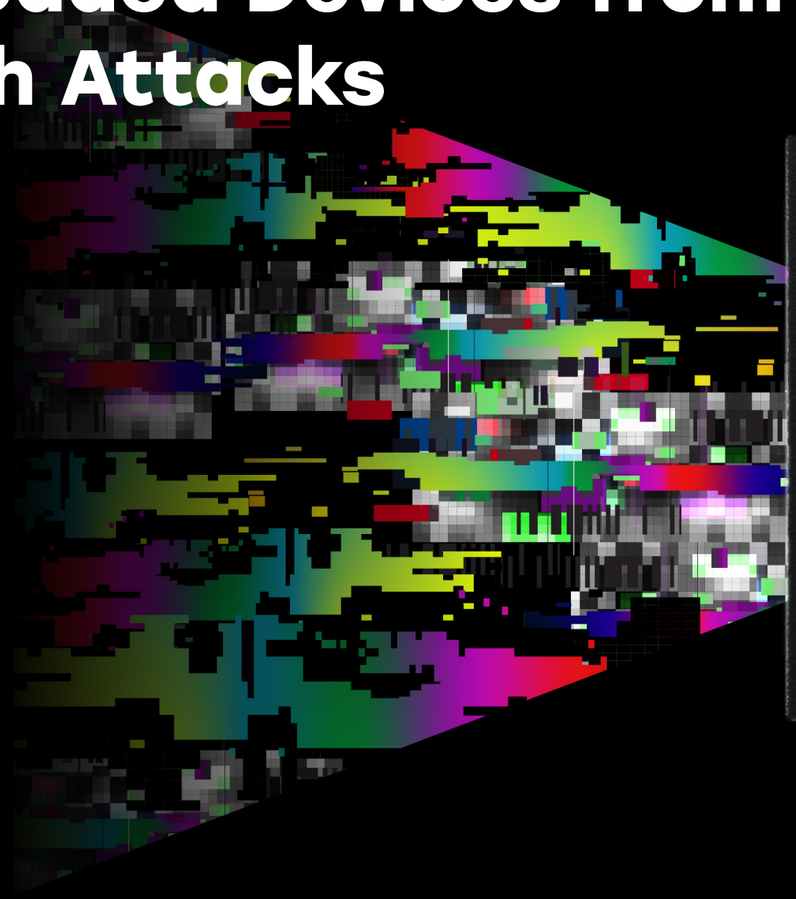
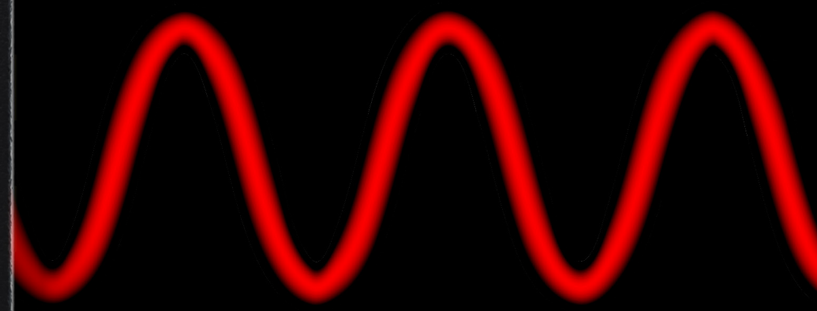
SILICON LABS
CONNECTED INTELLIGENCE

Silicon Labs Authenticated-XIP Protects Embedded Devices from Glitch Attacks





Embedded wireless devices now permeate every aspect of our lives, and the bad actors that once focused on desktop PCs for cyberattacks now have new targets: pacemakers, smart bulbs, and door locks are all fair game. Encrypting firmware stored in off-chip flash, common when microcontrollers execute in place (XIP), seems like the obvious defense. However, our testing shows that EMP-glitch attacks can corrupt the encrypted QSPI data reads and generate successful exploits in security-critical code.

Silicon Labs' **Authenticated-XIP (AXIP**) closes that loophole. Every read from external flash carries an authentication tag that the on-chip controller verifies before the data is used. A tag mismatch immediately aborts the transfer, giving firmware a clean hook for remediation or shutdown.

