IDEAS AND CONCEPTS FOR SIGNAL GENERATION AND PROCESSING WITH GTM

Signal generation and processing with GTM

- “Analog” signal generation

- Closed loop control

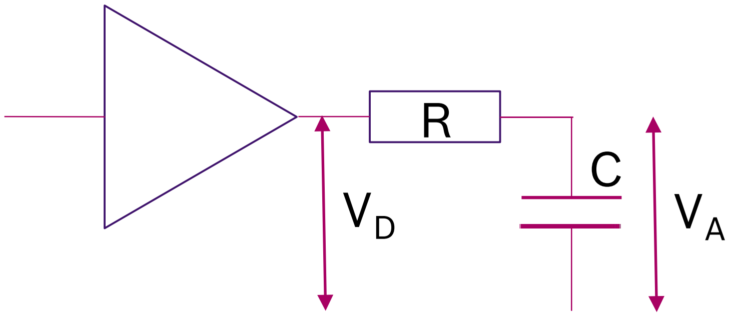
- FIR filter implementation with MCS

Signal generation and processing with GTM
“Analog” Signal Generation

- Nearly everything operates digital
  - Most micro controllers have no DA signal conversion capabilities
  - What can be done if applications need an amount of analog signals
    - Constant analog voltage
    - Periodic signals (sine wave, saw tooth ..)
- Can a GTM generated “analog” signal be an alternative

Possible solution:

