GTM Optimization of Electric Motor Algorithms

Robert Kearney & Robert Valascho

1 Common Motor Algorithms

2 MCS Implementation

3 Program Flow

4 MCS Improvements in GTM 1.x (A1G) -> 3.x (A2G)

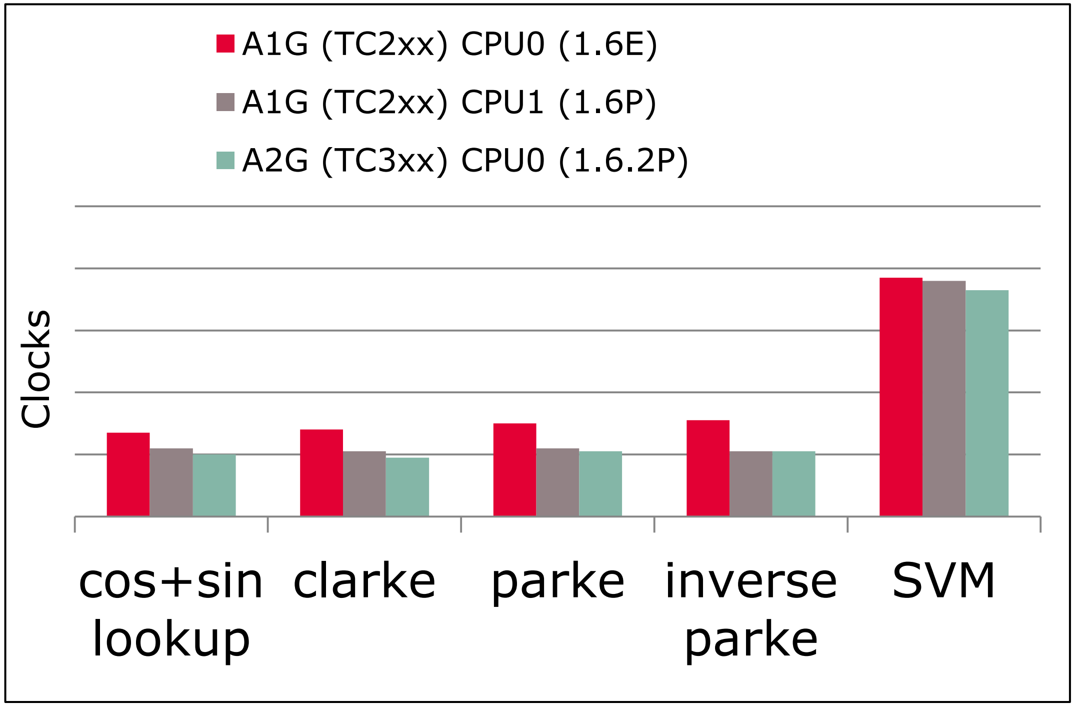
5 Aurix Benchmark 1G ---> 2G ---> 2G w/ MCS

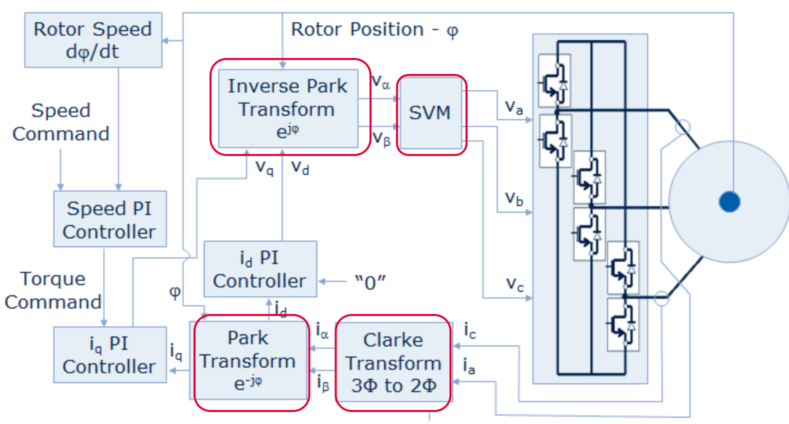
6 Future Optimizations

Common Motor Control Algorithms

- Trapezoidal Control
  - Block Commutation
  - Hall Effect sensors
  - 120 degree commutation scheme
- Sinusoidal Control
  - High precision feedback
    - Resolver
    - Encoder
    - Magnetic Sensor
- **Field Oriented Control For PMSM Motors**
    - **Space Vector Modulation**
    - **High Efficiency**
    - **Reduced Torque Ripple**

