

DS1180A Glitch Pattern Generator

Communication and fault injection of embedded chips



Context

The market

The market for high-security embedded chips has grown by leaps and bounds in the past five years, particularly in content protection and mobile devices. Compared to smartcards, which are fairly standardized, embedded chips pose a significant challenge in side channel and fault injection testing due to their large variety. Testers need a tool to address this challenge without increasing the complexity of their setups. Adding flexibility in triggering, control, and on-the-fly adjustments of fault injection campaigns greatly expands the testing surface. Moreover, combining protocols like joint test action group (JTAG), serial peripheral interface (SPI), inter-integrated circuit (I2C), or controller area network (CAN) with fault injection opens up new avenues of testing.

Our approach

We developed the DS1180A Glitch Pattern Generator, a highly versatile field programmable gate array or FPGA-based tool to:

- Reduce setup complexity for embedded device testing.
- Generate faults for power amplifier, laser, or electromagnetic fault injection (EM-FI) equipment.
- Enable on-the-fly adjustments of fault injection parameters.
- Communicate at a low level with embedded chips.
- Make it easy to develop proprietary interface and protocol extensions.

Other methods	Limitations
Smartcard Voltage and Clock Glitcher	<ul style="list-style-type: none">• Supports only smartcard protocol• Limited protocol flexibility
Custom FPGA design	<ul style="list-style-type: none">• Steep learning curve• R&D investment for a good digital and analog combination
Microcontrollers	<ul style="list-style-type: none">• Cannot provide rigid triggering• Cannot provide true parallelism

Challenges the DS1180A Glitch Pattern Generator Solves

1. Time-consuming setup process

Setting up a side channel test environment can be time-consuming. This is because embedded targets have a large variety of communication protocols, multiple power domains, and different I/O voltage levels.

As a tester, you want control over your target's interfaces so that you can effectively run your desired tests. Use the Glitch Pattern Generator to sniff and communicate with an embedded chip, making use of common chip-to-chip protocols.

2. Inadequate trigger performance

Based on what is observed on a chip's data bus, you may want to start a power measurement or inject a fault. The Glitch Pattern Generator can generate highly accurate triggers to accomplish this task. Because it supports JTAG, I2C, and SPI, the Glitch Pattern Generator offers great flexibility, allowing you to use it with a wide range of embedded chips.

3. Comprehensive attack flow coverage

To create effective security tests that leave no vulnerabilities undetected, testers must be able to easily change and customize a fault attack's flow. Create custom attack flows with ease, using Inspector software or the Glitch Pattern Generator's software development kit (SDK) library. Multiple options are available, including options to influence voltage level, change event ordering, and adjust glitch timing. With the Glitch Pattern Generator, testers can tailor the shape of a glitch without any user intervention.

Unique Features

1. All-in-one tool to control embedded targets

- Wide range of I/O voltage levels (1.0 – 3.3 V)
- Support for popular protocols
- Flexible trigger generation
- Control two lasers for simultaneous multi-pulse attacks

2. Versatile fault generator

- Drives faults to glitch amplifiers, lasers, EM-FI probes
- Program any attack flow
- Arbitrary waveform generation for faults

3. Flexible and easy-to-use software

- SDK in Python, Java, and C
- Plug and play from Inspector

4. Extensive protocol support

- SPI
- JTAG
- I2C
- CAN
- UART

Use Case: JTAG Unlocking

The test scenario

1. For security reasons, it is common practice to lock a JTAG interface.
2. The Glitch Pattern Generator can challenge the strength of a locked JTAG interface by controlling the reset line of the target.
3. Glitch during target booting:
 - a. Apply a normal voltage common collector (VCC) to the target for booting.
 - b. Lower the VCC to a minimum level just before an attack.
 - c. Generate a glitch via the 1.5 A Glitch Amplifier
4. Perform a standard device ID readout via JTAG communication.

Injecting faults into the chip in this test scenario enables testers to unlock a locked JTAG interface.