- Hookup the digital output of a PCM modulated signal to a low pass filter

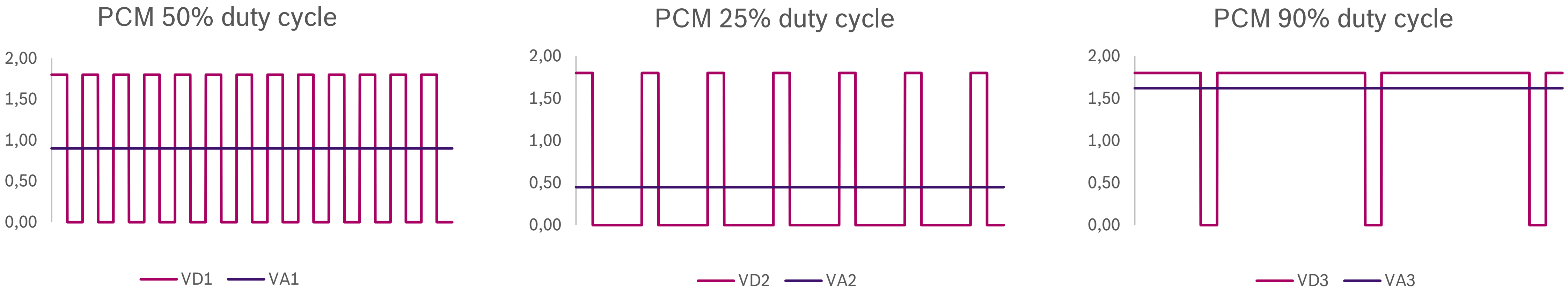


Signal generation and processing with GTM
“Analog” Signal Generation

Ripple of V_A depends on time constant T=R * C

Choose PCM clock period t_{PCM} << T to adjust ripple to application needs

Period 10 clock cycles



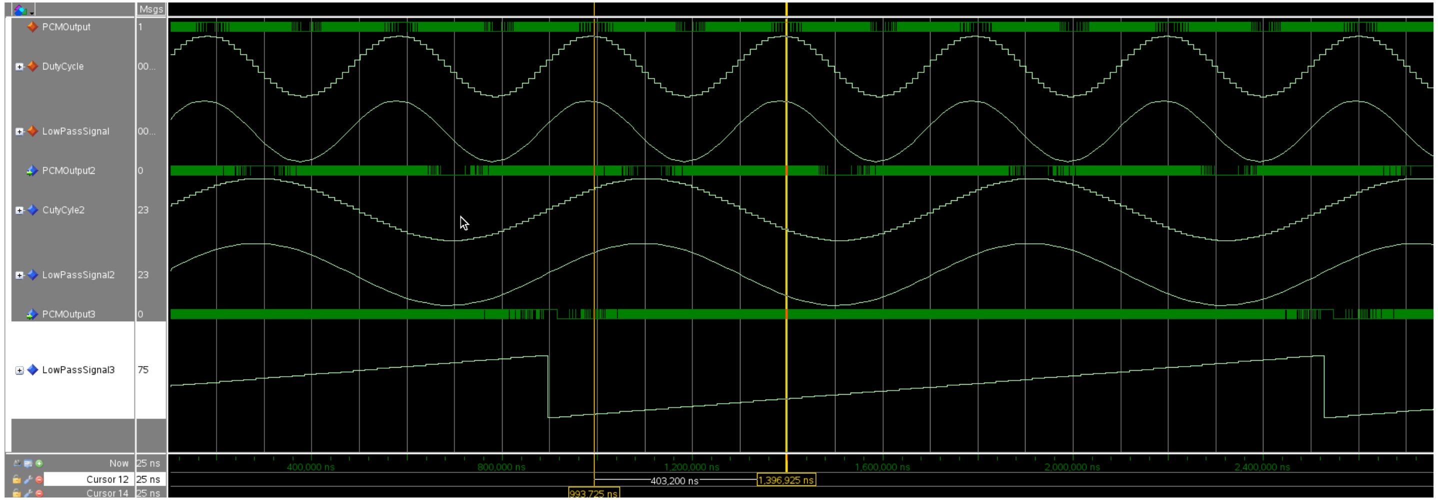
PCM clock = 100 MHz

7 Bit PCM: period 1,28 us ; Duty cycle can be adjusted in 128 steps

10 Bit PCM: period 10,24 us ; Duty cycle can be adjusted in 1024 steps

Signal generation and processing with GTM
“Analog” Signal Generation

- Any complex signal waveforms can be programmed by changing the PCM duty cycle in each period
  Possible with:
  - PSM (ringbuffer) - ARU - ATOM
  - MCS – (A)TOM
  - CPU/DMA – A(TOM)
- Generation of sine wave, saw tooth, even nonperiodic analog voltage characteristics can be generated



6 bit resolution

7 bit resolution

7 bit resolution

Saw tooth defined by 4 points, linear interpolated in MCS

BOSCH

Signal generation and processing with GTM

- "Analog" signal generation
- Closed loop control
- FIR filter implementation with MCS

Signal generation and processing with GTM
Closed loop control

- Performed by code executed with a CPU Core on the micro, will need a certain time to react from an input change to an output reaction

  Delay times are caused by:
  - Code execution
  - Read / write latencies for accesses to registers in peripherals
  - Interrupt scheduling
  - Task scheduling by OS

- Typical delay times for closed loop control by CPU core : > 10 us

- Decrease delay times by using closed loop control inside GTM

  Accuracy will increase

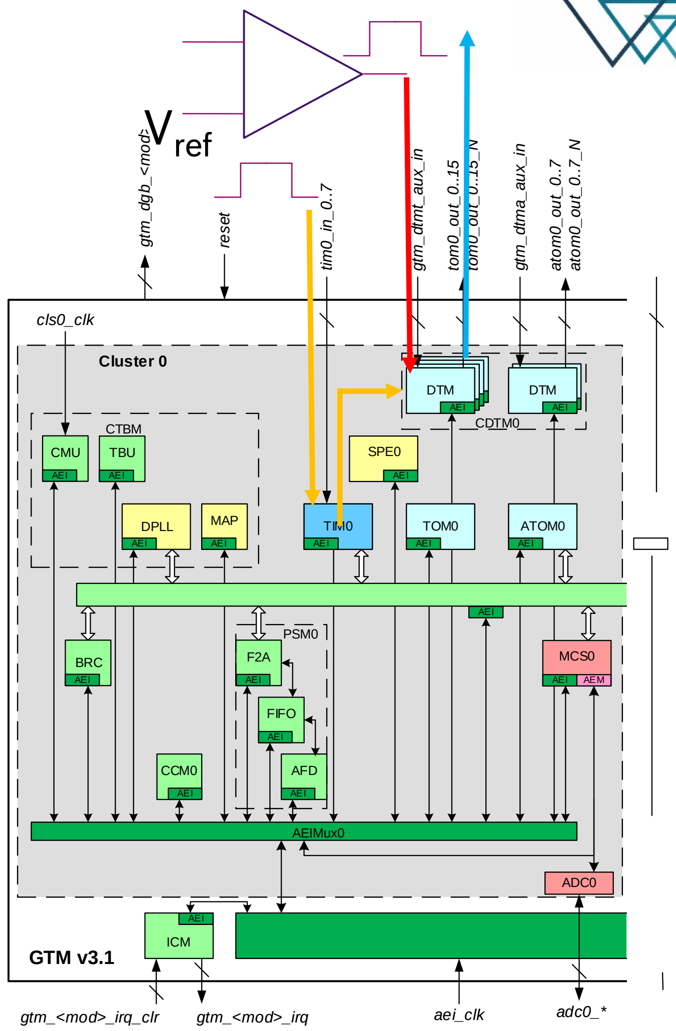
Signal generation and processing with GTM
Closed loop control