CPU cycles
A1G 200/300Mhz, A2G 300Mhz

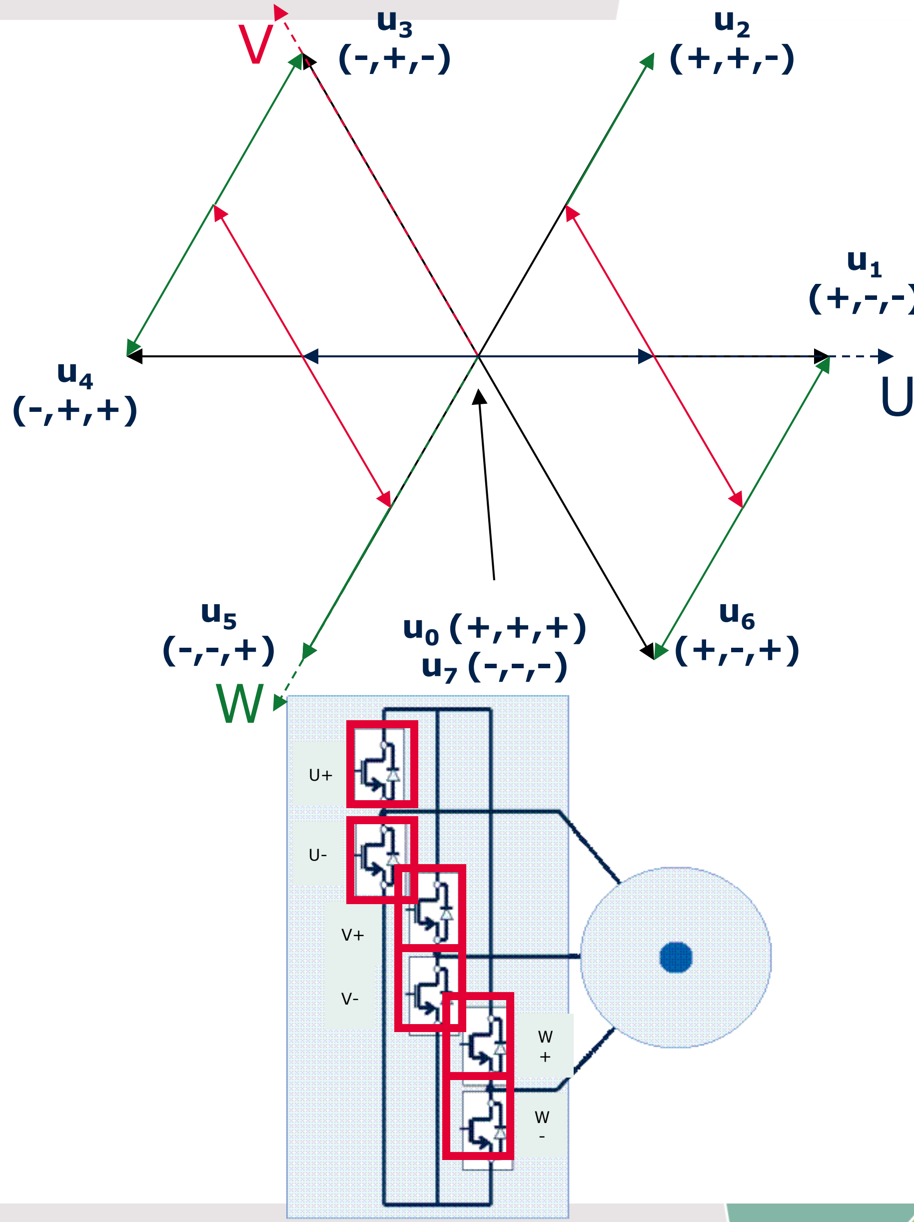




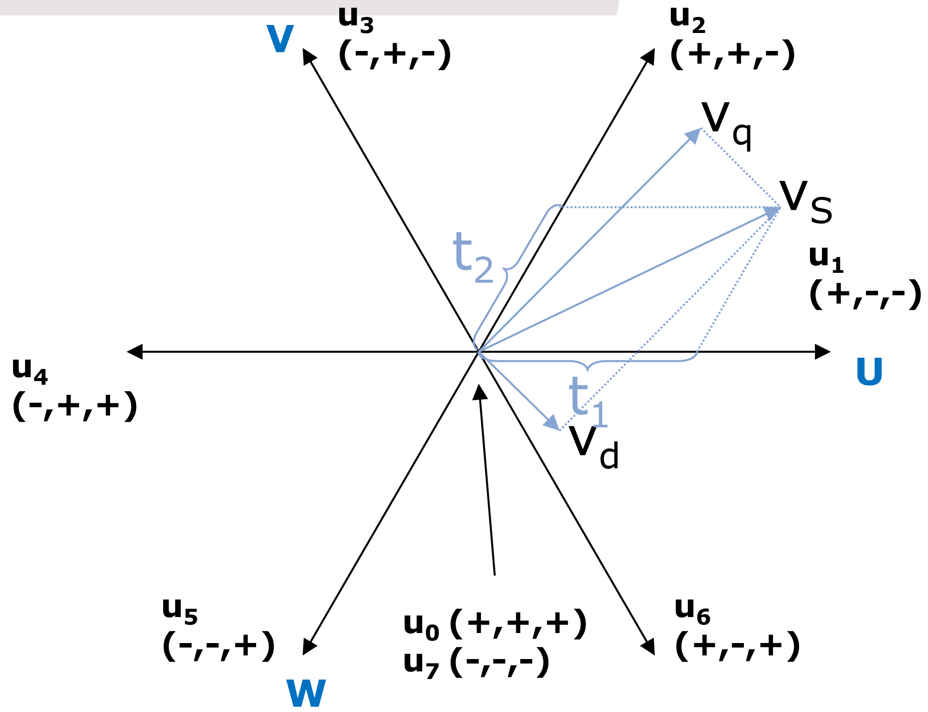
- Graph shows implementation of common motor transforms on different Aurix™ series devices
- SVM Algorithm is largest consumer of utilization
- SVM is essentially signal conditioning, and is a fitting task for MCS to handle
- How much can we save by offloading SVM?

