AN-1118
Eight Output 8-bit PCM LED Dimming Using ASM

Author: Yu-Han Sun
Date: August 31, 2016

Introduction

This application note will explain how to create an eight output LED controller, each channel with 8-bit resolution PCM (Pulse Code Modulation), using little more than the ASM block in GreenPAK5. Furthermore, the I2C interface enables offloading LED driving functions from an MCU, as shown in Figure 1. Only a portion of GreenPAK5's resources are needed in this implementation; many blocks are left available to implement other user functions.

Background

The ASM block in GreenPAK5 has built-in capability to simultaneously drive 8 outputs each with individual 8-bit codes which can be set arbitrarily by I2C. An 8-bit serial code which represents analog signal is referred to as Pulse Code Modulation (PCM).

In this application, we convert the PCM code to an analog dimming level by binary-weighting the serial timing of each output bit such that time-averaged voltage is proportional to the binary value of the code.

The result is very similar to PWM (Pulse Width Modulation) where the digital signal can be converted to analog simply by filtering.

The difference between the PCM scheme and PWM can be seen in Figure 2. Both the PCM and PWM waveforms have the same average value of 700mV. However, PWM requires fewer transitions per cycle, and thus can more suitable for applications where energy lost from switching needs to be minimized. However, in LED applications the cycle rate is relatively slow, so the additional transitions of the PCM dimming scheme are not problematic.

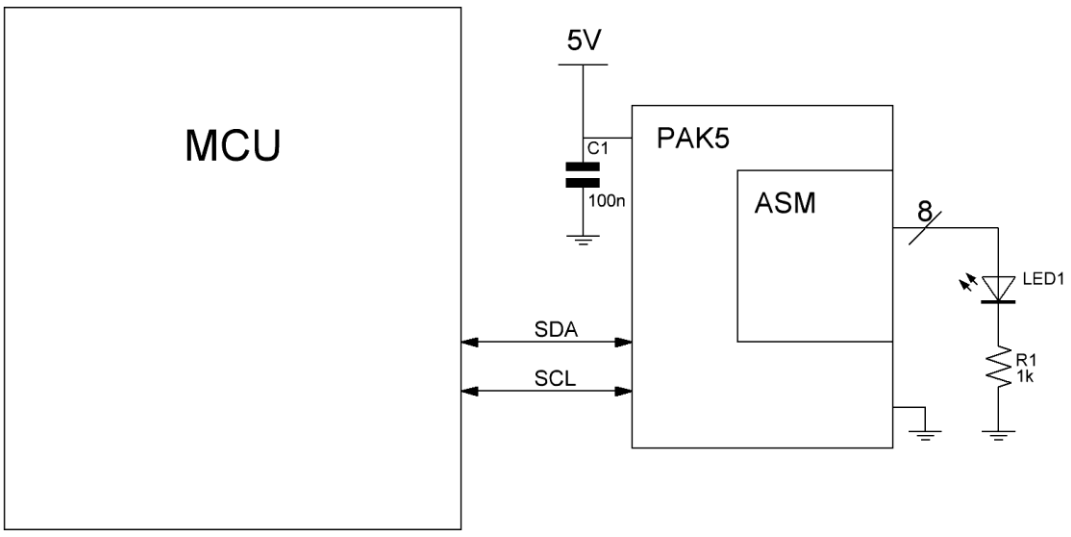


Figure 1. System Level View

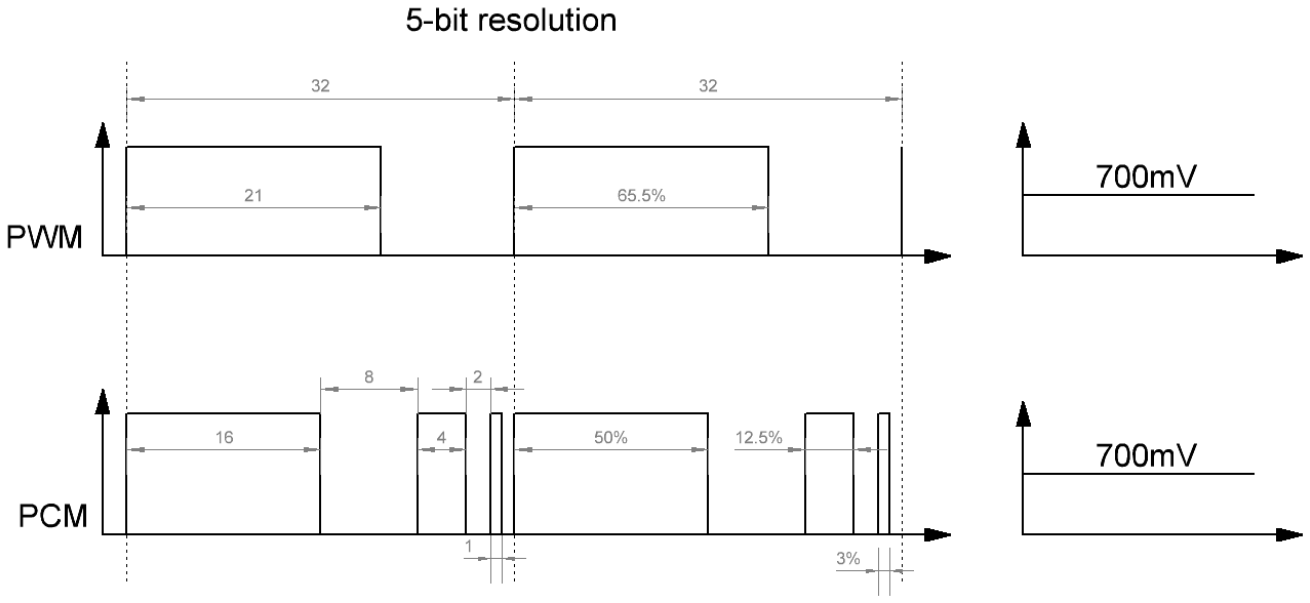


Figure 2. PWM vs PCM

The advantage is that the natural translation from binary code to PCM is more resource efficient for implementation in GreenPAK5.

