is enabled.
  - **00:** Use 3rd order polynomial calibration with gain values from pre-programmed calibration and offset value from *OFS* registers
  - **01:** Use 1st order linear calibration with gain value from *1ORD* registers and offset from *OFS* registers
  - **10:** Use 2nd order polunomial calibration with gain values form *1ORD* and *2ORD* registers and offset from *OFS* registers
  - **11:** Use 2nd order polunomial calibration with gain values form *1ORD*, *2ORD* and *3ORD* registers and offset from *OFS* registers

2 Code

```
// Macros
`define STA         0       // Status register          0x00    RO WIP
`define TMPH        1       // Temperature high byte    0x01    RO
`define TMPL        2       // Temperature low byte     0x02    RO
`define RAWH        3       // Raw data high byte       0x03    RO
`define RAWL        4       // Raw data low byte        0x04    RO
`define CTL         5       // Control register         0x05    RW
`define RES1        6       // Reserved                 0x06    RW
`define RES2        7       // Reserved                 0x07    RW
`define OFS_1       8       // Offset register 1        0x08    RW
`define OFS_0       9       // Offset register 0        0x09    RW
`define OFS_SHA     10      // Offset Shift amount reg  0x0A    RW
`define GAIN1_1     11      // Gain1 register 1         0x0B    RW
`define GAIN1_0     12      // Gain1 register 0         0x0C    RW
`define GAIN1_SHA   13      // Gain1 shift amount reg   0x0D    RW
`define GAIN2_1     14      // Gain2 register 1         0x0E    RW
`define GAIN2_0     15      // Gain2 register 0         0x0F    RW
`define GAIN2_SHA   16      // Gain2 shift amount reg   0x10    RW
`define GAIN3_1     17      // Gain3 register 1         0x11    RW
`define GAIN3_0     18      // Gain3 register 0         0x12    RW
`define GAIN3_SHA   19      // Gain3 shift amount reg   0x13    RW



module PulseDurationMeasurement(
    input wire clk,             // Clock input
    input wire reset_n,         // Reset input
    input wire RS,              // Register Select input
    input wire RW,              // Read/Write input
    input wire EN,              // Enable input
    input wire [7:0] data_in,   // Data bus input
    input wire signal_in,       // Signal input
    output reg [7:0] data_out   // Data bus output
);

// Internal variables
reg [5:0] address = 1'b000000;            // Address bus 6-bit
reg [7:0] data_reg [30:0];           // Data register 8-bit

reg state = 0;
reg [11:0] count, pulse_duration;

reg signed [31:0] c_offset, c_gain1, c_gain2, c_gain3;
reg signed [15:0] temperature_output = 0; // Signed temperature value output
reg [64:0] tmp;

reg [32:0] u_offset, u_gain1, u_gain2, u_gain3;
reg [7:0] u_offset_sha, u_gain1_sha, u_gain2_sha, u_gain3_sha;

// Constants declaration
parameter IDLE = 1'b0;
parameter COUNT = 1'b1;

// Constants for temperature conversion
parameter GAIN3 = -508; // (-0.0000004731 * 1073741824) scaled by 2^30
parameter GAIN2 = 2101; // (0.0020044419 * 1048576) scaled by 2^20
parameter GAIN = -2532; // (-2.4734734627 * 1024) scaled by 2^10
parameter OFFSET = 847;

parameter CALEN_bm = (1 << 7);  // Calibration enable bit mask
parameter CALSEL_bm = ((1 << 6) | (1 << 5));  // Calibration 3 bit mask
parameter CAL_3 = ((1 << 6) | (1 << 5));  // Calibration 3
parameter CAL_2 = ((1 << 6) | (0 << 5));  // Calibration 2
parameter CAL_1 = ((0 << 6) | (1 << 5));  // Calibration 1
parameter CAL_0 = ((0 << 6) | (0 << 5));  // Calibration 0


always_ff @(posedge clk) begin
    if(!reset_n) begin
        data_out = 8'b00000000;
        address = 6'b000000;
    end
    else if(EN == 1) begin
        if(RS == 1 && RW == 0) begin        // Read data
            data_out = data_reg[address];
            if(data_reg[address] != 0) begin
            end
        end
        else if(RS == 1 && RW == 1) begin   // Write data