- Direct control of output by onchip comparator using gtm_dtmt_aux_in (red)
  - Tie output to 0 or 1
  - Switch between 2 (A)TOM channels

  Delay time: Combinatoric path (no delay)

- Direct control of output by TIM input (orange)
  - Tie output to 0 or 1
  - Switch between 2 (A)TOM channels

  Delay time: 3 system clock cycles delay + Filter delay (if enabled)



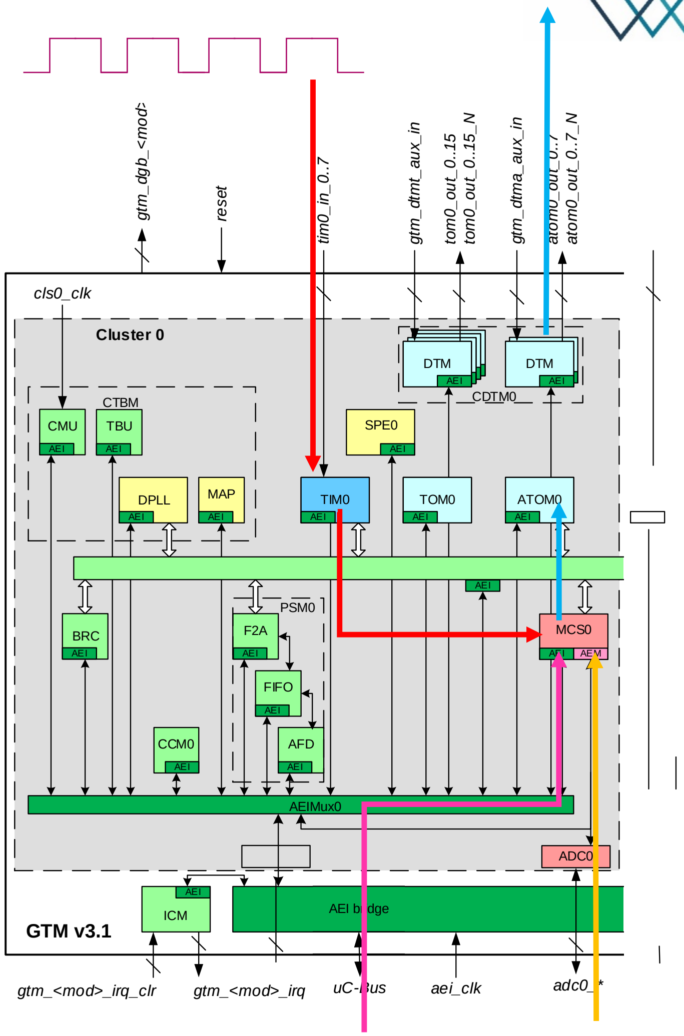
BOSCH

Signal generation and processing with GTM

Closed loop control

- MCS controls the regulation behalf of:
  - Input signal data (red)
    - Edge
    - PWM/Pulse measurement
    - Complex input signal e.g. serial protocol (LIN, SENT, SPI..)
  - Analog input (orange)
    - Measured with onchip ADC (voltage, current, temperature..)
  - Parameters provided by CPU / DMA (purple)
    - Computed by CPU
    - Fetched from Memory (DMA)
    - Received by ext. sensors via uC peripherals: (CAN, SPI, LIN,..)
  - Algorithm stored in MCS code
  - Parameter sets, Calibration data stored in MCS ram

Delay time: depending on complexity of calculations



BOSCH

Signal generation and processing with GTM
Closed loop control

- E.g: MCS running on 200 MHz; target delay time <= 1 us

- How complex can the regulation algorithm be ?
  - MCS operating 8 Channels in round robin mode.
    - target delay time <= 0,1 us: ~ 22 instructions per channel
    - target delay time <= 1 us: ~ 222 instructions per channel