GreenPAK Design, the ASM states

The design begins with the ASM, an eight-state asynchronous state machine. Each state represents a binary bit in the code.

	State0	State1	State2	State3	State4	State5	State6	State7
Bit	bit0 (LSB)	bit1	bit2	bit3	bit4	bit5	bit6	bit7 (MSB)
Period (T)	T/256	T/128	T/64	T/32	T/16	T/8	T/4	T/2
Duty %	0.390625%	0.78125%	1.5625%	3.125%	6.25%	12.5%	25%	50%
Transitions	CLK/2	CLK/4	CLK/8	CLK/16	CLK/32	CLK/64	CLK/128	Restart signal

Table 1. ASM State Machine

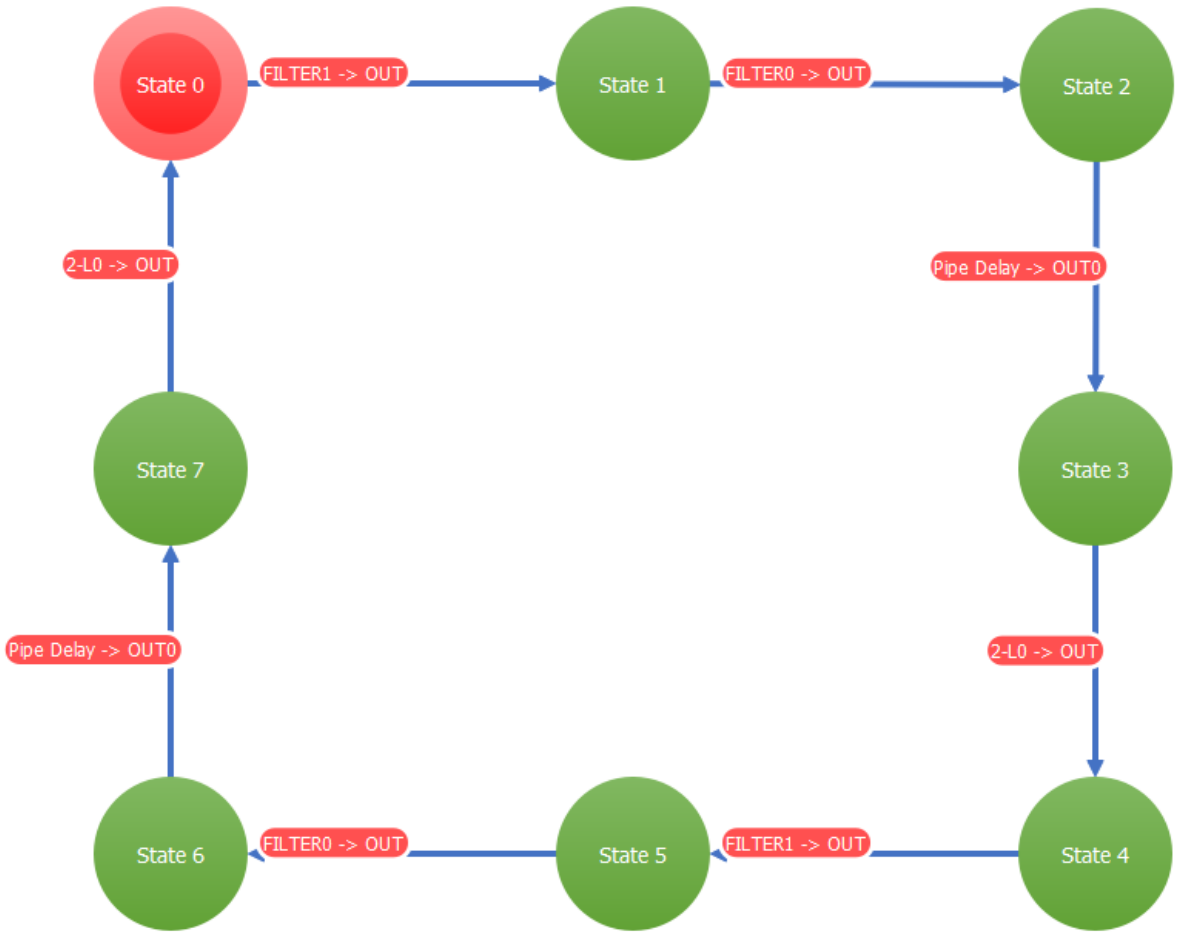


Figure 3. ASM State Machine

The state machine loops through all of the states, making sure each state lasts a specified amount of time. One complete loop through the ASM corresponds to one period. See the state diagram in Figure 3. The state diagram depicts the direction of state transitions starting with State 0, the initial state, and travels clockwise to State 7. State 7 then loops back to State 0, starting a new period.

State 0 represents the least significant bit, bit0, while State 7 represents the most significant bit, bit7. Refer to Table 1 for the bit, fraction of the period and duty cycle that each state represents.

The state transitory signals occurs at strict intervals to ensure each state lasts only a specified amount of time.

This amount is a fraction of the period calculated as $\frac{1}{2^7}, \frac{1}{2^6}, \frac{1}{2^5}, \cdots \frac{1}{2^0}$ for States 0 to 7 respectively. Refer to Figure 4. Each OSC division is connected to sequential transitioning signals. Each subsequent state lasts twice as long as the previous one since each subsequent state is a binary place to the left. I.e. State 0 has a duration of T/256 and State 1 has a period of T/128.

GreenPAK Design, the ASM inputs

The ASM state transitory inputs must be a binary division of the CLK, as explained in the previous section. In this design, the divisions are done with DFFs. There are a few ways to implement this and we will be looking at a brute force method as well as a more elegant solution.

The first method is chaining one DFF after the next to get 8 divided clocks. The second method is using a PGEN to generate all necessary transition signals. These are discussed below.

Note 1: It is important to remember that ASM transition inputs are active high, and that will be taken into consideration when generating ASM input signals.

