

GALS System Design

Side Channel Attack Secure Cryptographic Accelerators

Frank K. Gürkaynak

Integrated Systems Laboratory
ETH Zurich

22 November 2005

Ph.D. Thesis Presentation

Outline

- 1 Outline
- 2 Globally-Asynchronous Locally-Synchronous (GALS) Design
- 3 Cryptography
- 4 GALS implementation of the AES Algorithm
- 5 Results and Conclusions

What is wrong with the way we design chips now?

Modern System-on-Chip circuits ...

- Contain millions of transistors
- Require clock rates exceeding 100s of MHz
- Include 100s of subblocks
- Use 10s of different clock domains

What is wrong with the way we design chips now?

Modern System-on-Chip circuits ...

- Contain millions of transistors, trend **increasing**
- Require clock rates exceeding 100s of MHz, trend **increasing**
- Include 100s of subblocks, trend **increasing**
- Use 10s of different clock domains, trend **increasing**

What is wrong with the way we design chips now?

Modern System-on-Chip circuits ...

- Contain millions of transistors, trend **increasing**
- Require clock rates exceeding 100s of MHz, trend **increasing**
- Include 100s of subblocks, trend **increasing**
- Use 10s of different clock domains, trend **increasing**

... are not easy to design

- The clock signal must be distributed to an increasing number of elements with increased precision
- Many independently designed components must be combined to a large system.
- All subsystems must be able to reliably exchange data

Globally-Asynchronous Locally-Synchronous Design

GALS is a methodology to enable the design of complex digital systems on chip.

- System is divided into smaller GALS modules
- Each module works synchronously
- Interconnected modules communicate asynchronously

Globally-Asynchronous Locally-Synchronous Design

GALS is a methodology to enable the design of complex digital systems on chip.

- System is divided into smaller GALS modules
- Each module works synchronously
- Interconnected modules communicate asynchronously
- Was first developed by D. Chapiro in 1984
- First chip implementation by J. Muttersbach in 1999

Globally-Asynchronous Locally-Synchronous Design

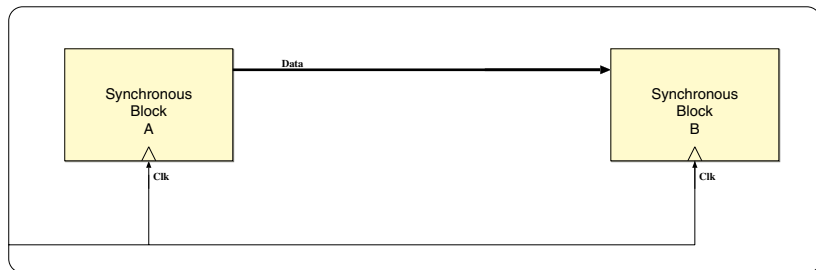
GALS is a methodology to enable the design of complex digital systems on chip.

- System is divided into smaller GALS modules
- Each module works synchronously
- Interconnected modules communicate asynchronously
- Was first developed by D. Chapiro in 1984
- First chip implementation by J. Muttersbach in 1999

GALS implementations differ in:

- synchronization method between blocks
- specific asynchronous communication protocol used

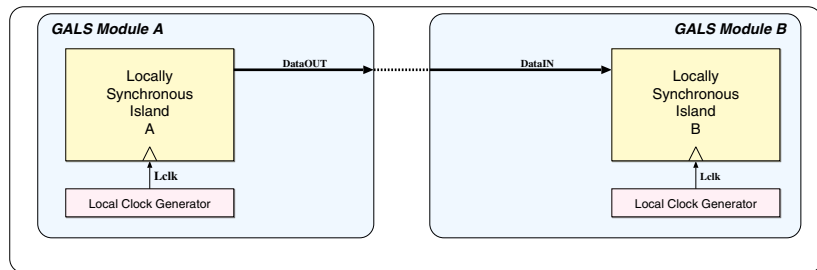
Basic GALS Structure



Synchronous system

Two large functional blocks of a synchronous system

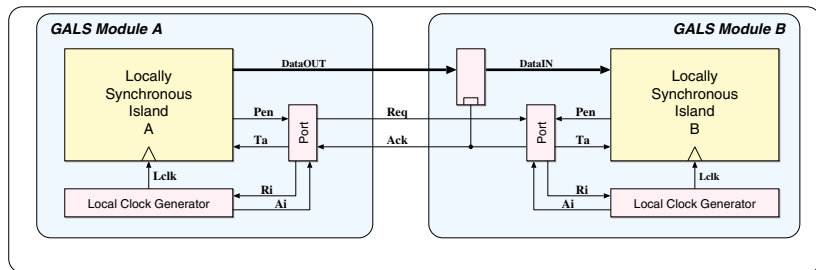
Basic GALS Structure



Local clock generators

GALS modules are formed by adding a local clock generator for each functional block

