Serial Output Tips & Techniques
SLG46537

The application note demonstrates serial output tips.

The application note comes complete with design files which can be found in the Reference section.

Contents

Figures

Tables

1. Terms and Definitions

ASM	Asynchronous State Machine
CLK	Clock
DFF	D Flip-Flop
LUT	Look-up Table

2. References

For related documents and software, please visit:

GreenPAK Programmable Mixed-Signal Products | Renesas

Download our free Go Configure Software [1] to open the .gp files [2] and view the proposed circuit design. Use the GreenPAK development tools [3] to freeze the design into your own customized IC in a matter of minutes. Renesas provides a complete library of application notes [4] featuring design examples as well as explanations of features and blocks within the Renesas IC.

[1] Go Configure Software, Software Download and User Guide, Renesas Electronics

[2] AN-1137 Serial Output Tips & Techniques.gp, GreenPAK Design File, Renesas Electronics

[3] GreenPAK Development Tools, GreenPAK Development Tools Webpage, Renesas Electronics

[4] Application Notes, GreenPAK Application Notes Webpage, Renesas Electronics

[5] SLG46537 Datasheet, Renesas Electronics

[6] SLG46620 Datasheet, Renesas Electronics

3. Introduction

This app note demonstrates various ways to output data serially from the GreenPAK devices. We will cover GreenPAK4 and GreenPAK5 devices but some of these techniques can apply to older PAKs as well. First, serial data is a pattern of logical 1s and 0s used to communicate with another device, such as an MCU, that will interpret the 1s and 0s for its own use. The data could represent device ID, serialized parallel data, or an arbitrary pattern. This app note explains the six basic serial output techniques using GreenPAK components from the simple pattern generator to the more complex 8-state state machine.

To follow along, download the GP files Serial_Output_Techniques_PAK4, Serial_Output_Techniques_PAK5_Horizontal and Serial_Output_Techniques_PAK5_Vertical. The first file contains general techniques and the latter two contain state machine examples.

4. PGEN

The simplest to understand, the PGEN is already designed to be a pattern generator. Each rising edge of CLK shifts out the next bit in the pattern and wraps around.

- Make matrix connections as shown in Figure 9.
- Configure Pattern with the desired serial data. (max. 16-bis)

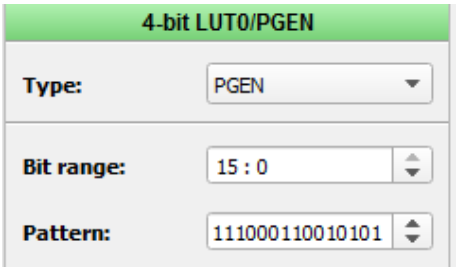


Figure 1. PGEN Properties

5. FSM and SPI

This example is written in another app note, AN- 1083. FSM counter data is converted to serial data via the SPI block. The output pin is preset to PIN12 for SLG46620V. To start-up the SPI, the nCSB input needs a high to low transition. For a detailed explanation, read AN-1083 SPI Parallel to Serial Converter.

- Make matrix connections as shown in Figure 9.
- Set SPI to 'P2S' mode
- Set Byte Selection to [15:0] or [7:0]
- Set PAR input data source to 'FSM0[7:0] FSM1[7:0]'
- Configure *Counter Data* with the desired pattern.

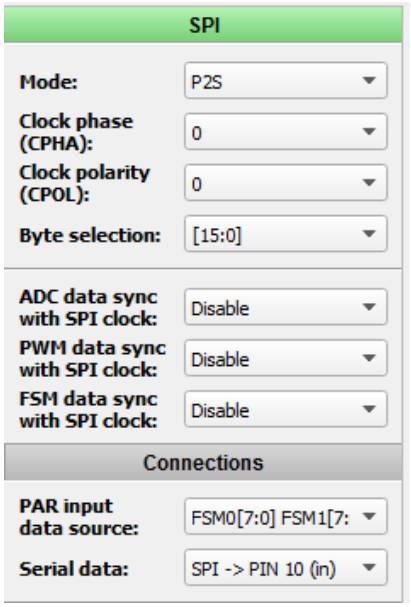


Figure 2. SPI Properties

6. DFF chain

A pattern is stored in D Flip Flops where each DFF stores one bit. By connecting DFFs in a register string, data is clocked out serially. The number of DFFs is linearly proportional to the number of bits in the pattern.