Method #1 – Eight DFFs

The first method uses a total of 8 DFF, one for each ASM input. Here, the Pipe Delay is used as a DFF but with an extra inverted output. Each DFF toggles to clock the next DFF such that each DFF is half the frequency of its preceeding DFF, generating the necessary $\frac{1}{2^7}, \frac{1}{2^6}, \frac{1}{2^5}, \cdots \frac{1}{2^0}$ input signals.

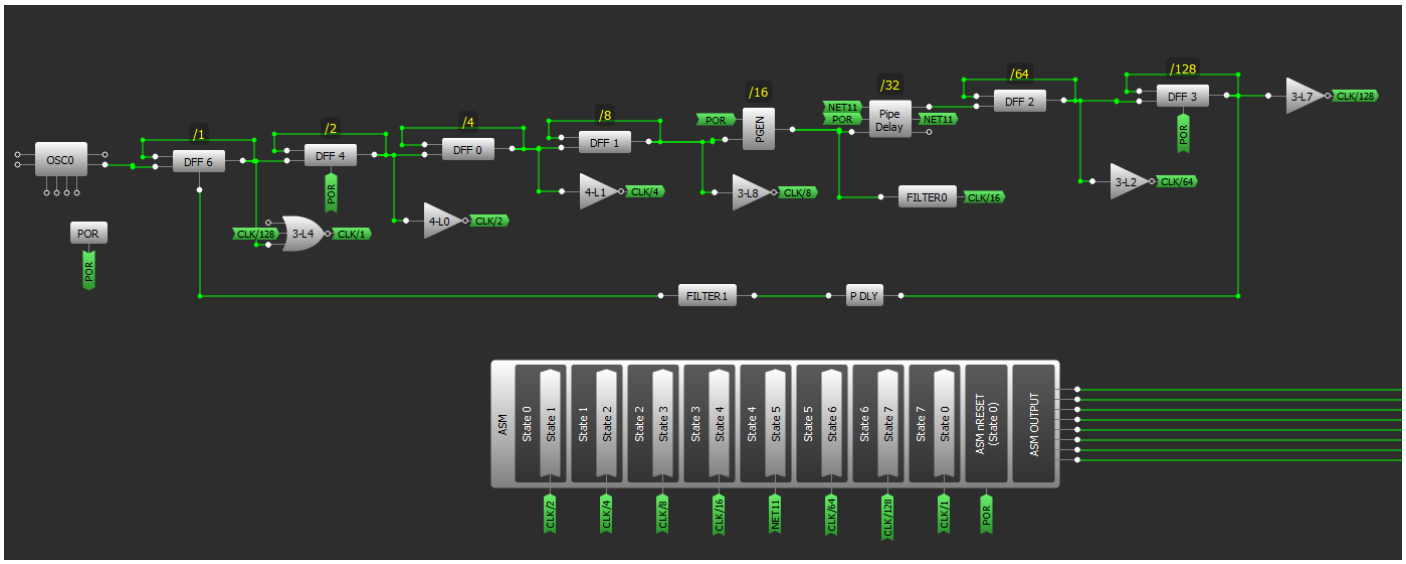


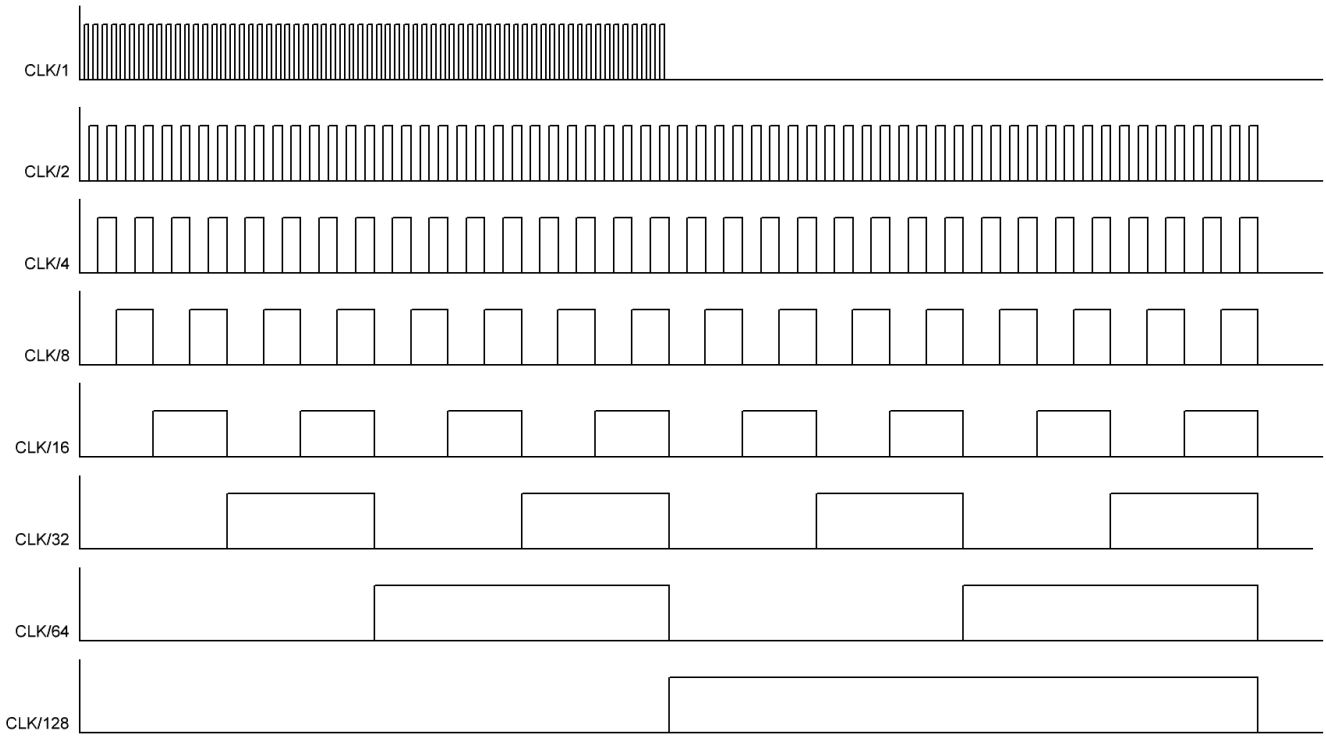
Figure 4. Variation #1, Eight DFFs

Each DFF output is inverted to prevent all signals from being high at the same time, which would cause the ASM to loop in a circle. Lastly, the cycle resets through a rising edge detector (PDLY) at the end of each period, this will change the period width from 256 to 255, the effect of which is discussed in the section 'Resolution'.