Basic GALS Structure



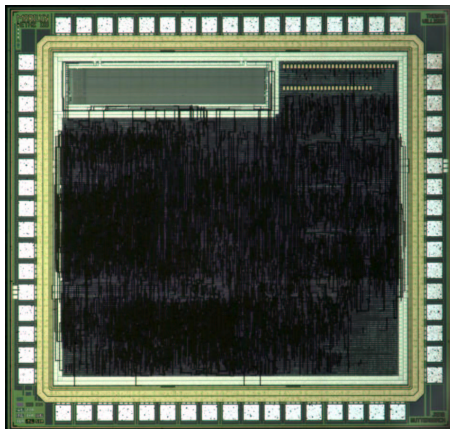
GALS system

Port controllers are added to regulate data transfers between GALS modules

GALS Works

GALS at IIS

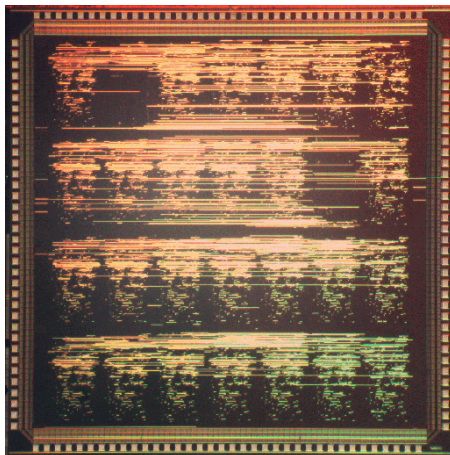
- **J. Muttersbach**
First implementation
- **T. Villiger**
- **S. Oetiker**
- **F. K. Gürkaynak**



GALS Works

GALS at IIS

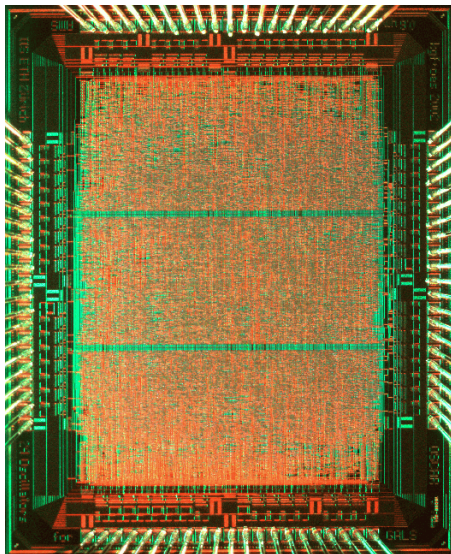
- J. Muttersbach
- T. Villiger
Multi-point
interconnect
- S. Oetiker
- F. K. Gürkaynak



GALS Works

GALS at IIS

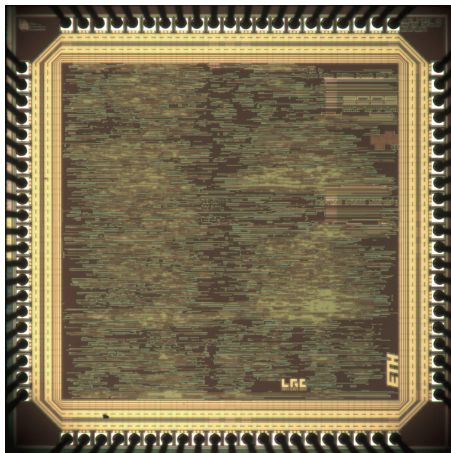
- J. Muttersbach
- T. Villiger
- S. Oetiker
Local clock
generators
- F. K. Gürkaynak



GALS Works

GALS at IIS

- J. Muttersbach
 - T. Villiger
 - S. Oetiker
 - F. K. Gürkaynak
- Design and test flow

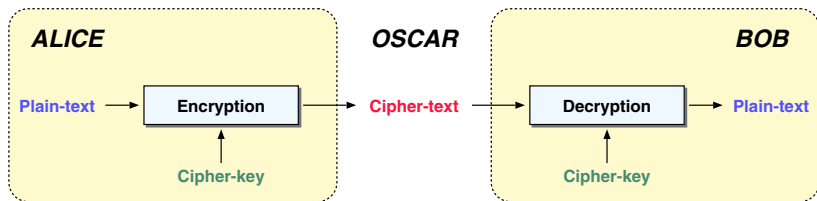


Why GALS ?