In internal fault-injection campaigns, nearly all successful glitches happened during the flash-read window. Enabling AXIP blocked the vast majority of exploits outright. The few remaining exploits were caught more than 90% of the time using our glitch detection technology, pushing real-world exploitation from "practical" to "improbable."

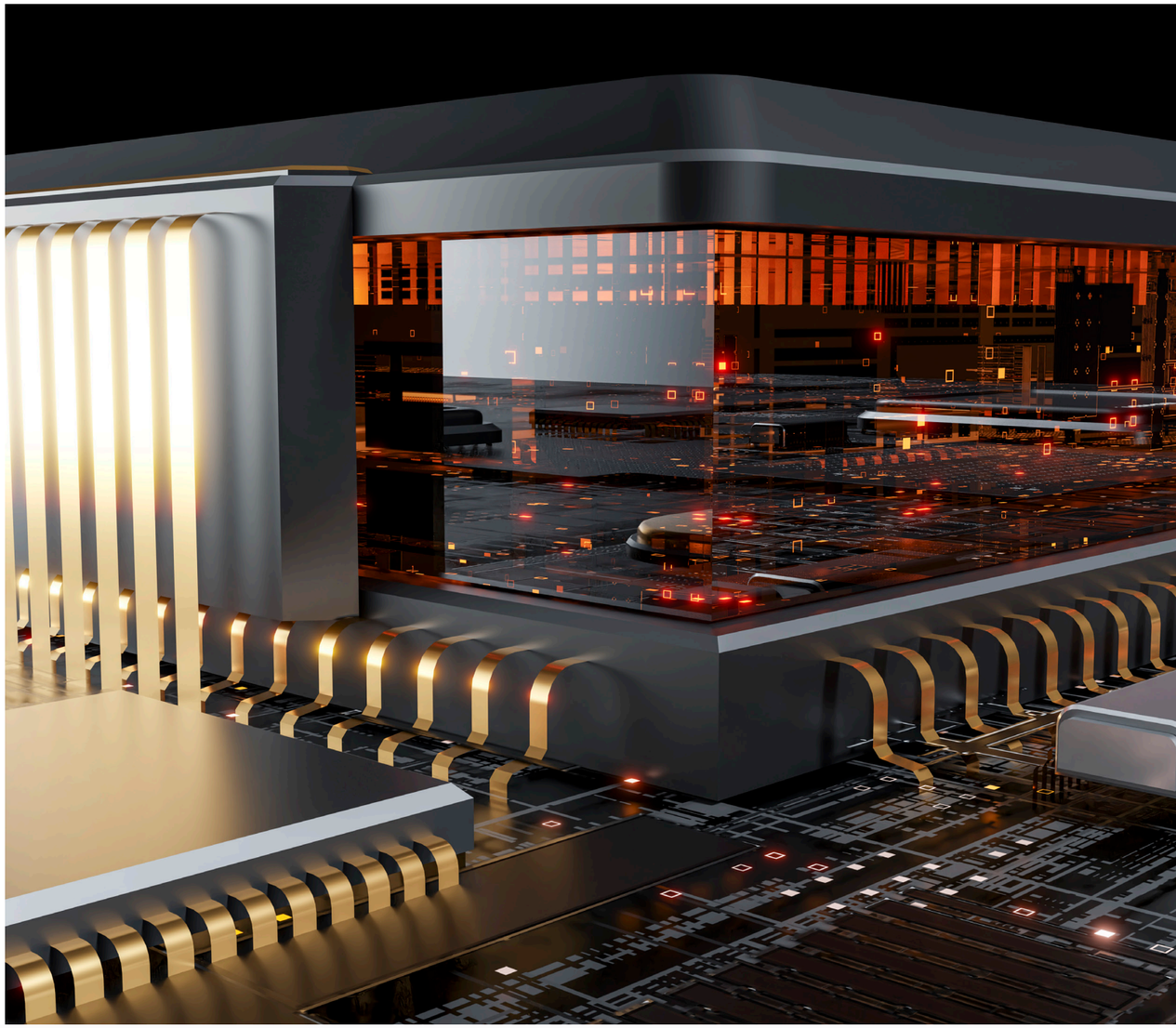
Why External Flash, and Encryption Alone Isn’t Enough