Notice DFF6 is NORed with CLK/128 such that when CLK/128 goes high, CLK/1 is forced to zero as seen in Timing Diagram 1. This is also to prevent both CLK/1 and CLK/128 from being high at the same time which would cause an unwanted transition from State 7 to State 0.

Method #2 – PGEN and Clock Multiplexing

The second method uses one PGEN and a clock multiplexing circuit to generate the timed pulses. The PGEN is configured with a pattern of 1s and 0s as see the configuration in Figure 6. The pattern's 1's and 0's are selected such that the rising and falling edges correspond with the timing of $\frac{1}{2^7}, \frac{1}{2^6}, \frac{1}{2^5}, \ldots \frac{1}{2^0}$ signals. EDGE_DET0 detects rising edges and EDGE_DET1 detects falling edges. The output of these blocks pulse high when the component detects an edge.



Timing Diagram 1. ASM inputs for Variation #1

We use two edge detectors instead of one such that every other ASM input has the same source.

The PGEN is limited to only 16 bit patterns, which is only enough bits for four consecutive edges. In order to stretch the PGEN to the next 4 edges, the input to the PGEN clock is multiplexed between a fast 2MHz/8 clock and a slow 2MHz/8/16 clock. The multiplexer's select bit comes from DFF5, which toggles from high to low when two RED pulses have been detected. DFF5 also enables the /16 Pipe Delay divider.

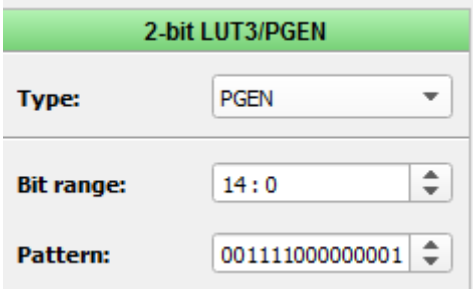


Figure 6. PGEN Configuration

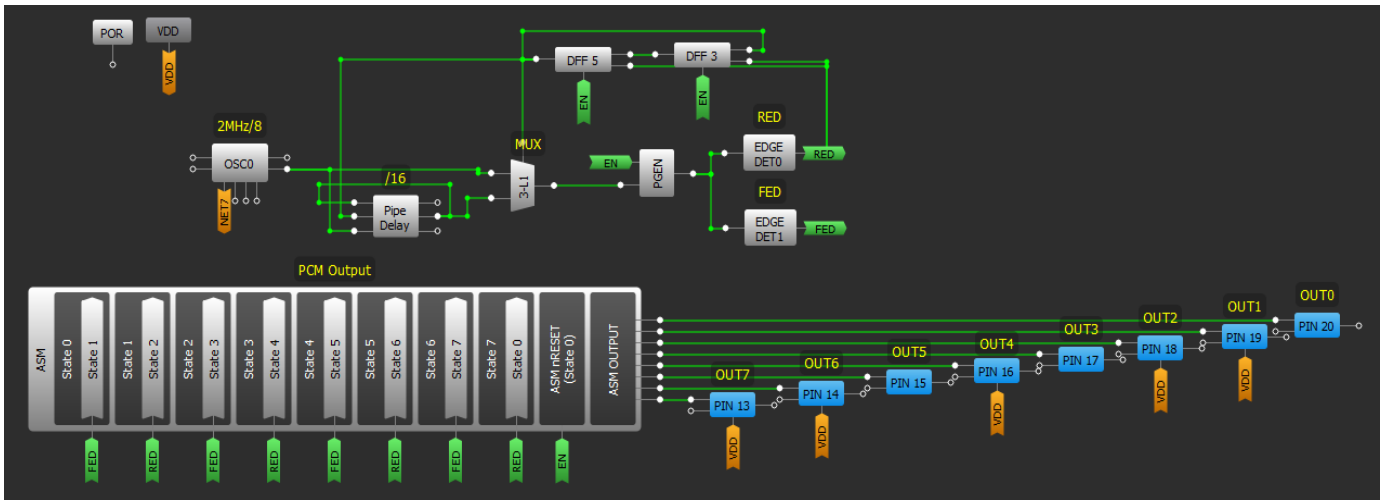
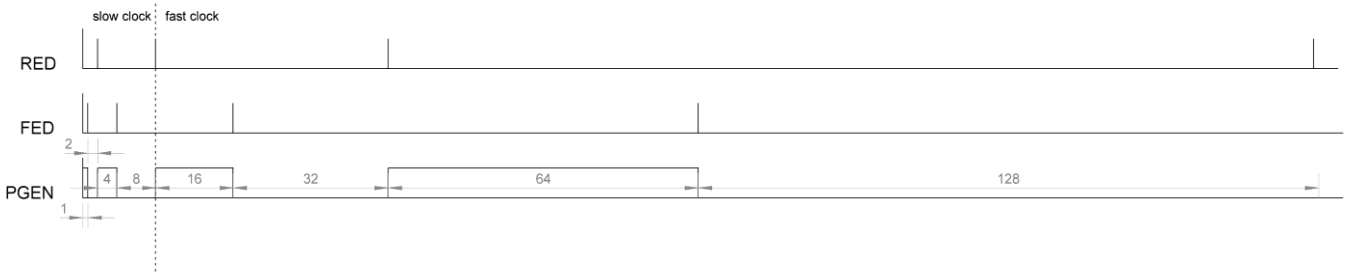


Figure 5. Variation #2. ASM Transition signals generated by DFFs



Timing Diagram 2. ASM Inputs for Variation #2

Comparison

Method #1 uses 17 components while Method #2 uses just 7, leaving at most, 12 combination function components, 7 of which can be configured as a counter. While the first design is easy to understand and port into another project, the multiplexing in the second method conserves more resources. In this app note, we will continue analyzing the pgen and clock multiplexing solution.