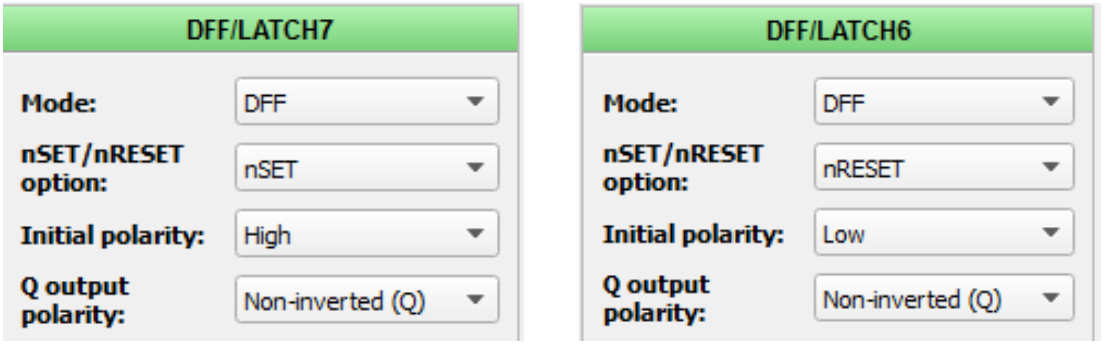
- Make matrix connections as shown in Figure 8.
- Connect the nSET/nRESET input to POR where applicable.

Configure the *initial polarity and nSET/nRESET* of each DFF with the corresponding bit in the data. Follow Figures 3a, 3b, 4a, and 4b.

7. Binary code DFFs

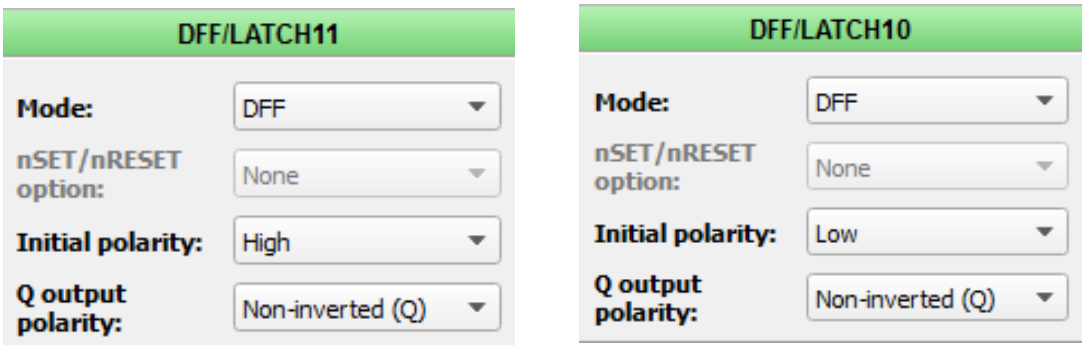
We covered how to make patterns by shifting data out one at a time either through the PGEN, the SPI or DFFs. Another way is to generate a pattern of states and set the output for each state. This is called a state machine and the pattern is dependent on the order that the states appear.

The GreenPAK 5 ASM is easy to configure, but for all other non-ASM devices, we can use binary or gray code to select the order of states. For binary code, the states are {000, 001, 010, 011, 100, 101, 110, 111} in that order. We will need three DFFs, one for each bit and a 3-bit LUT to configure the output pattern.



a) b)

Figure 3. Set Initial High and Low for DFFs with nSet/nReset option



a) b)

Figure 4. Voltmeter with 3-Digit 7-Segment Indicator Circuit Schematic

The number of DFFs used is exponentially proportional to the number of bits in the pattern (power of 2). The only caveat is the propagation glitches between state transitions. This happens because transitions are level sensitive and binary code could change multiple bits simultaneously. To avoid glitches, the output should be synced to the falling edge of clock, or filtered.

- Make matrix connections as shown in Figure 9.
- Connect the nSET/nRESET input to POR where applicable.
- Configure the *initial polarity* to High, and *Q output polarity* to Inverted(nQ)
- Configure the *OUT* of the Look-Up-Table with the desired pattern in the order of binary states from 000... to…111.