In the world of embedded systems, many of our chips are complete systems on a single die. This includes flash, RAM, CPU, and many diverse peripherals, including embedded radios. This increased complexity also increases code size, and system engineers have realized that it can be cheaper to offload some or all of our code storage to external flash chips. These external flash chips that use manufacturing techniques specific to the flash memory technology can provide a significant cost savings. A chip process that can support the radio, the CPU, as well as a variety of analog and digital peripherals either cannot support an on-die flash, or it can support an on-die nonvolatile storage at a significant cost premium.

Creating chips with off-die flash requires special consideration for the communications link. A fully unencrypted communications link, for example, means that the code is stored on this external flash in plain text. Anyone can pop the flash chip off the circuit and download the code. This will trivialize the process of reverse engineering or even cloning the device.

Everyone recognizes the need to protect this off-die storage with encryption, and there are plenty of technologies that facilitate this process. These include block cipher techniques that offer good protection with the ability to decrypt randomly accessed blocks of data and caching techniques that help to compensate for the significantly slower off-chip code storage access vs code storage on the die itself. There are also special one-time programmable memories that are cheap enough for all processes to store data that absolutely must be on the main die of the system.

However, encryption only protects against one type of attack. It can make the data secure against readout or even re-write. But it does not prevent an attacker from faulting the data in transit and forcing the system to execute corrupted firmware.



Threat Model: QSPI Read

As part of the architectural review for our latest Series 3 chip development, we considered a theoretical QSPI flash read attack. The idea is to run the QSPI bits through an XOR gate, allowing us to flip any combination of the bits using 4 control signals. We considered two cases: First, for a stream cipher-based encryption, where the AES output is used to XOR the plain text to encrypt/decrypt the data, flipping a single bit of ciphertext will yield a plain text change of a single bit. For a block cipher encryption, where the data itself is run through the AES encryption/decryption algorithms, a single bit flip of the ciphertext will yield, on average, a 50% change in the bits of the original plaintext.

The next question, then, is what's the probability that a random 16-bit word will yield a valid ARM opcode? We ran every value from 0x0000 to 0xFFFF through a disassembler. 1500 were marked as "invalid", which means that if we generate a random opcode (as would be the case in the stream cipher), it would be valid with a probability of P(valid) = 0.976. If we use a block cipher for external flash encryption, a single bit flip will yield 8 random op-codes, all of which will be valid with a probability of P(valid)^8=0.823. The conclusion is that you cannot rely on the potential to generate invalid op codes for protection against glitching attacks.

Next, we ran a theoretical attack against the sample password test code. The goal here was to give the software an invalid password. If, after initiating the glitch, the software reports the password as valid, then we have proven the exploit potential of the attack.

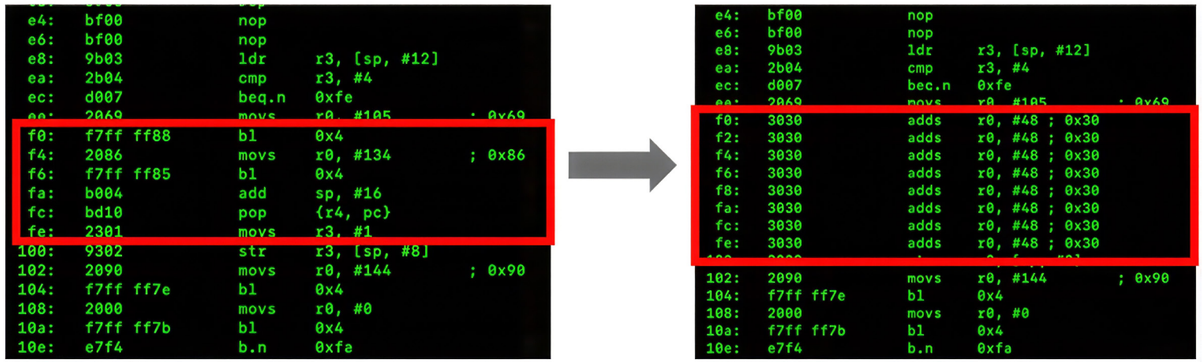
```
//INTERRUPTS_OFF();
GPIO_HIGH(); //Trigger on
//Small delay to have a bit of time between trigger to glitch
SMALL_DELAY();
for (i = 0;i < 4; i++) {
  if (binary_argument[i] == password[i]) {
    charsOK = charsOK + 1;
  }
}
GPIO_LOW(); //Trigger off

SMALL_DELAY();
SMALL_DELAY();
```