Advantages

- No global clock distribution problems
- Modular design flow
- Potential for low-power design
- Offers new possibilities for designers

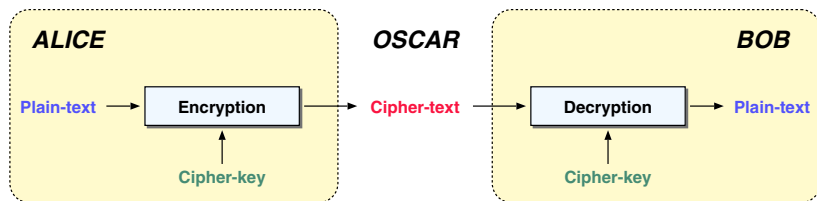
Cryptography 101



Private key ciphers

- Alice encrypts **plain-text** information by using a **cipher-key**.
- Bob can decrypt the resulting **cipher-text** only if he has access to the same cipher-key.

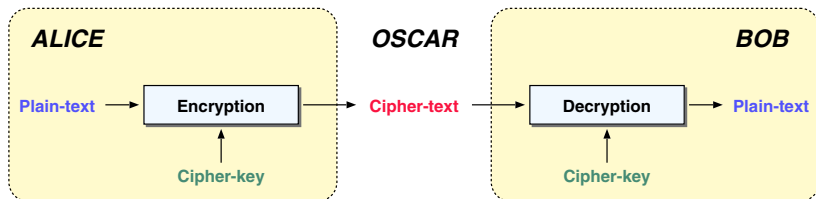
Cryptography 102



Security

- Oscar wishes to obtain the plain-text
- Oscar knows everything about the cryptographic algorithm
- Oscar can observe/modify the cipher-text
- but..

Cryptography 102



Security

- Oscar wishes to obtain the plain-text
- Oscar knows everything about the cryptographic algorithm
- Oscar can observe/modify the cipher-text
- but.. Oscar **does not know** the cipher-key

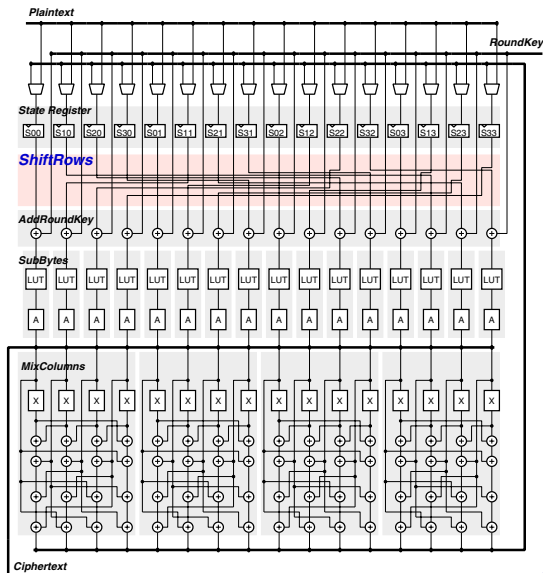
Advanced Encryption Standard (AES)