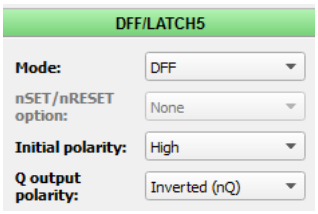
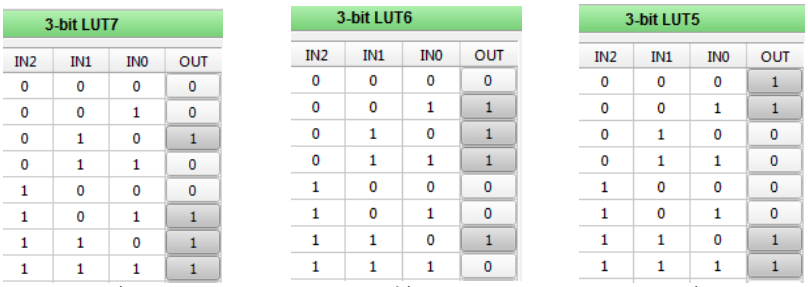


Figure 5. Binary code DFFs



3-bit LUT7

IN2	IN1	IN0	OUT
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

a)

3-bit LUT6

IN2	IN1	IN0	OUT
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

b)

3-bit LUT5

IN2	IN1	IN0	OUT
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

c)

Figure 6. DFF Properties

8. Gray code DFFs

Similar to the previous example, this one uses gray code as the state transitioning order {000, 001, 011, 010, 110, 111, 101, 100}, which avoids transitional glitches. We will need three sets of 3- bit LUT and DFF, one for each bit and one 3-bit LUT to configure the output pattern. The number of DFFs and LUTs used is exponentially proportional to the number of bits in the pattern (power of 2). Patterns requiring more than 8 states are highly inefficient in gray code because it uses too many circuit resources.

- Make matrix connections as shown in Figure 8.
- Connect the nSET/nRESET input to POR where applicable.
- Configure the *OUT* of the 3-bit Look-Up- Table with the desired pattern in the order of gray code states from 000... to…100.

Table 1. Gray Code Order

Gray Code		
Bit2	**Bit1**	**Bit0**
0	0	0
0	0	1
0	1	1
0	1	0
1	1	0
1	1	1
1	0	1
1	0	0

9. Pipe Delay

The Pipe Delay is a bunch of DFFs with three outputs. We can use it as a shift register and toggle the input such that we cycle through the states {00, 01, 11 and 10}. A 3-bit shift register states would look like {000, 001, 011, 111, 110, 100}. Using a shift register for the state machine also avoids transitional glitches like gray code, but the number of bits needed is linearly proportional to the number of bits in the pattern (2x).

- Make matrix connections as shown in Figure 9.

- Configure the *OUT* of the 2-bit Look-Up- Table with the desired pattern in the order of shift register states 00… to… 10.

10. GreenPAK5 – 64-bit Horizontal ASM

In the next two designs we will look at using the ASM to hold the output pattern. ‘Horizontal’ refers to storing the pattern by rows and ‘Vertical’ refers to storing the pattern by columns. The rows are the states and the columns are the outputs. A horizontal design scrolls through the outputs of one row and then moves on to the next row.

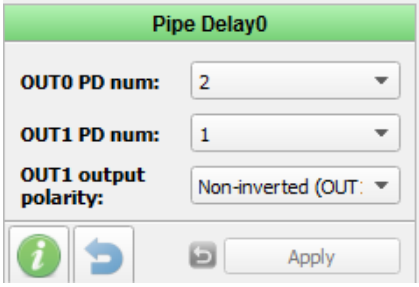


Figure 7. Pipe Delay Properties

In Figure 7, each ASM output is windowed for 1 period by the Pipe Delay, configured as in the previous section. The PGEN generates a pulse every 8 clocks which is used to transition between states.