This code translates to the following assembly:

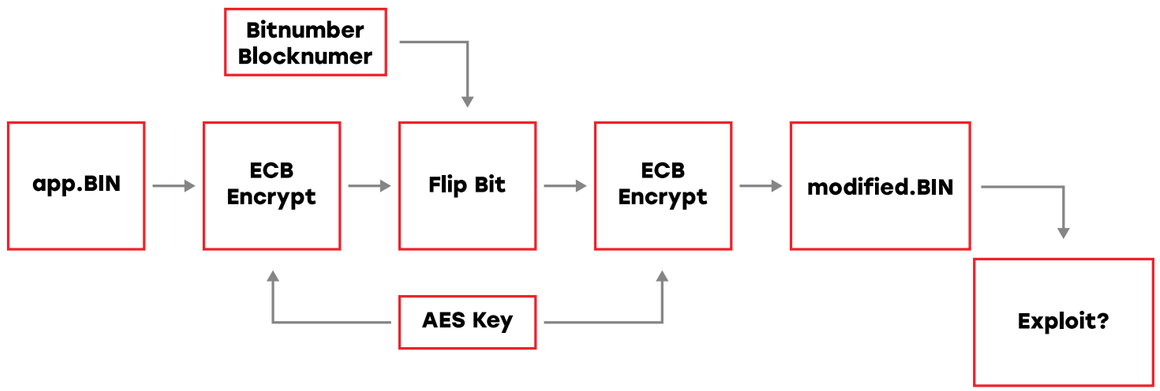
```
 e4:   bf00            nop
 e6:   bf00            nop
 e8:   9b03            ldr     r3, [sp, #12]
 ea:   2b04            cmp     r3, #4
 ec:   d007            beq.n   0xfe
 ee:   2069            movs    r0, #105        ; 0x69
 f0:   f7ff ff88       bl      0x4
 f4:   2086            movs    r0, #134        ; 0x86
 f6:   f7ff ff85       bl      0x4
 fa:   b004            add     sp, #16
 fc:   bd10            pop     {r4, pc}
 fe:   2301            movs    r3, #1
100:   9302            str     r3, [sp, #8]
102:   2090            movs    r0, #144        ; 0x90
104:   f7ff ff7e       bl      0x4
108:   2000            movs    r0, #0
10a:   f7ff ff7b       bl      0x4
10e:   e7f4            b.n     0xfa
```

Using an editor, we modified 8 of the byte codes to see if we could create an exploit:



Running on an actual ARM processor, this assembly code on the right would report a valid password regardless of the input.

Finally, we decided to try a simulated SPI read where I would encrypt the data, flip a bit, decrypt the data, then run the modified code on an ARM processor to see if I yielded an exploit. Here is a schematic of the attack pattern:



To prove we didn't just get lucky with the initial exploit, we ran this test on two different versions of the code with different byte alignments (we added a few NOP instructions to change the block alignment of the password check loop). For the first program, 11 of the 128 single-bit flips yielded an exploit. For the second program, 19 out of 128 bit flips yielded an exploit. This means the probability of finding an exploit once the block has been identified is about 11.7%, which is very high compared to traditional glitching attacks that can require thousands of glitch attempts before identifying a single exploit. Further, for this particular attack, once an exploit has been identified, it will be 100% effective compared to normal successful exploit recipes of greater than 10%.

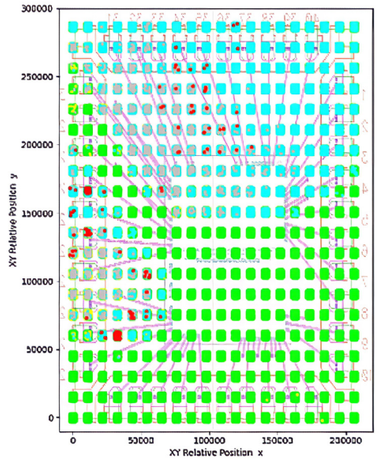
This work motivated us to include authentication for our encrypted execute in place.