Space Vector Modulation (SVM)
Overview

- We can use current/flux space vectors to transform the individual stator currents (i_u, i_v & i_w) into a space vector
- Similarly the vector summation of v_d & v_q is a voltage space vector v_S
- But how can we turn v_S into duty cycles or ON-times for the phase U, V, and W half bridges?
  - Project v_S onto the various switching states of the inverter
  - The length of the projections correspond to the time each switching state should be active
  - Be careful, the u, v & w axes are not orthogonal



Space Vector Modulation (SVM)
Calculations



- The length of the projections of v_S onto the two closest switching states represents the time to apply each switching state
  - e.g. Apply switching state u_1 for $t_1 \cdot T_{PWM}$ sec, u_2 for $t_2 \cdot T_{PWM}$ sec
  - Apply one of the zero vectors (u_0 or u_7) for the remainder of the PWM period ($t_0 \cdot T_{PWM}$)
- If v_S lies outside of the first sector, then subtract 60° until it is in the first sector

$$\begin{array}{cc} \begin{bmatrix} T_u \\ T_v \\ T_w \end{bmatrix} = \begin{bmatrix} 2.(\boxed{T_k} + \boxed{T_{k+1}}) + T_0 \\ 2.\boxed{T_{k+1}} + T_0 \\ T_0 \end{bmatrix} & \begin{matrix} t_1 \\ t_2 \\ t_3 \end{matrix} \end{array}$$

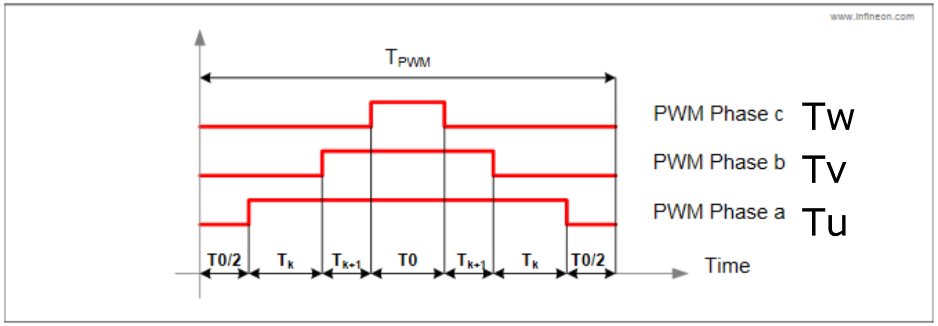


Figure 9: Timing diagram for voltage vector in sector 1

AP32013 Hybrid Kit Inverter App Note..

Space Vector Modulation (MCS Implementation)

MCS RAM Variables..Initialization

```
mcs0_ch0_ramBuffer:
atom_pwmhl:
mcs0_ch0_ramBuffer_ctrl:               .var 0
        ; bit 0: if set, SVM is computed from 'm'
        ; bit 1: if set, PWM is updated from 'ton'
mcs0_ch0_ramBuffer_channelCount:       .var 0
mcs0_ch0_ramBuffer_glbCtrlDisableUpdate:.var 0
mcs0_ch0_ramBuffer_glbCtrlEnableUpdate: .var 0
mcs0_ch0_ramBuffer_period:             .var 0
mcs0_ch0_ramBuffer_trigger:            .var 0
mcs0_ch0_ramBuffer_minPulse:           .var 0
mcs0_ch0_ramBuffer_maxPulse:           .var 0
mcs0_ch0_ramBuffer_ton:                .var 0
                                       .var 0
                                       .var 0
                                       .var 0
                                       .var 0
                                       .var 0
                                       .var 0
                                       .var 0
mcs0_ch0_ramBuffer_addrGlbCtrl:        .var 0
mcs0_ch0_ramBuffer_addrSrPeriod:       .var 0
mcs0_ch0_ramBuffer_addrSrTrigger:      .var 0
mcs0_ch0_ramBuffer_addrSr:             .var 0
                                       .var 0
                                       .var 0
                                       .var 0
                                       .var 0
                                       .var 0
                                       .var 0
                                       .var 0
mcs0_ch0_ramBuffer_m: ; DEBUG ONLY
                                       .var 0
                                       .var 0

mcs0_ch0_ramBuffer_runTimeSvm:         .var 0
mcs0_ch0_ramBuffer_runTimePwmUpdate:.var 0


mcs0_ch0_ramBuffer_end:
```