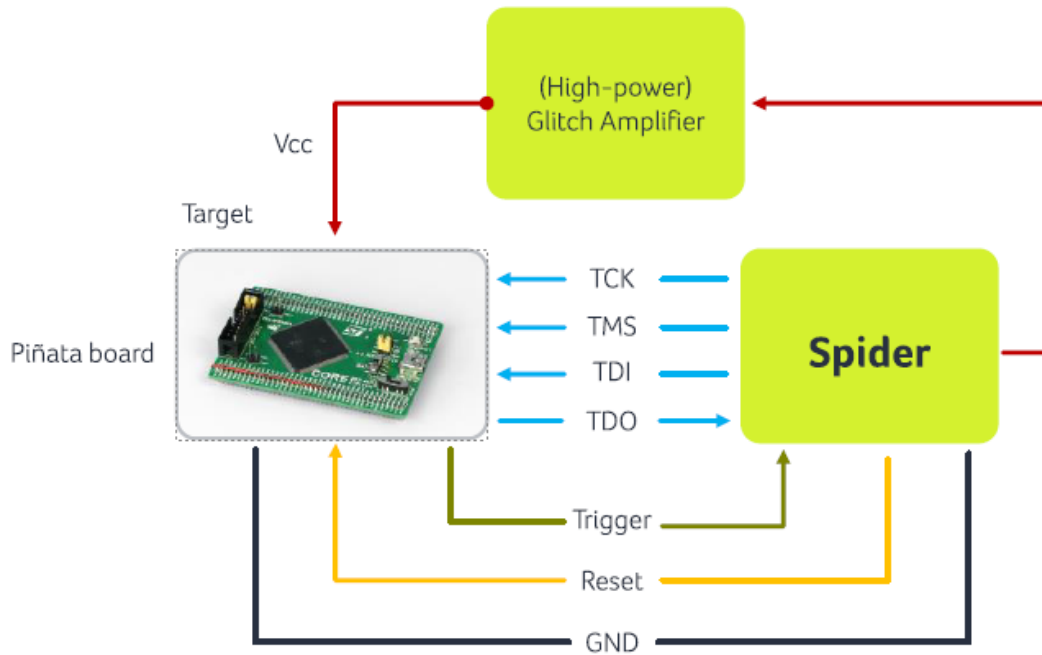


Figure 1. The DS1180A Glitch Pattern Generator, formerly called the Spider, manages resetting, glitching, and JTAG communication

Use Case: Drive Dual Laser Fault Injection

The test scenario

1. The Glitch Pattern Generator can drive the Dual Laser Fault Injection System's two lasers independently.
2. This allows for a simultaneous attack on the crypto-core and memory storage, getting results that would otherwise be impossible.