Emperical Results on First Silicon

Test 1 — Plain QSPI (unencrypted, unauthenticated)

When our first silicon came back from production, we immediately began testing our part for glitch susceptibility and the effectiveness of our glitch detectors. This first plot shows the results of a glitch campaign against our part with external flash running unencrypted with no authentication. *Note, this is not how we recommend running our chips.*

In this plot, we ran 400 glitch attempts per XY location over a 20x20 scan of the chip package. The green dots represent glitch trials where the part was unaffected by the glitch. Teal dots represent glitches that were detected. Grey dots represent trials where the glitch detectors activated. Finally, the red dots represent exploits of the password code with no glitch detection. You can see there are quite a few red dots in our picture.

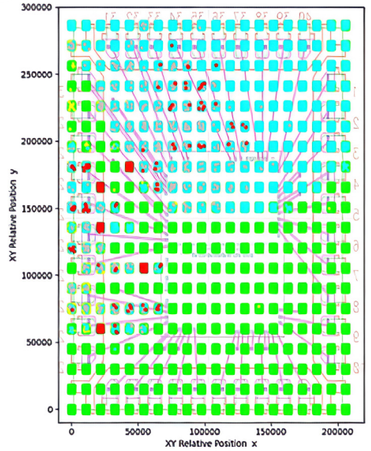


The red smudges are interesting. Looking at the data in the time domain reveals these are areas where the exploited code had been loaded into the cache and remained resident until the next power cycle. We were expecting some exploits to go undetected, as we still had not optimized the glitch detectors on our silicon. Based on our theoretical work on the QSPI glitch attack, we had expected to see evidence of an exploit. However, it was more common than we had anticipated.

Test 2 — Encrypted QSPI (encryption but no authentication)

For the next picture, we re-ran the same experiment with a single change: encrypting the external QSPI flash.

As before, we ran 400 glitch attempts per XY location over a 20x20 scan of the chip package.
As you can see, encrypting the external flash without authentication offered no glitch exploit protection over the plaintext external flash.

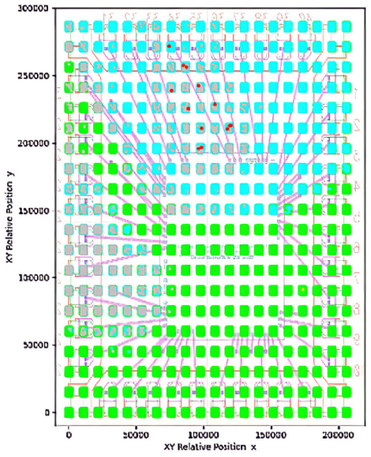


Test 3 — AXIP enabled (encryption + per-read authentication)

For the last test, we enabled both encryption and authentication (i.e., full AXIP mode) of the external flash memory, which will be the default configuration for our customers.

Here are the results.

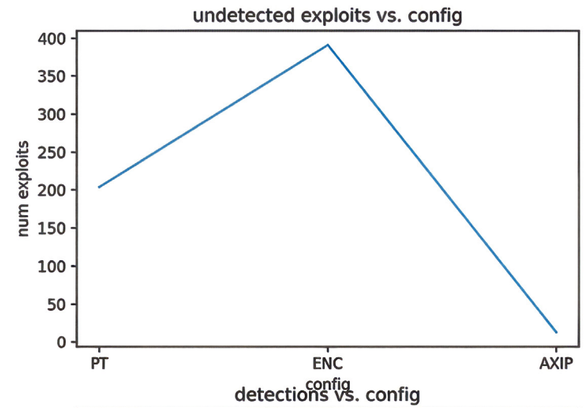
You can see a significant reduction in exploit activity. This means that most of the exploits we had seen in the first two experiments were QSPI flash read exploits. Adding the authentication provided more protection than we had anticipated. But when we look at the raw numbers, we see additional benefits from the authentication as well.