Tricore Initialization..MCS RAM Pointer

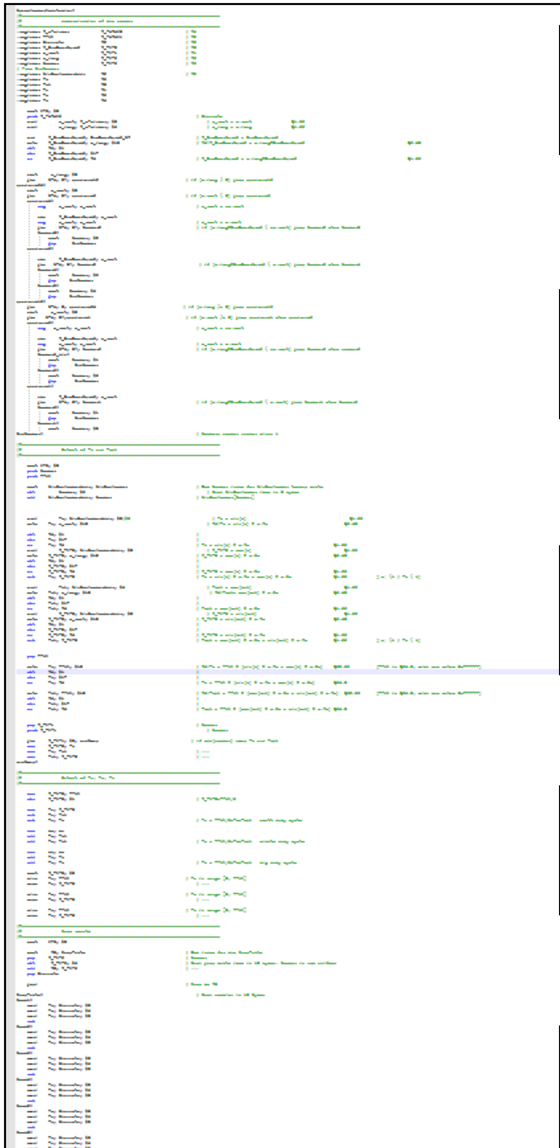
```
typedef struct
{
    uint32 ctrl;
    uint32 channelCount;
    uint32 glbCtrlDisableUpdate;
    uint32 glbCtrlEnableUpdate;
    uint32 period;
    uint32 trigger;
    uint32 minPulse;
    uint32 maxPulse;
    uint32 ton[IFXGTM_ATOM_PWMHL_MAX_NUM_CHANNELS];
    uint32 addrGlbCtrl;
    uint32 addrSrPeriod;
    uint32 addrSrTrigger;
    uint32 addrSr[IFXGTM_ATOM_PWMHL_MAX_NUM_CHANNELS];
    fract mReal;
    fract mImag;
    sint32 runTimeSvm;
    sint32 runTimePwmUpdate;
}McsPwmHl;
McsPwmHl *mcsPwmHlData = NUL
```

Pointer Reflects MCS RAM Buffer

Init of MCS RAM

```
boolean App_initMcsData(App *app)
{
    int i = 0;
    { /* Init mcs variables */

        mcsPwmHlData = (McsPwmHl *)&pGtmMcsMem[LABEL_MCSX_ATOM_PWMHL];;
        mcsPwmHlData->ctrl = 0;
        mcsPwmHlData->channelCount = app->drivers.pwmDriverDtm.base.channelCount;
        mcsPwmHlData->glbCtrlDisableUpdate = app->drivers.timerDriver.agcDisableUpdate;
        mcsPwmHlData->glbCtrlEnableUpdate = app->drivers.timerDriver.agcApplyUpdate;
        mcsPwmHlData->period = IfxGtm_Atom_Timer_getPeriod(&app->drivers.timerDriver);
        mcsPwmHlData->trigger = IfxGtm_Atom_Timer_getTrigger(&app->drivers.timerDriver);
        mcsPwmHlData->minPulse= app->drivers.pwmDriverDtm.base.minPulse;
        mcsPwmHlData->maxPulse = app->drivers.pwmDriverDtm.base.maxPulse;
        for (i=0;i<IFXGTM_ATOM_PWMHL_MAX_NUM_CHANNELS;i++)
        {
            mcsPwmHlData->ton[i] = 0;
            mcsPwmHlData->addrSr[i] = 0;
        }

        // Manual mapping. FIXME implement API for automated mapping using table
        mcsPwmHlData->addrGlbCtrl = MCS_BM_ATOM_AGC_GLB_CTRL;
        mcsPwmHlData->addrSrPeriod = MCS_BM_ATOM_CH4_SR0;
        mcsPwmHlData->addrSrTrigger = MCS_BM_ATOM_CH4_SR0;
        mcsPwmHlData->addrSr[0] = MCS_BM_ATOM_CH5_SR0;
        mcsPwmHlData->addrSr[1] = MCS_BM_ATOM_CH6_SR0;
        mcsPwmHlData->addrSr[2] = MCS_BM_ATOM_CH7_SR0;
        mcsPwmHlData->mReal = 0;
        mcsPwmHlData->mImag = 0;

    }

    return TRUE;
}
```

Space Vector Modulation (MCS Implementation)



SpaceVectorModulation(Vs, Switching Period, GTM_Channels)