	Method #1 – Eight DFFs	Method #2 – PGEN and Clock Multiplexing
Used macrocells	16 + 1 PDLY	7
Leftover macrocells	3	12
Segments	255	255

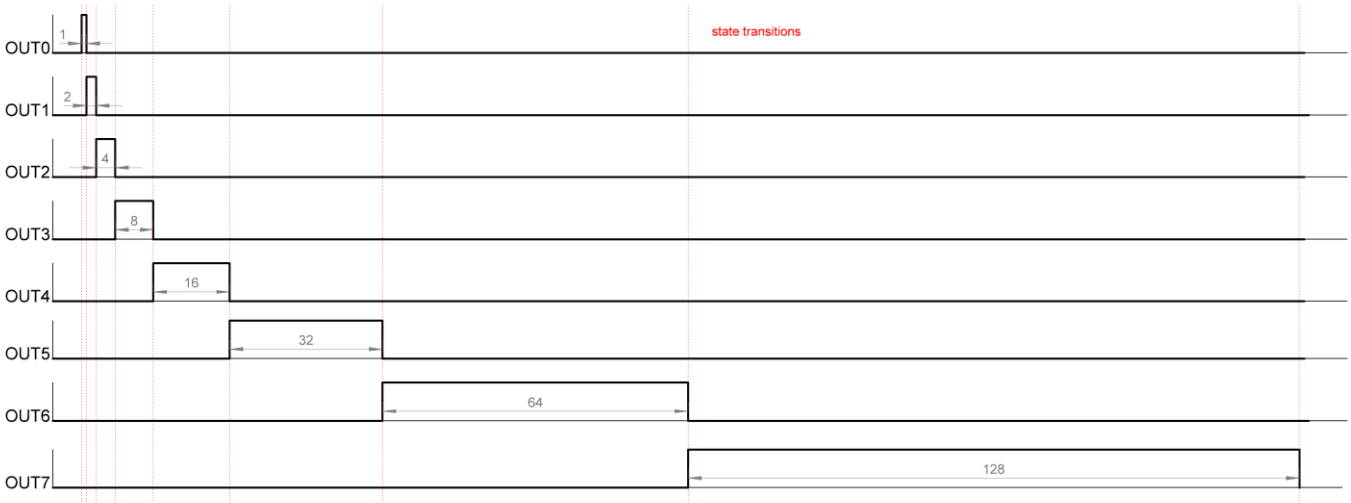
Table 2. Comparison

	ASM Outputs								
	OUT7	OUT6	OUT5	OUT4	OUT3	OUT2	OUT1	OUT0	Clock divider
State 0	0	0	0	0	0	0	0	1	2^0
State 1	0	0	0	0	0	0	1	0	2^1
State 2	0	0	0	0	0	1	0	0	2^2
State 3	0	0	0	0	1	0	0	0	2^3
State 4	0	0	0	1	0	0	0	0	2^4
State 5	0	0	1	0	0	0	0	0	2^5
State 6	0	1	0	0	0	0	0	0	2^6
State 7	1	0	0	0	0	0	0	0	2^7

Figure 7. ASM Outputs and the Output configuration

GreenPAK Design, the ASM outputs

The ASM Outputs are the PCM codes connected to GPIOs 20 through 13.



Timing Diagram 3. ASM Output Timing for the output configuration in Figure 6

The output table in Figure 7 is configured such that each ASM output represents one ASM state. This is to show the relative timing of each state as seen in Timing Diagram 3. OUT0 is first and shortest because it represents State0-bit0. OUT7 is last and longest because it represents State7-bit7. Each output has no overlap with its neighboring outputs and each is twice as long as the previous one.

Resolution

The total number of unique PCM codes is 256, from 0 to 255. For this design, the period is divided into 255 segments. Typically, a GreenPAK's PWM block outputs a period with 256 segments.

The last segment being forced high or low, such that the output duty cycle ranges from 0-99.6% or 0.4-100%.

In order to implement a 256th segment, we require a ninth state. GreenPAK5's ASM, however, has only eight states.

To avoid the reset, we forced the period to 255 segments. This causes the duty cycle to range from 0 to 100% and the resolution is 0.39215% (1/255) instead of 0.390625% (1/256). This logic is already integrated into the PGEN which has a 15 bit pattern. The PGEN will clock through at 15 fast clocks, and then 15 slow clocks. Each slow clock is 16 times slower than the fast clock. Therefore, the total number of segments is: 16*15 + 15 = 255.

14 total modulated outputs

In addition to the eight outputs from the ASM, we can utilize the leftover counters as more modulated outputs. Refer to Figure 8.

Since PCM and PWM are interchangeable for time-averaging voltage applications, the remaining counters can be used to create PWM outputs. One counter creates the period and the rest of them are one-shots shorter than the period. 8 ASM outputs + 6 CNT outputs makes 14 total PWM/PCM outputs.

I2C

The MCU can utilize I2C to change the PCM value, updating only when needed.