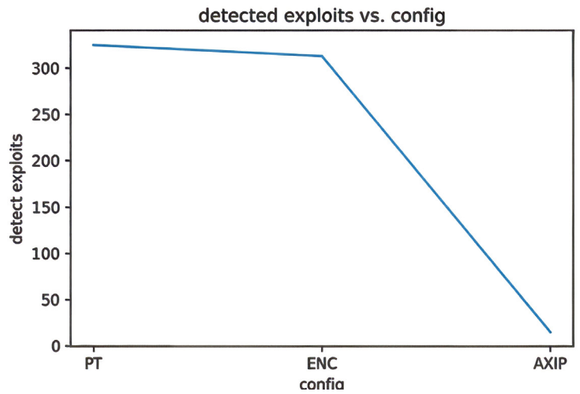


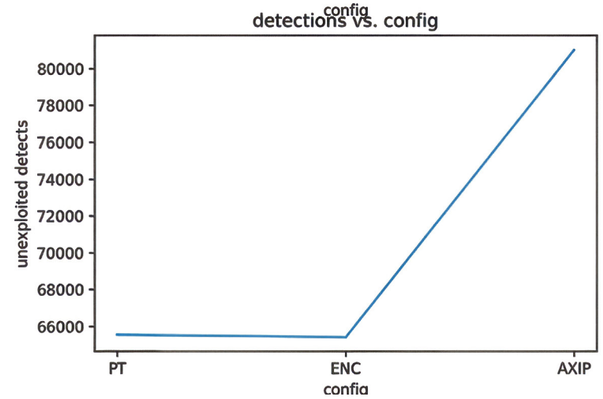
This graph shows us undetected exploits vs. QSPI flash configuration, detected exploits vs. QSPI flash configuration, and raw glitch detections vs. QSPI flash configuration. First, as expected, enabling authentication reduces the detected exploits from about 300 per campaign to about 10. Second, enabling authentication reduces the number of undetected exploits from about 300 on average to about 10.

But there was another significant benefit. Enabling authentication increased the number of detection events from 66,000 to over 80,000. This is an improvement of about 21%. This is because many of the events that would have been detected actually resulted in some non-exploit corruption of the operation of the chip, resulting in a reset or some other random behavior. By improving the processor's overall stability in the face of glitching attacks, we also improved our ability to detect exploits.

I should point out that the glitch detector data shown here is based on unoptimized configuration settings. They perform much better in practice, but we do not share the final data without an NDA.







Why AXIP Works

There are two main reasons for the improved processor resistance to glitching attacks. First, every external flash read is accompanied by an authentication tag. The on-chip controller can then verify this tag before the data reaches the CPU or Cache. A mismatch aborts the data transfer immediately and causes the firmware to re-read the flash.

Second, glitches that would have turned into valid but malicious instructions are converted into verify fail events. This allows the clean firmware to successfully read the glitch detector sensors in place.