Determine 60 degree segment to decide U,V,W role

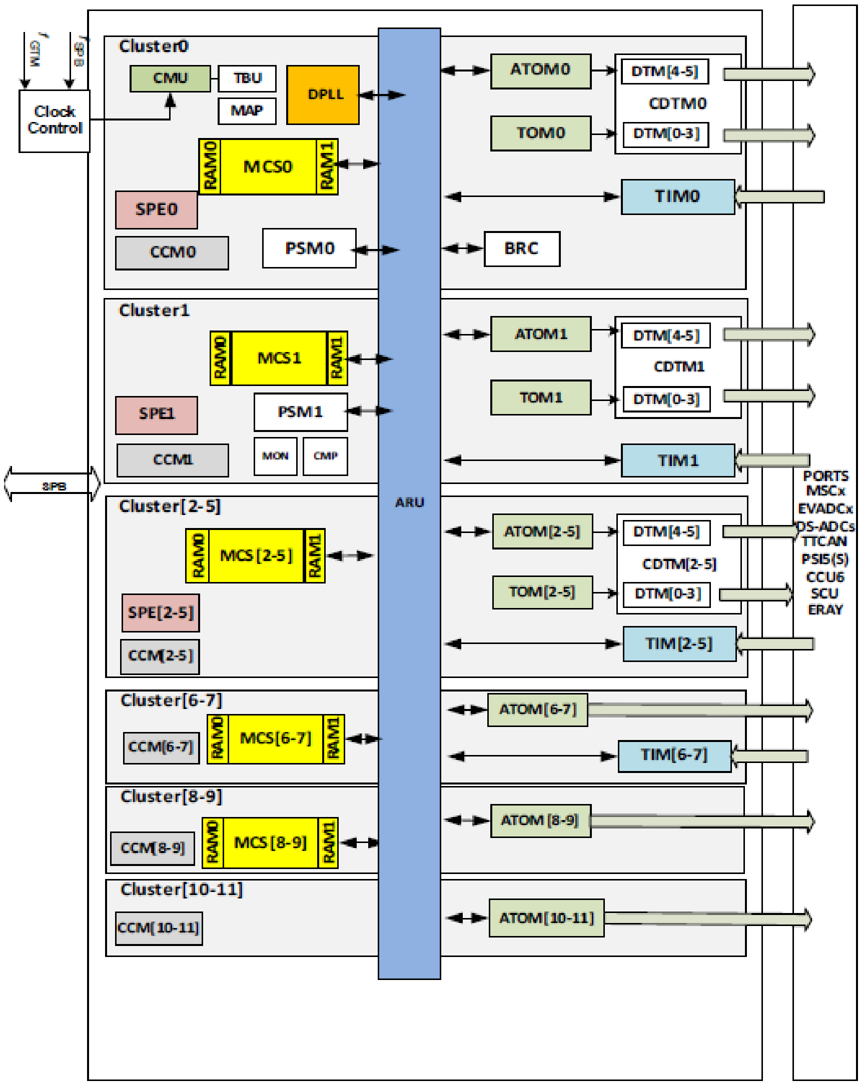
Calculate T_k, and T_{k+1}

Convert T_k, and T_{k+1} to U, V, and W Switching Times

Physically Update the SRx Registers of the PWM Timers

**BWRI improvement, over ARU*

GTM (Generic Timer Module) AURIX2G - GTM V3.1.x



- GTM 3.1.5.x is logically divided in Clusters
- The first 5 clusters (0-4), can work up to 200MHz
- The DTM numbering is changed. They are grouped in CDTMx modules (Cluster Dead Time Module)
- New CCMx Modules (Cluster Configuration Module)
  - Cluster Clock Frequency
  - Module Clock Gating
  - MCS bus master observation
  - Address Range Protection
- ARU Routing extended
- MCS Instructions set Increased
- New MCS Bus inside every cluster

AURIX vs AURIX2G
GTM Wrapper Major Changes

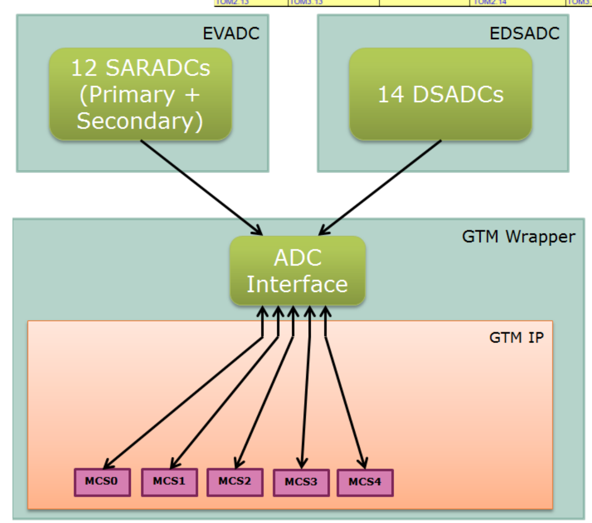
	Aurix	Aurix 2G
GTM2PORTS	4 selectable GTM OUTs for every PORT	12 selectable GTM OUTs for every PORT
ADC/DSADC	2 Selectable Triggers to ADC	5 Selectable Triggers to ADC
MCS Connections to ADC	None	MCS 0 – 4 Result for DSADC and SARADC