  - MCS operating 1 Channel in accelerated mode.
    - target delay time <= 0,1 us: ~ 200 instructions per channel

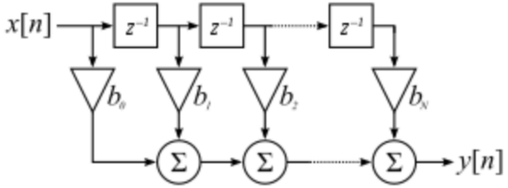
Signal generation and processing with GTM

- "Analog" signal generation
- Closed loop control
- FIR filter implementation with MCS

Signal generation and processing with GTM
FIR filter implementation

- FIR Filters are commonly used for digital signal processing

$$y[n] = b_0 x[n] + b_1 x[n-1] + \cdots + b_N x[n-N]$$
$$= \sum_{i=0}^{N} b_i \cdot x[n-i],$$



- Easy to implement on MCS
- MCS code size : 25 words
- MCS data size for coefficients b_i and input delay line x[n-i]
  - MCS standard ram_size : 3072 words ~ max 1500 taps

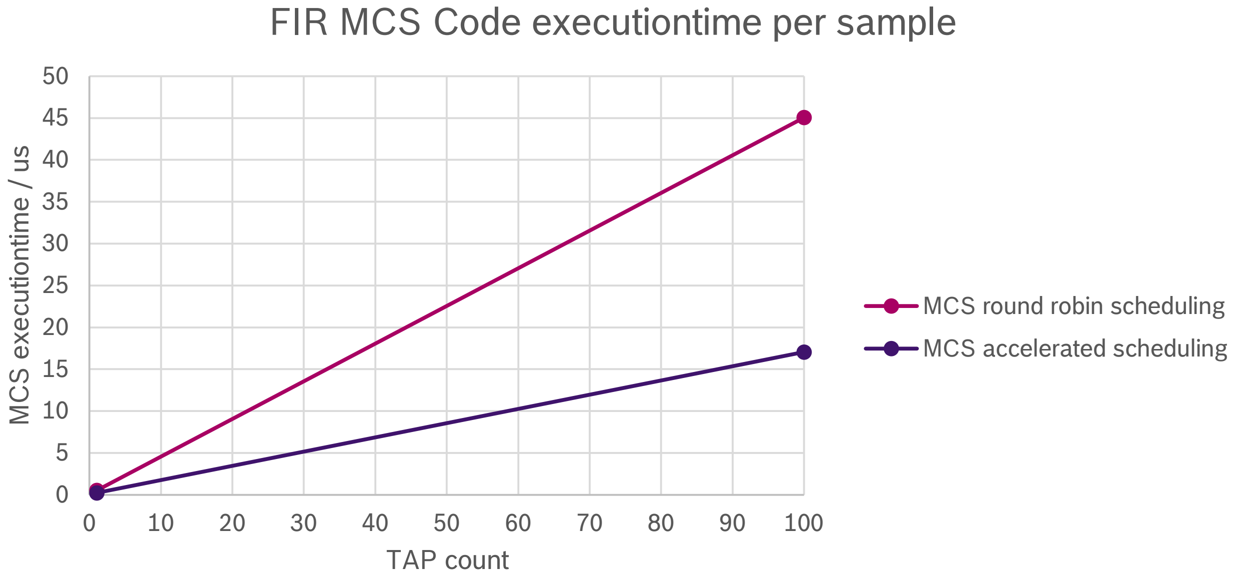
    mcfg borrow mode : 5120 words ~ max 2500 taps