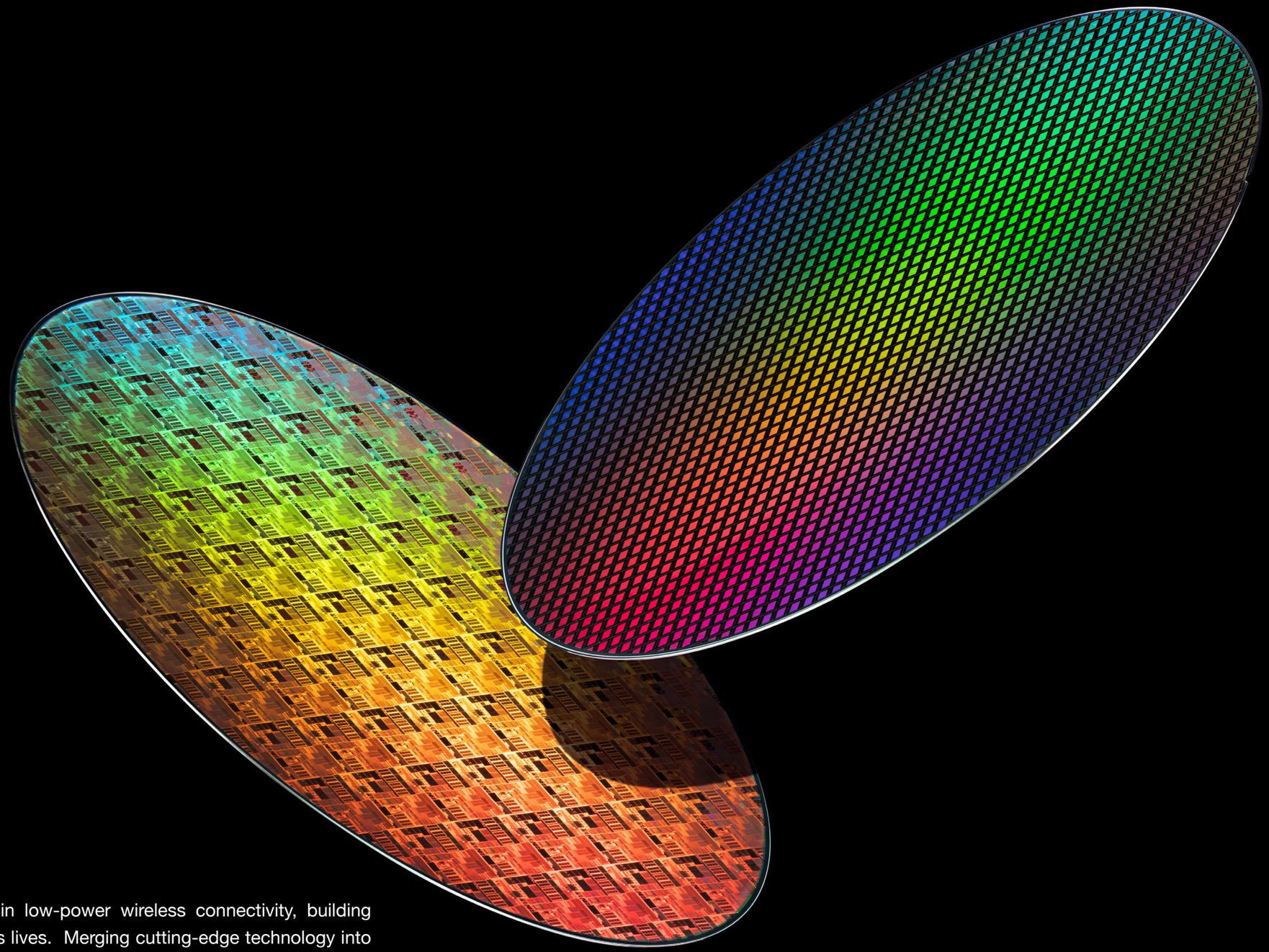
Conclusion: Authenticated XiP Turns Practical Glitch Attacks into Improbable Ones

Contrary to popular opinion, encrypting external flash reads for XIP offers no protection against glitching attacks. In fact, in our experience, encrypting the data flow can enhance the effects of glitching attacks. For block cipher operation, a small number of bit transitions on the encrypted blocks results into 50% bit transitions in the plain text. This is by design of the block ciphers.

By adding per-read authentication, AXIP significantly reduces the attack surface: most glitches never materialize as executable code, and the system is significantly more likely to read out the glitch detection hardware. In our silicon, AXIP reduced both detected and undetected exploits by about 30% and increased raw detection events by 21%.

We recommend enabling both encryption and authentication for XIP. AXIP turns a practical attack into an improbable one. When combined with our glitch detection technology, this makes an EMP attack difficult to carry out in practice.

SILICON LABS
CONNECTED INTELLIGENCE



Silicon Labs

Silicon Labs (NASDAQ: SLAB) is the leading innovator in low-power wireless connectivity, building embedded technology that connects devices and improves lives. Merging cutting-edge technology into the world's most highly integrated SoCs, Silicon Labs provides device makers with the solutions, support, and ecosystems needed to create advanced edge connectivity applications. Headquartered in Austin, Texas, Silicon Labs has operations in over 16 countries and is the trusted partner for innovative solutions in the smart home, industrial IoT, and smart cities markets. Learn more at www.silabs.com.