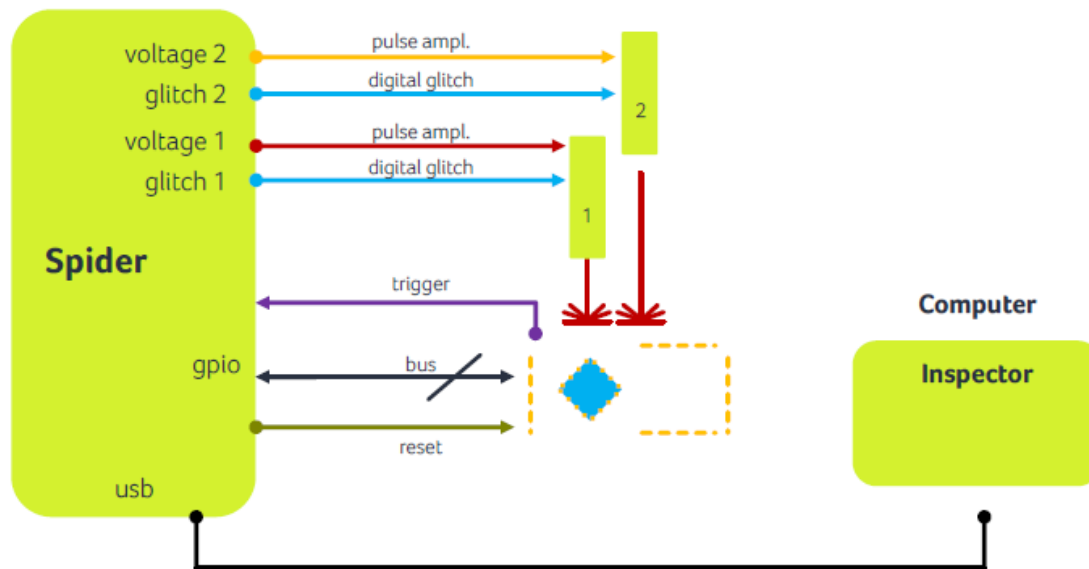
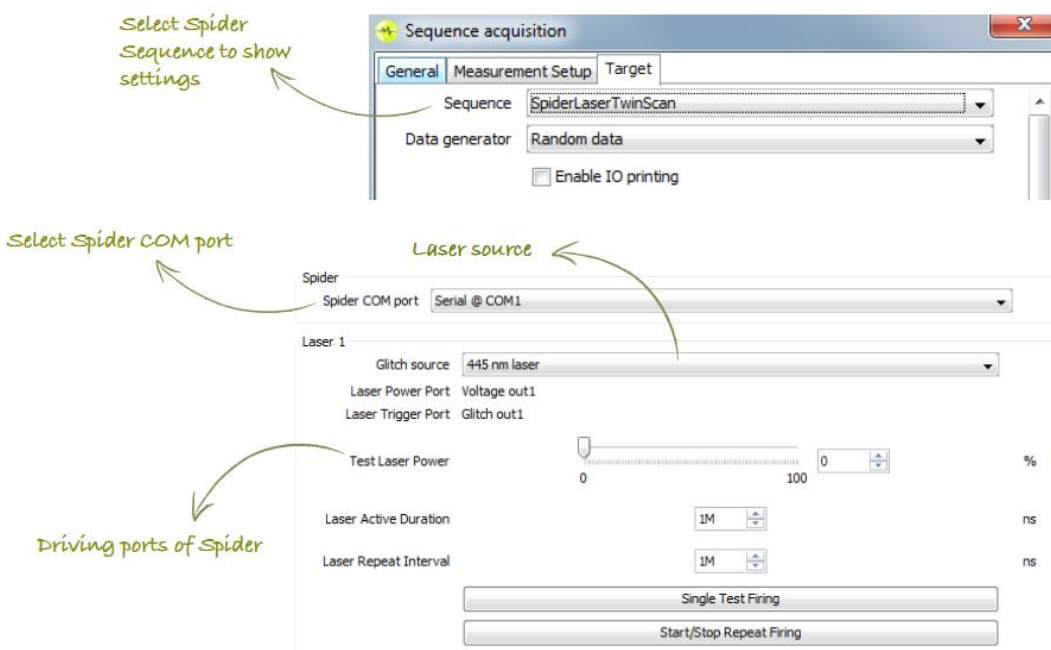


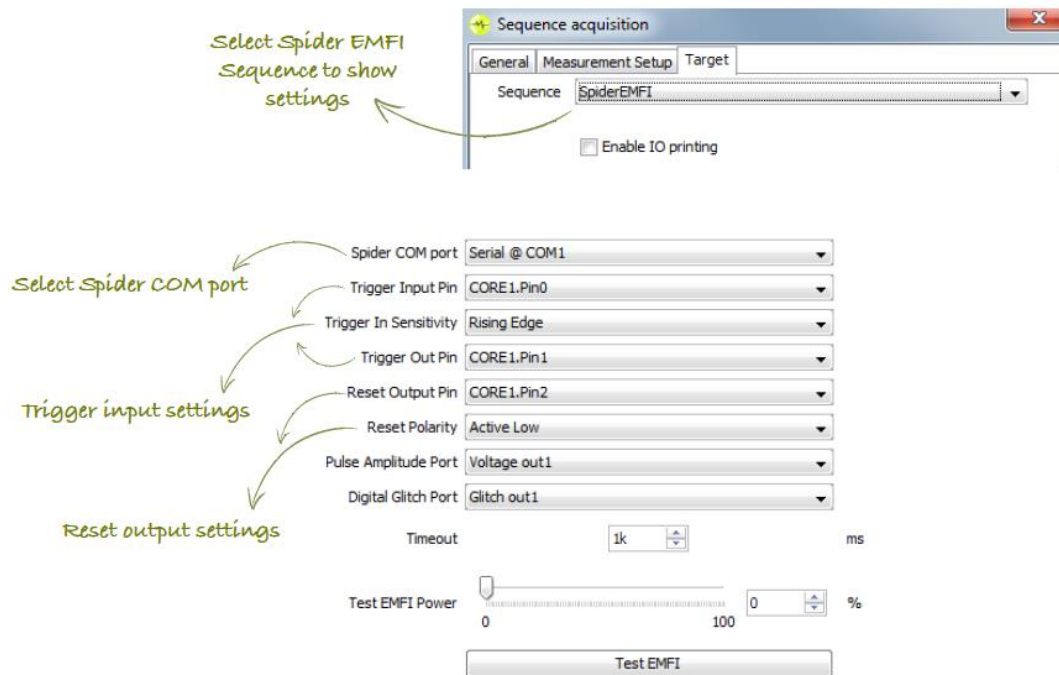
Figure 2. Typical use case featuring the Keysight Dual Laser Fault Injection solution