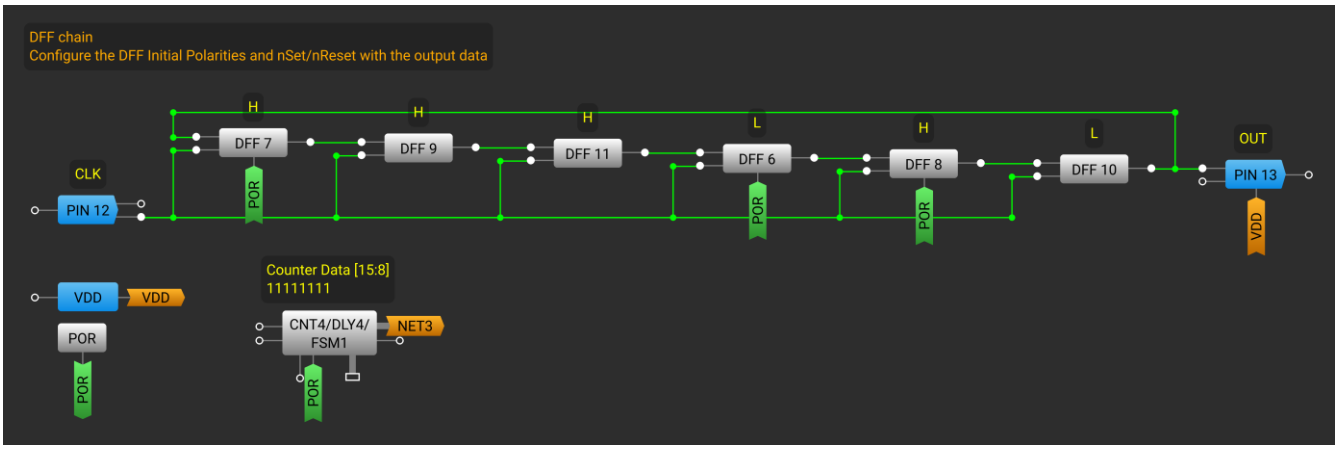


Figure 8. Matrix 1

11. GreenPAK5 – 64-bit Vertical ASM

In the ‘Vertical’ ASM, the design scrolls through the states of one column and then moves on to the next column. In Figure 9, each ASM output is windowed for 8 clock periods by the Gray Code DFFs. The DFF3 generates a pulse every period which is used to transition between states.