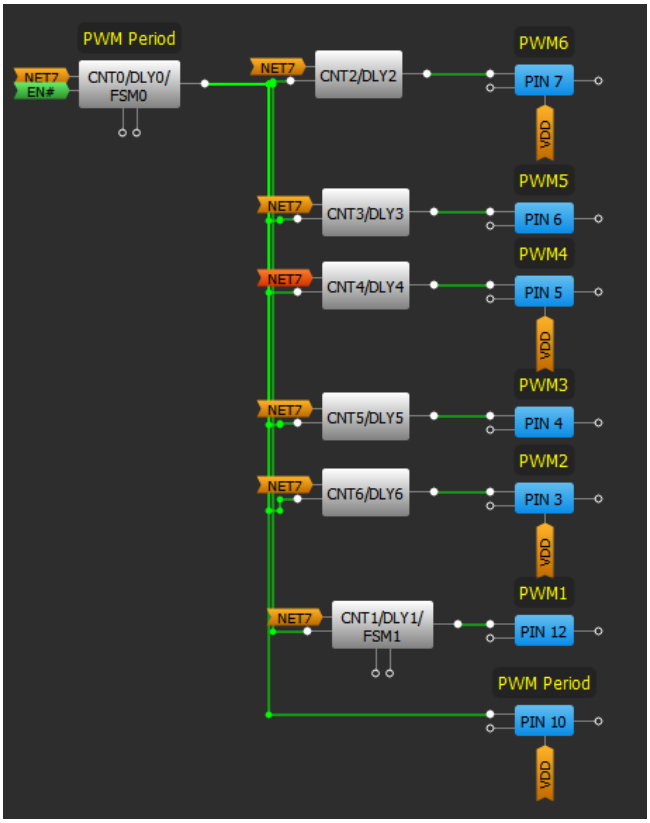


Figure 8. Additional Counter Outputs

The ASM output configuration is located at word addresses (0x0h) D0, D1, D2, D3, D4, D5, D6 and D7. View table 2.

In order to change one ASM output, we must re-write all 8 registers. For example, in Table 2, OUT2 is changed to 0x42. This requires us to write 8 bytes of data, D0 through D7 where the 3rd least significant bit is changed in 'red' below. The command would be the following:

[0xSA, 0xD0, 0x22, 0xE7, 0xB0, 0xE4, 0xB0, 0xE4, 0x70, 0xE8]

Where SA is the slave address.

At 400kHz, the maximum speed that an MCU can change the code of an output is 225µs (4.4kHz) not including the start and stop bit.

This gives plenty of margin since the naked eye cannot decipher changes faster than 24Hz.

During I2C write, the output will not glitch as the data is being updated. This is because ASM state outputs are always latched and don't change until the states transition.

There could be, however, a period that is neither one PCM code nor the other but a mix of the two due to data changing in the middle of a period.

Other examples commands are:

[0xSA, 0xD0, 0xAA 0xCC 0xF0 0x00 0x00 0x00 0x00 0x00] outputs 7,6,5,4,3,2,1,0

[0xSA, 0xD0, 0x01 0x02 0x04 0x08 0x10 0x20 0x40 0x80] outputs the identity matrix from Figure 6

Word Address	7	6	5	4	3	2	1	0	**Data Byte**
D0	0	0	1	0	0	0	1	0	0x22
D1	1	1	1	0	0	1	1	1	0xE7
D2	1	0	1	1	0	0	0	0	0xB0
D3	1	1	1	0	0	1	0	0	0xE4
D4	1	0	1	1	0	0	0	0	0xB0
D5	1	1	1	0	0	1	0	0	0xE4
D6	0	1	1	1	0	0	0	0	0x70
D7	1	1	1	0	1	0	0	0	0xE8
Decimal	**190**	**234**	**255**	**84**	**128**	**42**	**3**	**2**	

Table 3. I2C Word Address and Data Byte Example

Functionality Waveforms

D0 – PIN#20 (OUT0)
D1 – PIN#19 (OUT1)
D2 – PIN#18 (OUT2)
D3 – PIN#17 (OUT3)
D4 – PIN#16 (OUT4)
D5 – PIN#15 (OUT5)
D6 – PIN#14 (OUT6)
D7 – PIN#13 (OUT7)
D8 – (RED)
D9 – (FED)
D10 – (PGEN out)
D11 – PIN#8 (SCL)
D12 – PIN#9 (SDA)