AES Standard

- by NIST 2001
- 128 bit data
- 128 bit key
- 10/12/14 rounds

Components

- ShiftRows



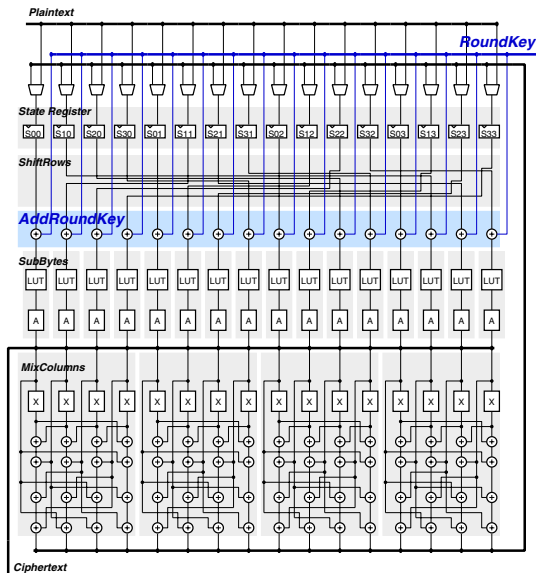
Advanced Encryption Standard (AES)

AES Standard

- by NIST 2001
- 128 bit data
- 128 bit key
- 10/12/14 rounds

Components

- ShiftRows
- AddRoundKey



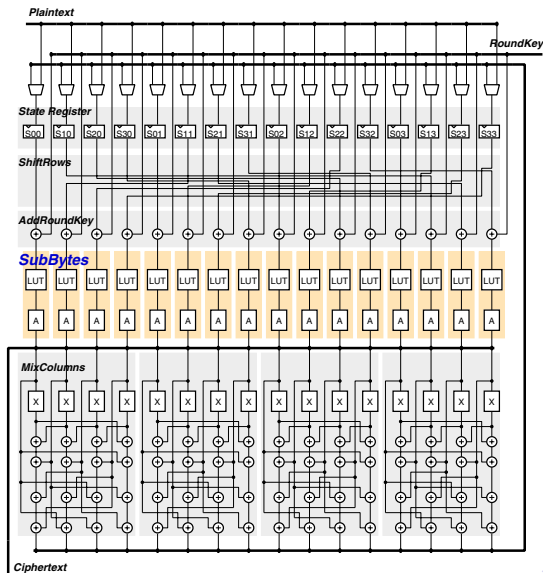
Advanced Encryption Standard (AES)

AES Standard

- by NIST 2001
- 128 bit data
- 128 bit key
- 10/12/14 rounds

Components

- ShiftRows
- AddRoundKey
- SubBytes



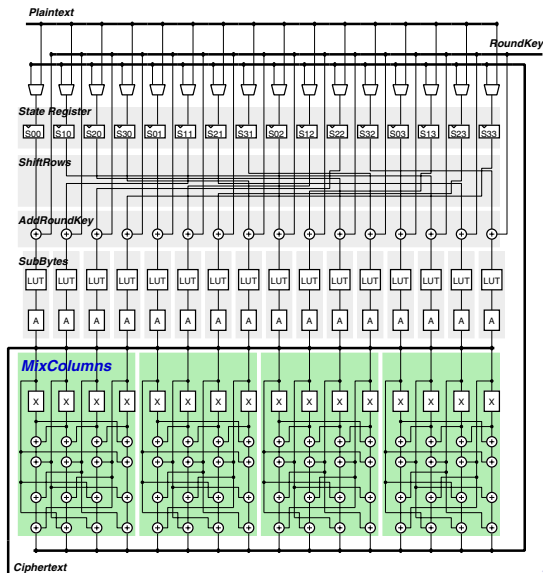
Advanced Encryption Standard (AES)

AES Standard

- by NIST 2001
- 128 bit data
- 128 bit key
- 10/12/14 rounds

Components

- ShiftRows
- AddRoundKey
- SubBytes
- MixColumns



Side-Channels

Once an otherwise secure algorithm is implemented in either Hardware or Software it gains physical properties that can be observed:

- Time required to finish the operation
- Power consumption
- Electromagnetic Radiation
- Heat dissipation
- Sound

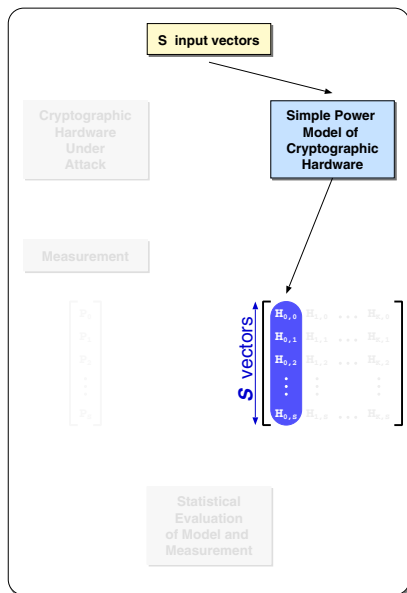
These properties are called **Side Channels**

Side-Channel Attacks

In 1996, P. Kocher showed that it is possible to obtain additional information on the cipher-key by observing these side-channels.