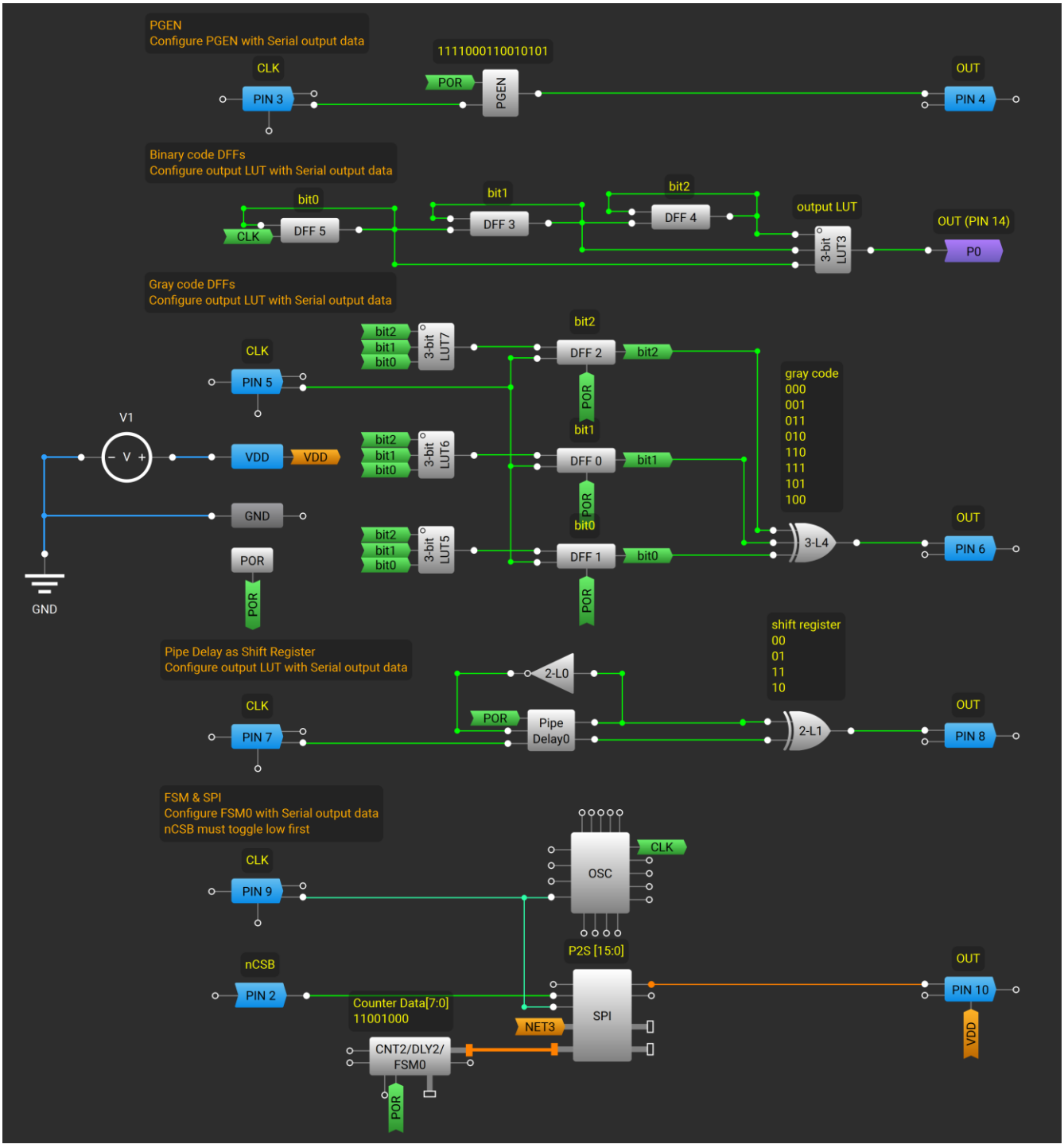


Figure 9. Matrix 0

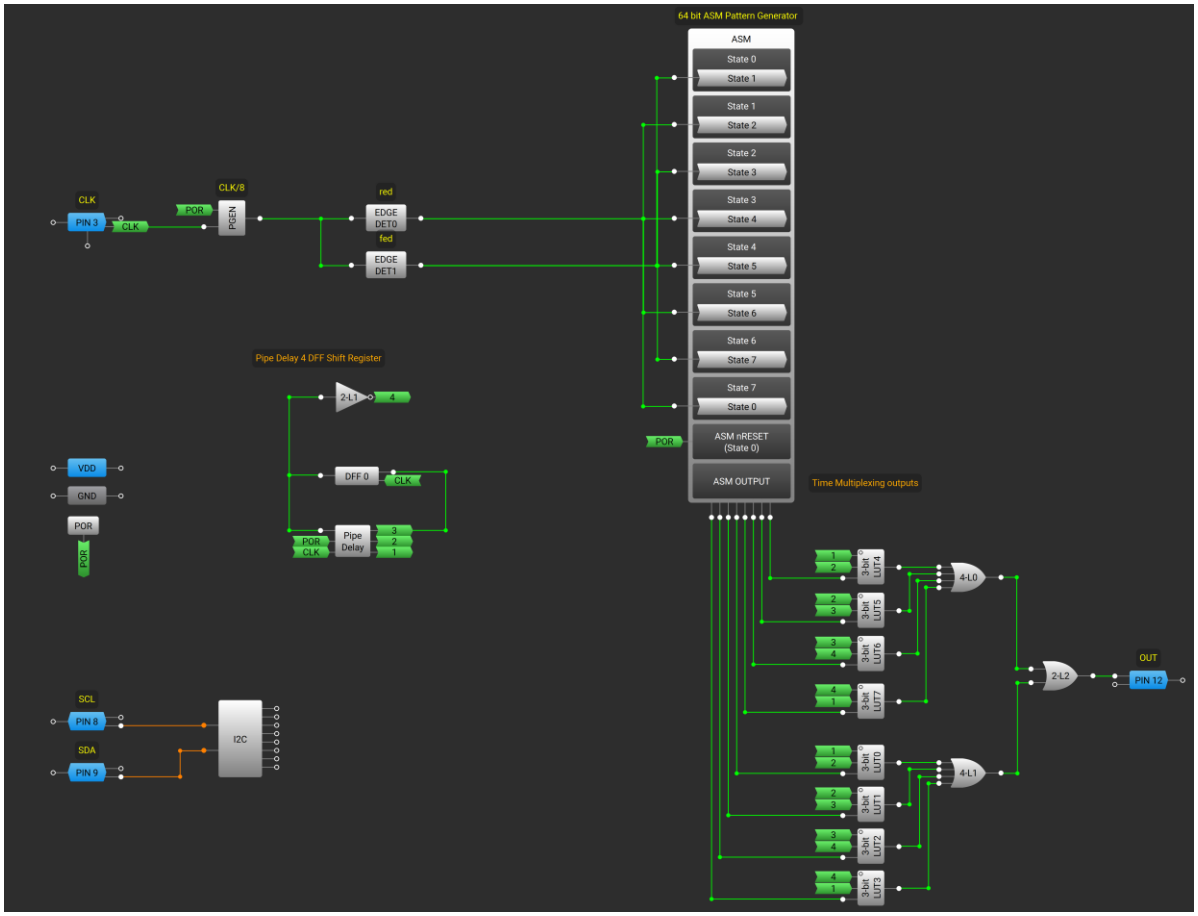


Figure 10. GreenPAK5 ASM Horizontal Pattern generator

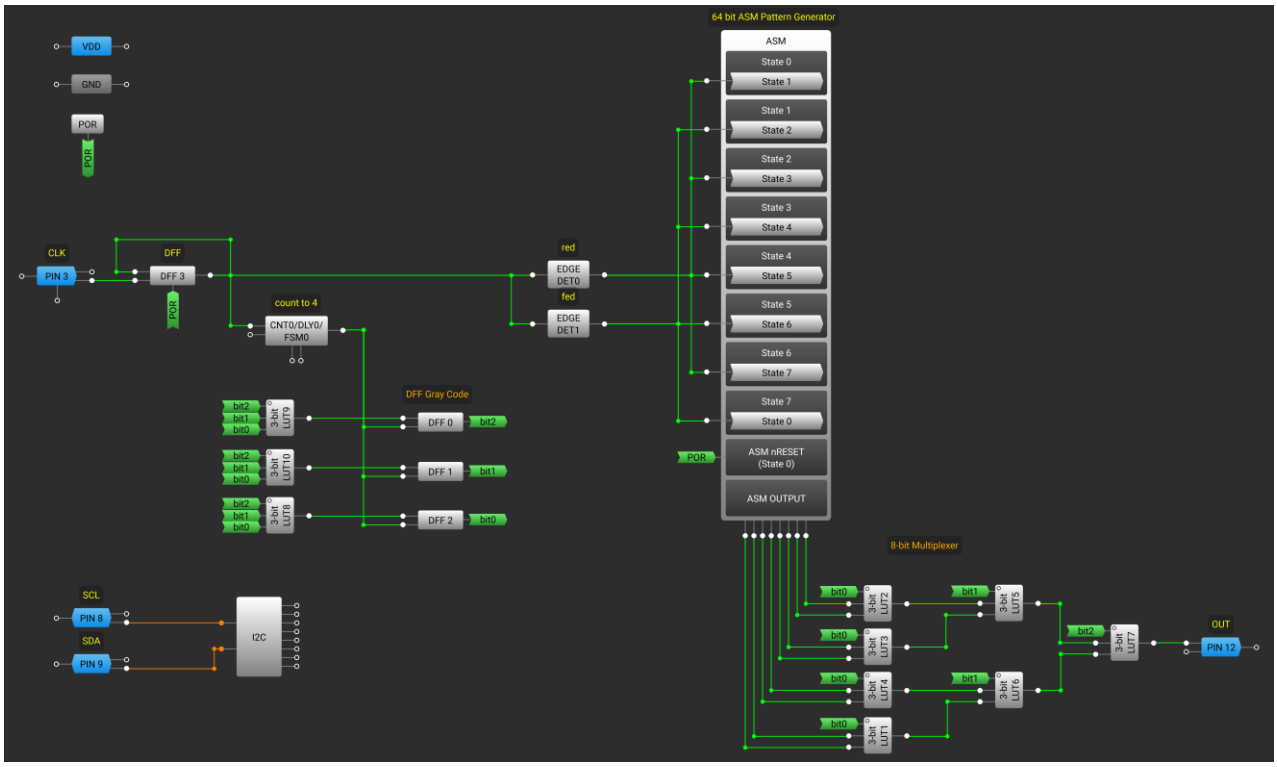


Figure 11. GreenPAK5 ASM Vertical Pattern generator

12. Conclusion

There are numerous ways to generate a serial pattern in GreenPAK. The basic forms above provide a starting point and each of them have their pros and cons. While the PGEN is simplest and produces 16 bits, it is available only in GreenPAK4 and GreenPAK5. The FSM and SPI buffer is a great alternative but only available in GreenPAK4, limited to one or two byte length and requires an extra input. If the pattern is short, simply use DFFs in a ring structure and set an initialized value.

There are also state machine alternatives that use DFF's and LUT's to create a code that can help you generate a pattern. In this app note, we covered using binary and gray code DFF's or the register shifting Pipe Delay to traverse states in a particular order. Binary and Gray code yields 2^x pattern bits and the Pipe delay yields 2*x pattern bits where x is the number of DFF's used. Both gray code and the pipe delay eliminate output glitches that could occur from using binary counting DFF's.

Select the form that best suits the circuit resources available and the length of data needed to output. Just about any block can store data and there are more than a few ways to send that data out as this app note only covers the simplest of forms. Manipulate or expand by adding more DFF's, optimize for long run lengths of 1s or 0s, or even re-use components to maximize pattern storage.

13. Revision History

Revision	Date	Description
1.00	September 19, 2016	Initial release.
2.00	April 28, 2026	The part number has been changed from SLG46531V to SLG46537V.

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.