            if(address >= 5 && address <= 30) begin
                data_reg[address] = data_in;
            end
        end
        else if(RS == 0 && RW == 0) begin   // Read address
            data_out = address;
            if(address != 0) begin
            end
        end
        else if(RS == 0 && RW == 1) begin   // Write address
            address = data_in[5:0];
        end
        else begin                          // Default
            data_out = 8'b00000000;
        end
    end
end

always_ff @(posedge clk) begin
    if(!reset_n) begin
        state <= IDLE;
        count <= 12'h0;
    end else begin
        case(state)
            IDLE: begin
                count <= 12'b000000000001;
                if (signal_in == 0) begin
                    state <= COUNT;
                end
            end

            COUNT: begin
                if (signal_in == 0) begin
                    count <= count + 1;
                end else begin
                    pulse_duration <= count;

                    data_reg[`RAWH] <= 8'b0 | pulse_duration[11:8];
                    data_reg[`RAWL] <= pulse_duration[7:0];
                    state <= IDLE;
                end
            end
        endcase
    end
end

//always_comb begin
always_ff @(posedge clk) begin
    // // Concatenate the offset registers to form a 16-bit signed integer


    c_offset =  data_reg[`OFS_1] << 8 | data_reg[`OFS_0];
    c_gain1 =  data_reg[`GAIN1_1] << 8 | data_reg[`GAIN1_0];
    c_gain2 =  data_reg[`GAIN2_1] << 8 | data_reg[`GAIN2_0];
    c_gain3 =  data_reg[`GAIN3_1] << 8 | data_reg[`GAIN3_0];


    if(!reset_n) begin
        temperature_output = 16'b0;
    end
    else if (data_reg[`CTL] & CALEN_bm) begin // Calibration enabled
        if ((data_reg[`CTL] & CALSEL_bm) == CAL_0) begin

            u_gain1 = GAIN3;
            u_gain2 = GAIN2;
            u_gain3 = GAIN;
            u_offset = c_offset;

            u_gain1_sha = 30;
            u_gain2_sha = 20;
            u_gain3_sha = 10;
            u_offset_sha = data_reg[`OFS_SHA];

        end
        else if ((data_reg[`CTL] & CALSEL_bm) == CAL_1) begin

            u_gain1 = c_gain1;
            u_gain2 = 0;
            u_gain3 = 0;
            u_offset = c_offset;

            u_gain1_sha = data_reg[`GAIN1_SHA];
            u_gain2_sha = 0;
            u_gain3_sha = 0;
            u_offset_sha = data_reg[`OFS_SHA];

        end
        else if ((data_reg[`CTL] & CALSEL_bm) == CAL_2) begin

            u_gain1 = c_gain1;
            u_gain2 = c_gain2;
            u_gain3 = 0;
            u_offset = c_offset;

            u_gain1_sha = data_reg[`GAIN1_SHA];
            u_gain2_sha = data_reg[`GAIN2_SHA];
            u_gain3_sha = 0;
            u_offset_sha = data_reg[`OFS_SHA];

        end
        else if ((data_reg[`CTL] & CALSEL_bm) == CAL_3) begin

            u_gain1 = c_gain1;
            u_gain2 = c_gain2;
            u_gain3 = c_gain3;
            u_offset = c_offset;

            u_gain1_sha = data_reg[`GAIN1_SHA];
            u_gain2_sha = data_reg[`GAIN2_SHA];
            u_gain3_sha = data_reg[`GAIN3_SHA];
            u_offset_sha = data_reg[`OFS_SHA];

        end

    end
    else begin  // Custom calibration disabled
```

```
        u_gain1 = GAIN3;
        u_gain2 = GAIN2;
        u_gain3 = GAIN;
        u_offset = OFFSET;

        u_gain1_sha = 30;
        u_gain2_sha = 20;
        u_gain3_sha = 10;
        u_offset_sha = 0;

    end

    tmp = ((u_gain3*pulse_duration*pulse_duration*pulse_duration) >>
    ↪   u_gain3_sha)
            + ((u_gain2*pulse_duration*pulse_duration) >> u_gain2_sha)
            + ((u_gain1*pulse_duration) >> u_gain1_sha)
            + u_offset;
    temperature_output = tmp[15:0];

    data_reg[`TMPH] = temperature_output[15:8];
    data_reg[`TMPL] = temperature_output[7:0];

end

endmodule
```