Technical Details

User control — EM-FI



User control — Dual Laser Fault Injection System



Programming — Inspector


```
spiderCore1 = new Spider(Spider.CORE1, connections.get(rawIOSpider));  
spiderCore1.resetSettings();  
glitcher = new Chronology(spiderCore1);
```

Figure 3. Create a glitcher using the DS1180A Glitch Pattern Generator Core 1

```
// Add new events  
glitcher.setVcc(selectedGlitchPort, normalVcc);  
glitcher.setGPIO(triggerOutIndex, 0);  
glitcher.waitTrigger(triggerInIndex, selectedTriggerSensitivity, 1);  
glitcher.setGPIO(triggerOutIndex, 1);  
glitcher.glitch(selectedGlitchPort, glitchVcc, glitchDelay, glitchDuration);
```


Figure 4. Add events and customize their order

Programming — Python






```
com_port = serial.Serial();  
com_port.port = "COM1";  
com_port.open();
```

Figure 5. Assign and open the COM port



```
spiderCore1 = Spider(Spider.CORE1, com_port);  
spiderCore1.resetSettings();  
glitcher = Chronology(spiderCore1)
```

Figure 6. Create a glitcher using Core 1










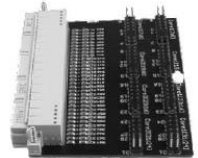

```
glitcher.waitTrigger(0, Spider.RISING_EDGE, 1)  
glitcher.setVcc(Spider.GLITCH_OUT1, 3.3)  
glitcher.waitTime(100e-6)  
glitcher.glitch(Spider.GLITCH_OUT1, -4.0, 0, 100e-9)  
glitcher.glitch(Spider.GLITCH_OUT1, -1.0, 100e-6, 500e-9)  
glitcher.waitTime(0.5)  
glitcher.setVcc(Spider.GLITCH_OUT1, 0)
```

Figure 7. Add events and customize their order

Technical Specifications

Parameter	Minimum	Typical	Maximum	Unit
gpio voltage level (V_{Logic})	1.0	-	3.3	V
gpio V_{OH}	$V_{Logic} - 0.45$	-	-	V
gpio V_{OL}	-	-	0.4	V
gpio V_{IH}	$0.65 V_{Logic}$	-	$V_{Logic} + 0.3$	V
gpio V_{IL}	-0.3	-	$0.35 V_{Logic}$	V
Voltage output	0.0	-	5.0	V
Voltage output current	-	-	100	mA
Glitch output voltage	-4.0	-	4.0	V
Glitch output current	-	-	72	mA
Glitch output timing resolution	-	4	-	nS
UART signal voltage level	-	3.3	-	V
UART baud rate	1907	-	1.5 M	baud

Package

Quantity	Description	Photo
1	DS1180A Glitch Pattern Generator	
1	15 V DC power supply unit, input 100 – 240 V, AC 50 – 60 Hz Included: power cable with country-specific jack	
1	Communication cable: USB-A – USB-B, 2 m	
4	Signal cable: SMB – SMB	
10	Jumper wires: Male – female	
10	Jumper cables: Female – female	
4	Output impedance adapter SMB, 50 Ohms	
1	Breakout board	
1	DS1180A Glitch Pattern Generator SDK USB stick	
1	User manual	

Keysight enables innovators to push the boundaries of engineering by quickly solving design, emulation, and test challenges to create the best product experiences. Start your innovation journey at www.keysight.com.