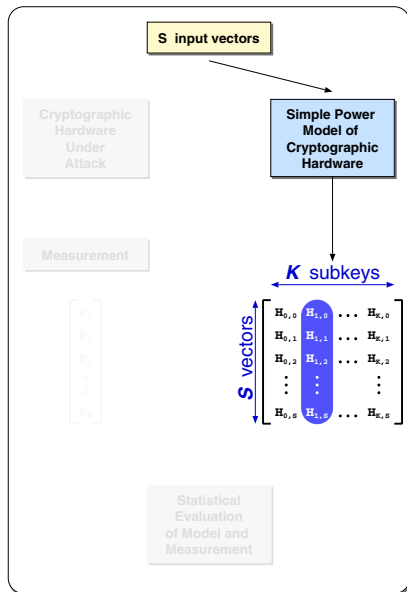
Differential Power Analysis (DPA)

- 1 Select a *subkey* and a *target operation*
- 2 Use a simple model to *predict* the power consumption for **S input vectors**



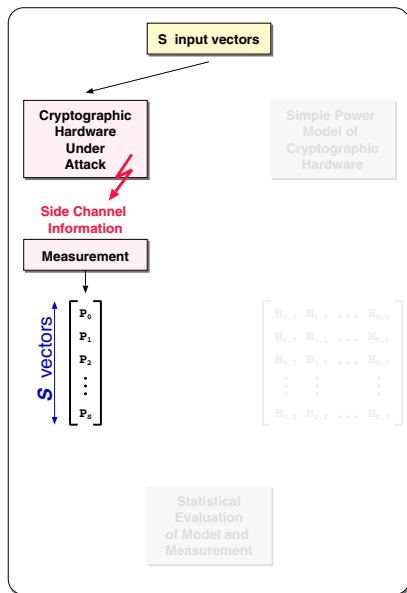
Differential Power Analysis (DPA)

- 1 Select a *subkey* and a *target operation*
- 2 Use a simple model to *predict* the power consumption for S input vectors
- 3 predict the power consumption for **all K subkey permutations**



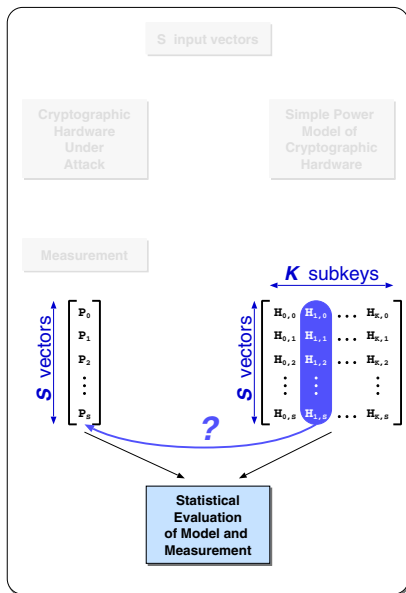
Differential Power Analysis (DPA)

- 1 Select a *subkey* and a *target operation*
- 2 Use a simple model to *predict* the power consumption for S input vectors
- 3 predict the power consumption for all K subkey permutations
- 4 *Measure* the power consumption using the same S **input vectors**



Differential Power Analysis (DPA)

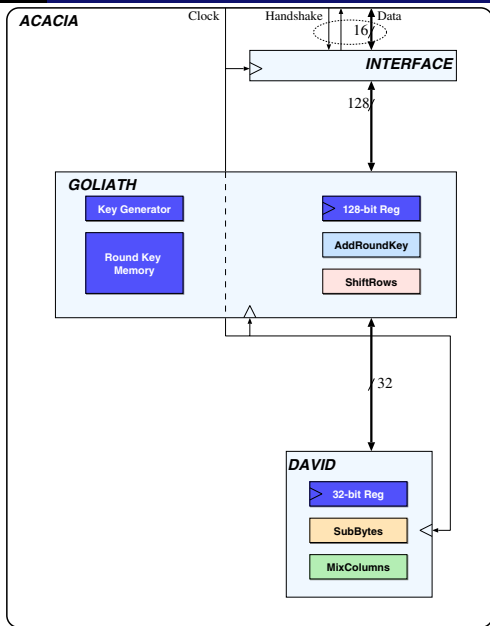
- 1 Select a *subkey* and a *target operation*
- 2 Use a simple model to *predict* the power consumption for S input vectors
- 3 predict the power consumption for all K subkey permutations
- 4 *Measure* the power consumption using the same S input vectors
- 5 *Determine* if one of the power hypotheses shows a **distinctively higher correlation** to the measurement.