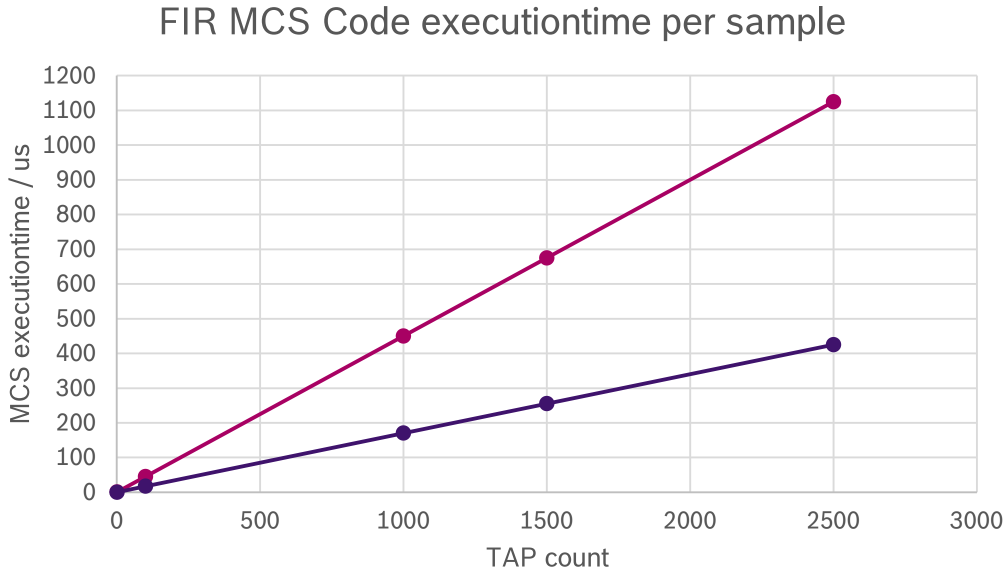
```
.org 0x0
    jmp    fir_init

.org 0x20
x_inp_v:  .var 0x0          # memory location for input sample x(n)
y_outp_v:  .var 0x0         # memory location for output sample y(n)

# reserve space for sample delay line
x_vec_v:
    .var 0x0
.org (x_vec_v+4*(tap_len_c-1))

# initialize vector with filter coefficients
h_vec_v:
.org (h_vec_v+4*tap_len_c)

fir_init:
    movl    R7, 4*(tap_len_c-2)     # initialize delay index
fir_sample_loop:
    mrd    R6, x_inp_v      # read input sample x
    mrd    R0, h_vec_v      # load coefficient h0
    mulu   R0, R6           # multiply x*h0
    movl   R5, 4*(tap_len_c-1)  # set coeff index to h[tap_len_c-1]
fir_mac_loop:
    mrdi   R1, R7, x_vec_v  # load delayed sample
    mrdi   R2, R5, h_vec_v  # load coefficient
    mulu   R1, R2           # multiply
    add    R0, R1           # accumuluate
    subl   R7, 4            # decrement delay index
    jbc    STA, N, fir_skip_delay_wrap_1  # branch if no wrap occured
    movl   R7, 4*(tap_len_c-2)  # reset delay index on wrapping
fir_skip_delay_wrap_1:
    subl   R5, 4            # decrement coeff index
    jbc    STA, Z, fir_mac_loop # branch tap_len_c-1 times to inner MAC loop
    mwr    R0, y_outp_v     # write filtered sample to RAM
    mwri   R6, R7, x_vec_v  # write actual sample to delay line
    movl   STA, 0x3         # rise IRQ
    subl   R7, 4            # decrement delay index
    jbc    STA, N,  fir_skip_delay_wrap_2     # branch if no wrap occured
    movl   R7, 4*(tap_len_c-2)  # reset delay index on wrapping
    jmp    fir_sample_loop
fir_skip_delay_wrap_2:
    nop                     # force equal sample time for each iteration
    jmp    fir_sample_loop
```


BOSCH

Signal generation and processing with GTM
FIR filter implementation

- execution time for a N tap FIR MCS implementation operating 200 MHz





FIR calculation performance in one MCS

- 8 channels round robin : 8 FIR filters with 10 tap each can be calculated in 5 us
- 1 channel accelerated : 1 FIR filter with 30 tap can be calculated in 5 us

BOSCH

Signal generation and processing with GTM
FIR filter implementation

- Cascading of filters is a usual technique
  - A high end GTM with 10 MCS provides 80 MCS channels
    - Via ARU data can be distributed from one MCS to others
  - Complex filters can make use of more than one MCS

Application: Audio signal processing with GTM

Typical data rate: 48 kHz/ 44,1 kHz sample rate 24 Bit samples

- N Audio signal input via serial protocol I2S: resources 3*N TIM channels
- MCS: audio signal processing (volume, mix, equalize, balance, fade)
  - 48 kHz ~ 20 us MCS processing time per sample
  - partioning of distinct functions to individual MCS channels
    - FIR with ~40 taps can be used
- M Audio signal output via serial protocol I2S: resources ~ 2*M ATOM channels

Signal generation and processing with GTM

Got you inspired ?

Try it out in your application

GTM makes it !