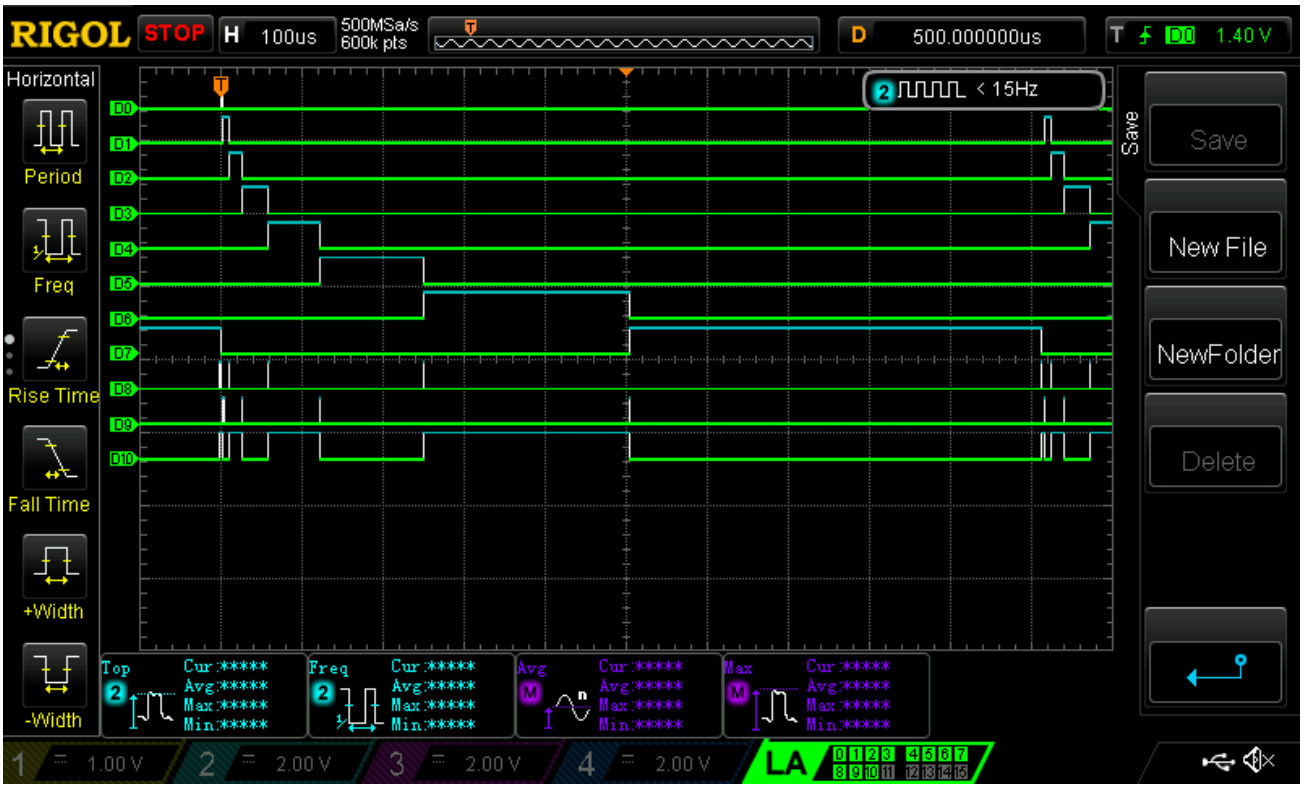


Figure 9. Identity Matrix

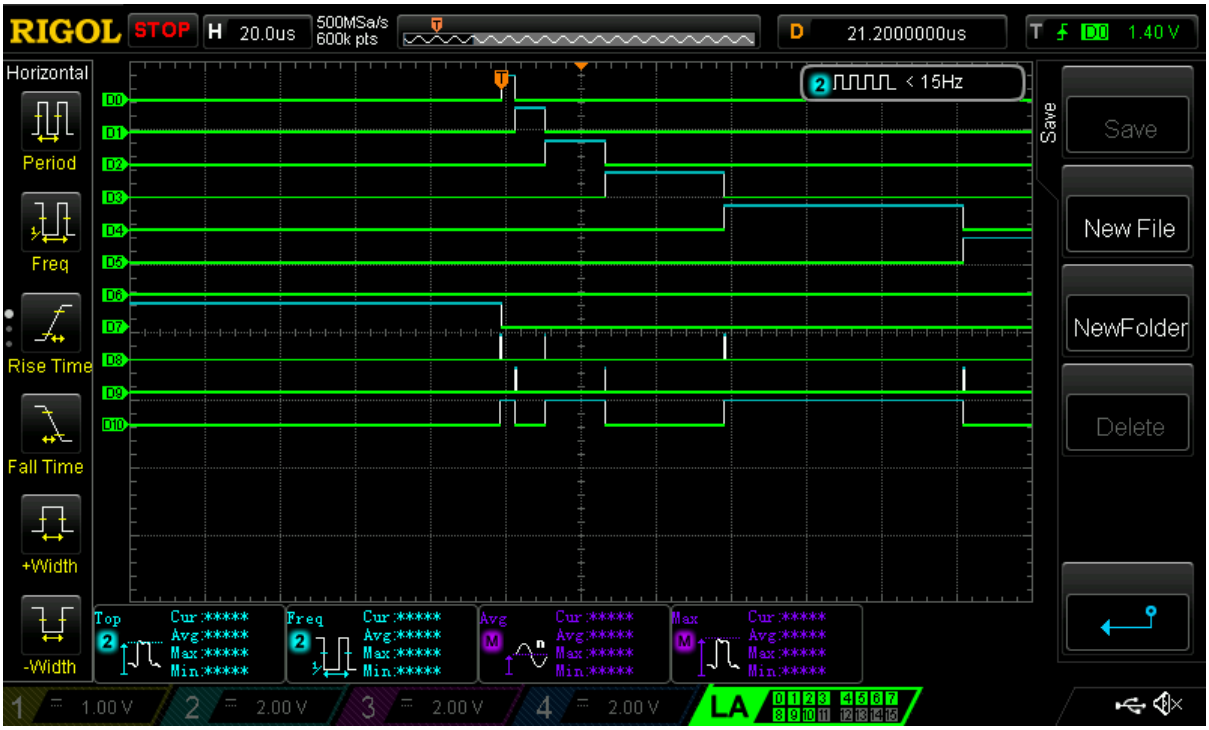


Figure 10. Zoom In (20us/div)