Block Diagram

The GALS implementation is called **Acacia**.

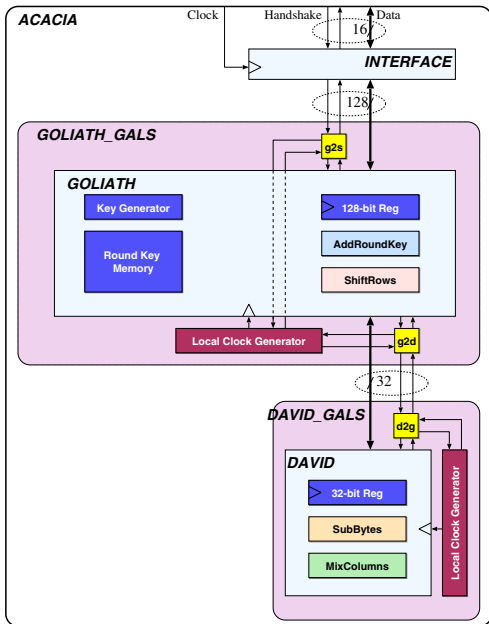
- Operations are divided between a 128-bit **Goliath** and a 32-bit **David** unit



Block Diagram

The GALS implementation is called **Acacia**.

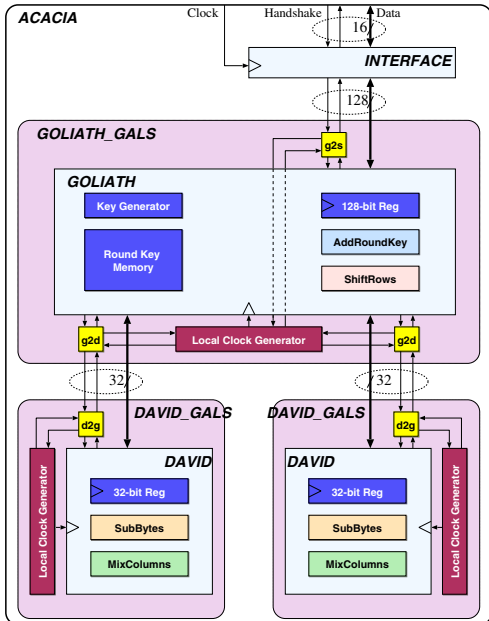
- Operations are divided between a 128-bit **Goliath** and a 32-bit **David** unit
- David and Goliath are separate GALS modules



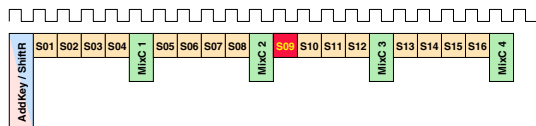
Block Diagram

The GALS implementation is called **Acacia**.

- Operations are divided between a 128-bit **Goliath** and a 32-bit **David** unit
- David and Goliath are separate GALS modules
- There is a second David unit running in parallel.
- One round of AES requires 1 Goliath and 4 David operations.



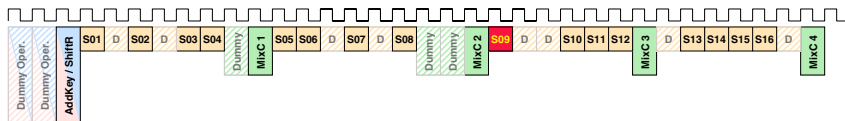
Implemented Countermeasures



Normal Operation

The attacker will normally target a single operation, and will measure the power consumption of this particular clock cycle.