Port	TOUT	Output Timer Mapped (TOUTSELn (n = 0-33))											
		A	B	C	D	E	F	G	H	I	J	K	L
P00.0	TOUT 9	TOM0 _8	TOM1 _0	ATOM 0_0	ATOM 1_0	TOM0 _0N	TOM1 _5N	TOM0 _4	TOM1 _4	ATOM 0_5N	ATOM 1_5N	ATOM 2_1N	Reserved
P00.1	TOUT 10	TOM0 _9	TOM1 _1	ATOM 0_1	ATOM 1_1	TOM0 _1N	TOM1 _6N	TOM0 _4N	TOM1 _4N	ATOM 0_6N	ATOM 1_6N	ATOM 2_2N	Reserved
P00.2	TOUT 11	TOM0 _9	TOM1 _1	ATOM 0_1	ATOM 1_1	TOM0 _2N	TOM1 _6N	TOM0 _5	TOM1 _5	ATOM 0_6N	ATOM 1_6N	ATOM 2_3N	Reserved
P00.3	TOUT 12	TOM0 _10	TOM1 _2	ATOM 0_2	ATOM 1_2	TOM0 _3N	TOM1 _7N	TOM0 _5N	TOM1 _5N	ATOM 0_7N	ATOM 1_7N	ATOM 2_0N	Reserved
P00.4	TOUT 13	TOM0 _11	TOM1 _3	ATOM 0_3	ATOM 1_3	Reserved	TOM1 _4N	TOM0 _6	TOM1 _6	ATOM 0_4N	ATOM 1_4N	Reserved	Reserved
P00.5	TOUT 14	TOM0 _12	TOM1 _4	ATOM 0_4	ATOM 1_4	TOM0 _1N	TOM1 _1N	TOM0 _6N	TOM1 _6N	ATOM 0_1N	ATOM 1_1N	ATOM 3_1N	Reserved
P00.6	TOUT 15	TOM0 _13	TOM1 _5	ATOM 0_5	ATOM 1_5	TOM0 _2N	TOM1 _2N	TOM0 _7	TOM1 _7	ATOM 0_2N	ATOM 1_2N	ATOM 3_2N	Reserved
P00.7	TOUT 16	TOM0 _14	TOM1 _6	ATOM 0_6	ATOM 1_6	TOM0 _3N	TOM1 _3N	TOM0 _7N	TOM1 _7N	ATOM 0_3N	ATOM 1_3N	ATOM 3_3N	Reserved
P00.8	TOUT 17	TOM0 _15	TOM1 _7	ATOM 0_7	ATOM 1_7	TOM0 _0N	TOM1 _0N	Reserved	Reserved	ATOM 0_0N	ATOM 1_0N	ATOM 3_0N	Reserved
P00.9	TOUT 18	TOM0 _0	TOM1 _0	ATOM 2_0	ATOM 3_0	Reserved	TOM1 _5N	Reserved	Reserved	ATOM 3_5N	ATOM 4_5N	Reserved	Reserved
P00.10	TOUT 19	TOM0 _1	TOM1 _1	ATOM 2_1	ATOM 3_1	Reserved	TOM1 _6N	Reserved	Reserved	ATOM 3_6N	ATOM 4_6N	Reserved	Reserved