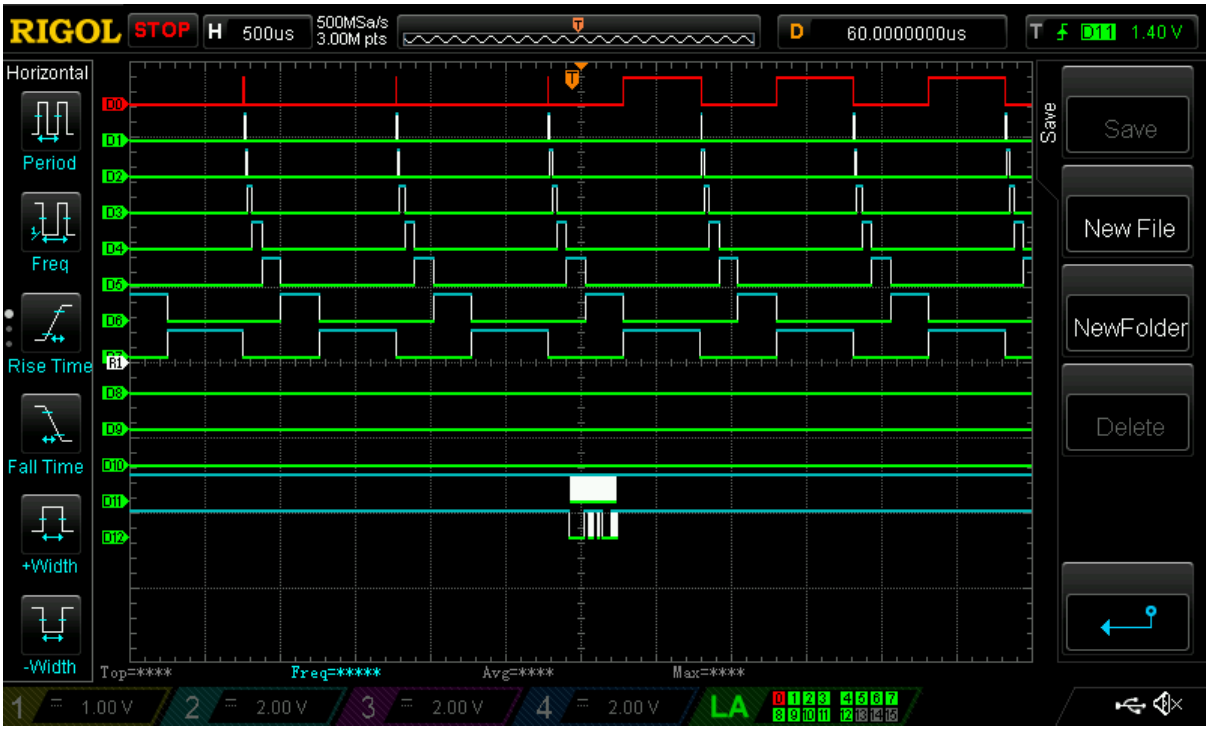


Figure 11. I2C write at 0xD7

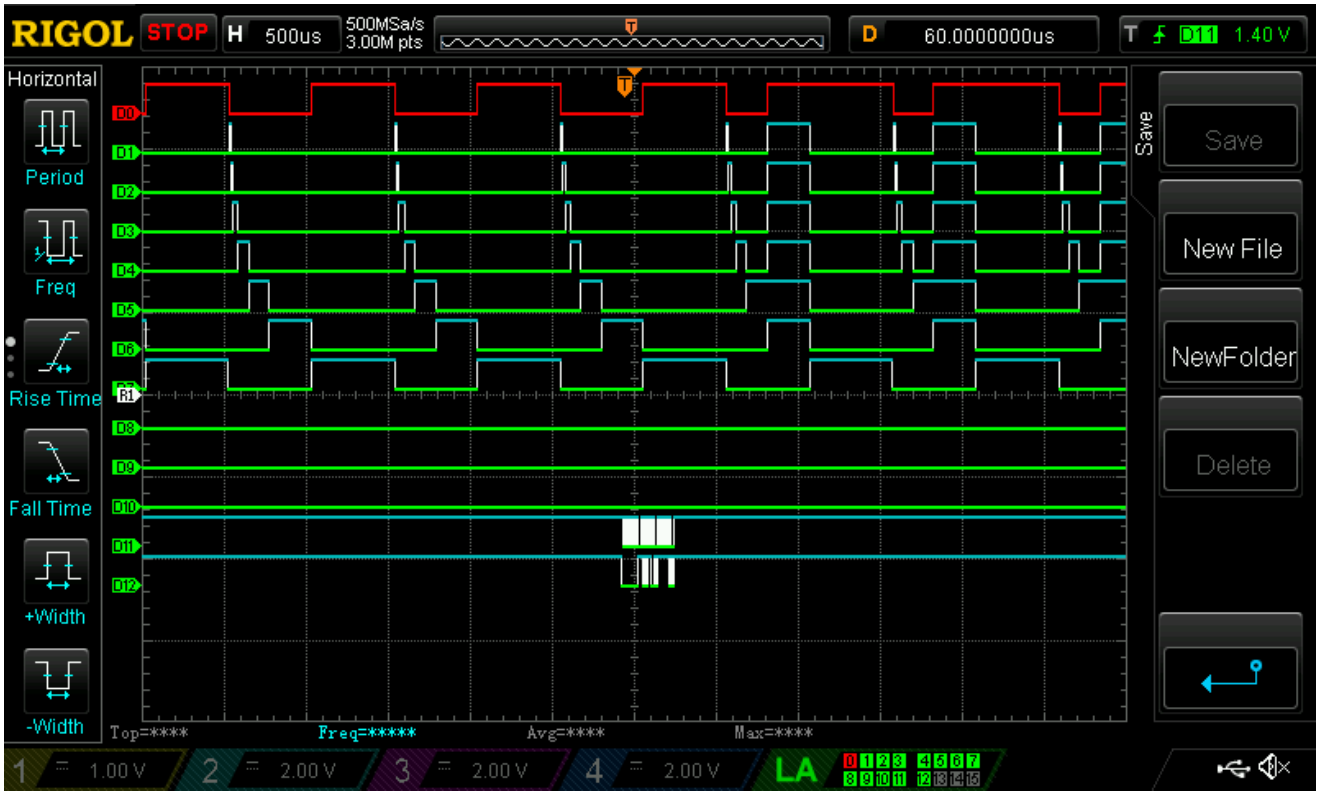


Figure 12. I2C write at 0xD6

Conclusion

The SLG46531V ASM can be configured to implement eight channels of PCM outputs each with 8-bits of resolution. All outputs are synced to the ASM period and the output code can be rewritten with I2C. This allows GreenPAK5 to run independently of the MCU, which can wake up once in a while to update the PCM code, saving power and GPIOs.

With the multiplexed clock and pattern generator, 7 components were used to make the ASM inputs, leaving 12 other components to spare.

Configuring the combinational macrocells in counter/delay mode allows up to 6 PWM output signals for a total of 14 pulse modulated outputs. In a 20 pin GreenPAK5, we can efficiently drive 14 outputs, receive information from 2 I2C inputs and still have 2 GPIOs which can be used for enables or interrupts between the MCU and GreenPAK. From this app note, you should be able to make your own basic resource-efficient PCM outputs using a GreenPAK5 and the ASM.

IMPORTANT NOTICE AND DISCLAIMER

RENESAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES (“RENESAS”) PROVIDES TECHNICAL SPECIFICATIONS AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF THIRD-PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for developers who are designing with Renesas products. You are solely responsible for (1) selecting the appropriate products for your application, (2) designing, validating, and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. Renesas grants you permission to use these resources only to develop an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third-party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising from your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use of any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Disclaimer Rev.1.01)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit www.renesas.com/contact-us/.

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.