Legacy mapping
New mapping

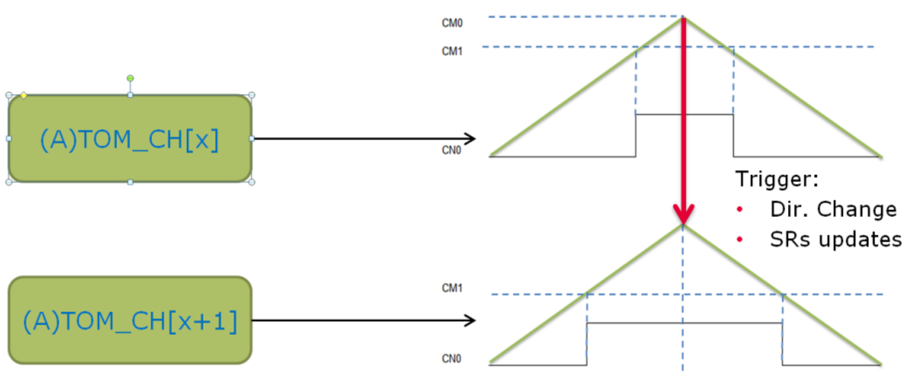
GTM to EVADC triggers

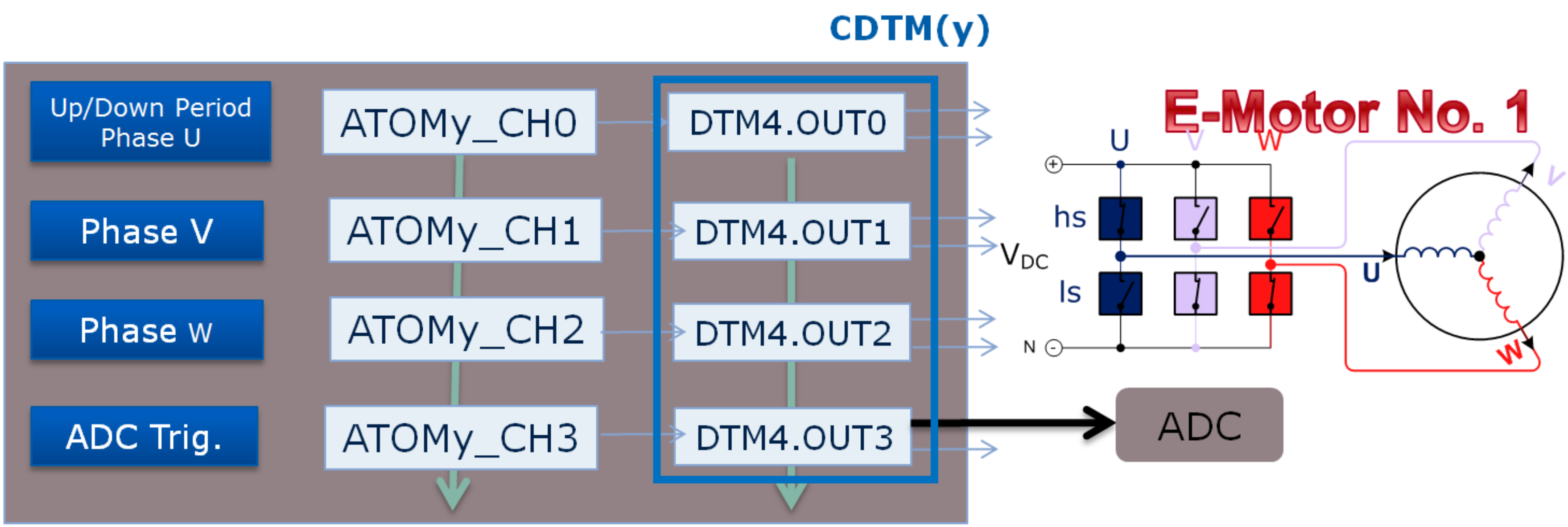
Trig 0	Trig 1	Trig 2 (also 33x)	Trig 3	Trig 4	Trig 0 (33x)	Trig 1 (33x)
SEL0/1/2	SEL0/1/2	SEL0/1/2	SEL0/1/2	SEL0/1/2	SEL0/1/2	SEL0/1/2
No trigger	No trigger	No trigger	No trigger	No trigger	No trigger	No trigger
TOM0.6	TOM1.6	TOM0.3	ATOM0.3	TOM2.3	TOM0.1	TOM1.1
TOM0.7	TOM1.7	TOM0.4	ATOM0.5_N	TOM2.4	TOM0.2	TOM1.2
TOM0.13	TOM1.13	TOM0.5_N	ATOM1.3	TOM2.5_N	TOM0.3	TOM1.3
TOM0.14	TOM1.14	TOM0.6_N	ATOM1.5_N	TOM2.7	TOM0.4	TOM1.4
ATOM0.4	ATOM1.4	TOM0.7	ATOM2.3	TOM2.11	TOM0.5	TOM1.5
ATOM0.5	ATOM1.5	TOM0.11	ATOM2.5_N	TOM2.12	TOM0.6	TOM1.6
ATOM0.6	ATOM1.6	TOM0.15	ATOM3.3	TOM3.3	TOM0.7	TOM1.7
ATOM0.7	ATOM1.7	TOM1.3	ATOM3.4	TOM3.4	TOM0.8	TOM1.8
ATOM1.4	TOM0.14	TOM1.4	ATOM3.5_N	TOM3.5_N	TOM0.9	TOM1.9
ATOM1.5	TOM0.15	TOM1.5_N	ATOM3.7	TOM3.7	TOM0.10	TOM1.10
ATOM1.6	ATOM0.4	TOM1.6_N	ATOM4.3	ATOM5.3	TOM0.11	TOM1.11
ATOM1.7	ATOM0.5	TOM1.7	ATOM4.4	ATOM5.4	TOM0.12	TOM1.12
ATOM2.4	ATOM0.6	TOM1.11	ATOM4.5_N	ATOM5.5_N	TOM0.13	TOM1.13
ATOM2.7	ATOM0.7	TOM1.15	ATOM4.7	ATOM5.7	TOM0.14	TOM1.14
TOM2.13	TOM3.13		TOM2.14	TOM3.14	TOM0.15	TOM1.15



Specific Advantages for Motor Control

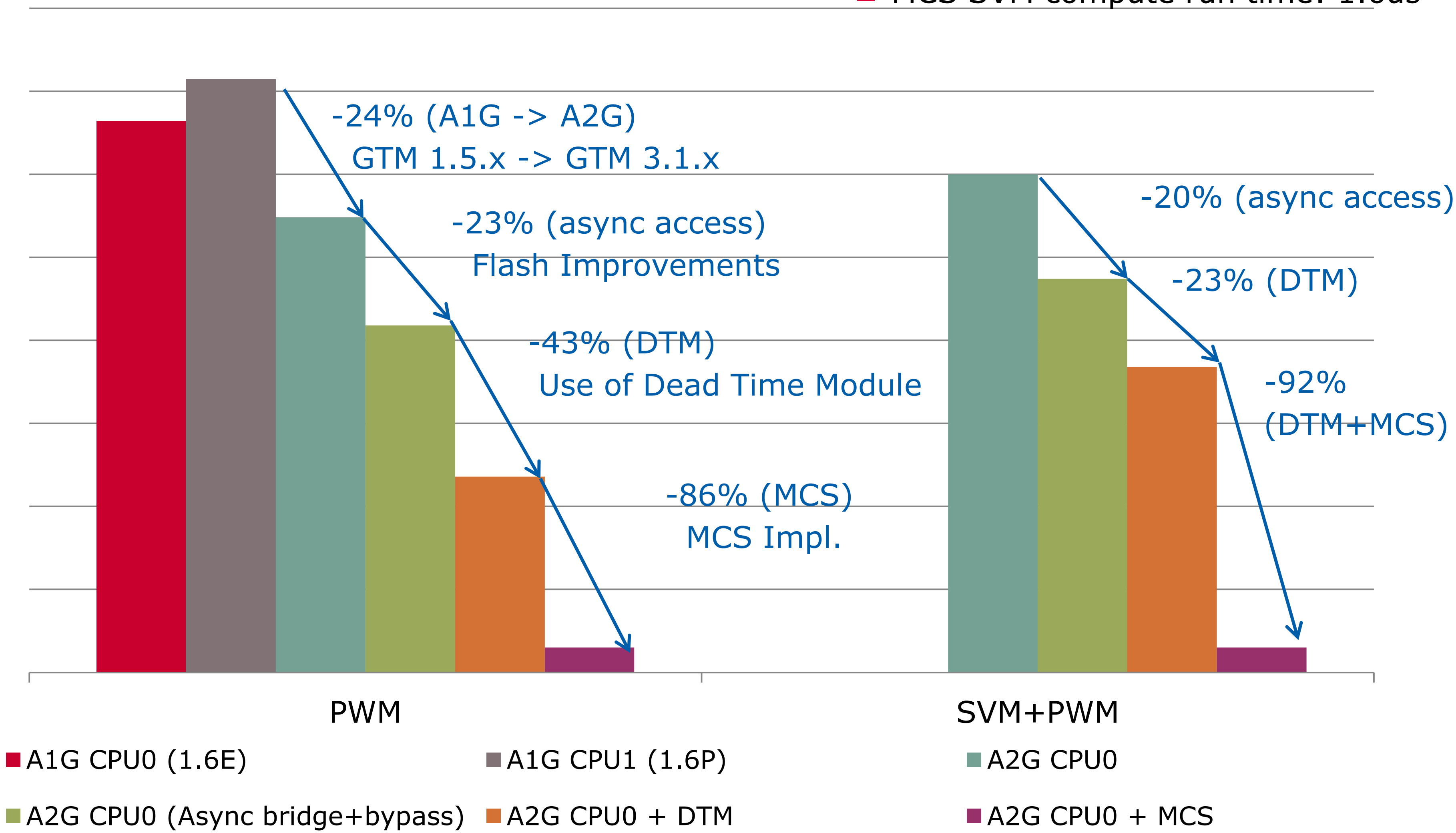
- Motor Control
  - New Up-Down Mode
  - DTM usage with Trigger to ADC



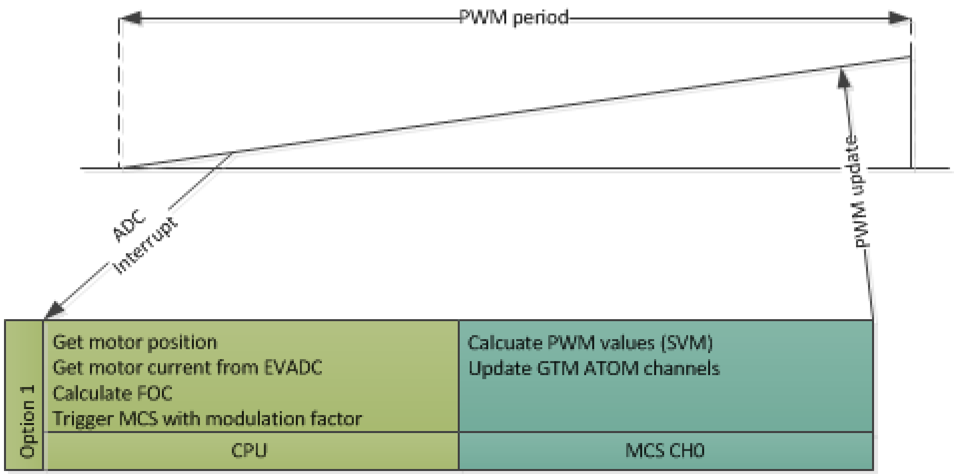


Benchmark Results
A1G 200Mhz, A2G 300Mhz

- MCS PWM update run time: 3.6us
- MCS SVM compute run time: 1.6us

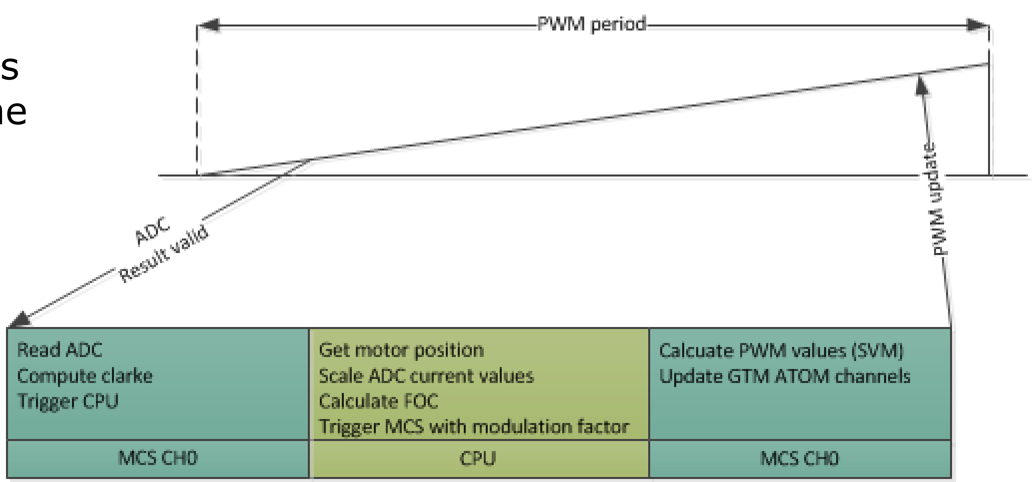


Future Optimizations



- › Today's GTM demo implementation just includes MCS management for the Space Vector Modulation Algorithm.
- › In theory, with the expansion of MCS capabilities in > GTM 3.1.x, we could implement the entire FOC algorithm on the MCS core

- › TC3xx devices feature direct access features to feedback peripherals, such as ADC, which will give direct hard real-time paths to the "sense" half of FOC control.
- › This will increase possibilities to directly offload the TriCore CPU.



Part of your life. Part of tomorrow.