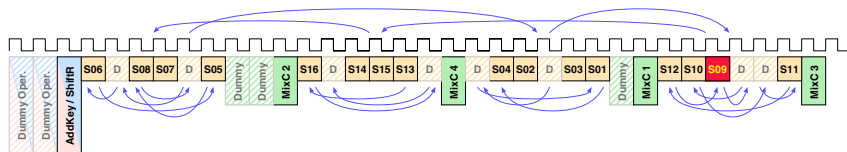
Implemented countermeasures



Inserting dummy operations

Inserting random dummy cycles will confuse the attacker, since the targeted operation will not always be executed at a specific clock cycle. Unfortunately, this also increases the run-time.

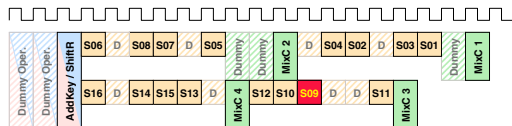
Implemented Countermeasures



Change ordering of operations

Independent operations can be re-ordered arbitrarily. Contrary to inserting dummy cycles, this does not increase the run-time.

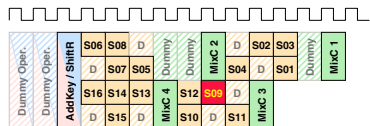
Implemented Countermeasures



Parallelization

Executing operations in parallel creates more activity at the same time, this appears as noise for the attacker.

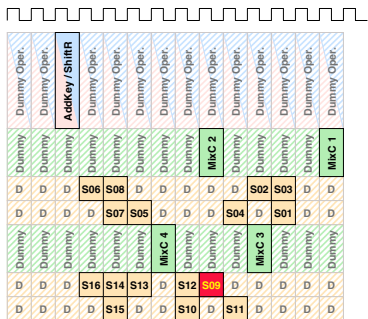
Implemented Countermeasures



Parallelization

Executing operations in parallel creates more activity at the same time, this appears as noise for the attacker.

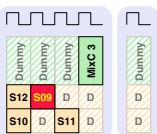
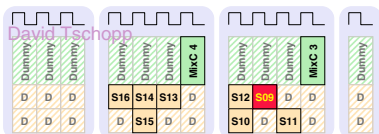
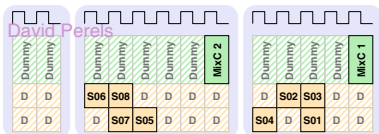
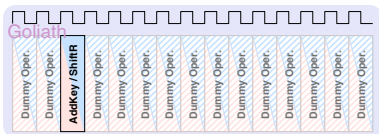
Implemented Countermeasures



Parallelization

Executing operations in parallel creates more activity at the same time, this appears as noise for the attacker.

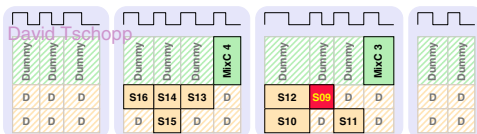
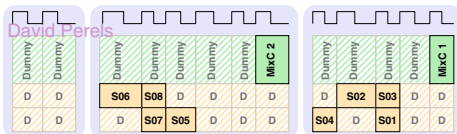
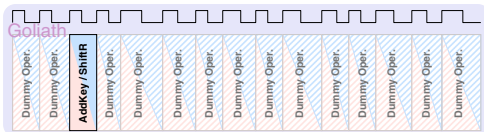
Implemented Countermeasures



Introducing GALS modules

GALS modules have their own local clock generator, their clocks are independent and can not be controlled by the attacker

Implemented Countermeasures



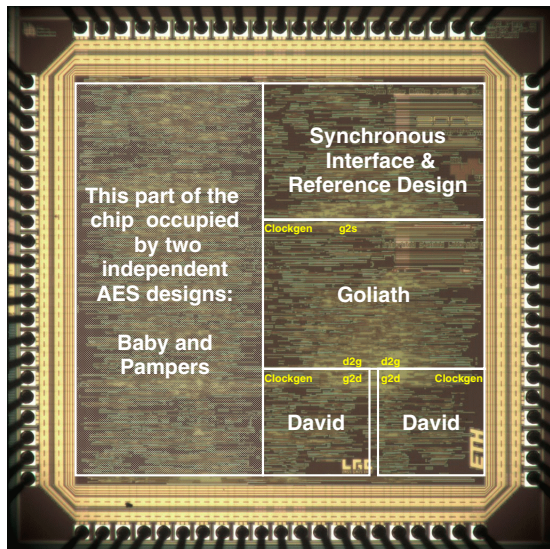
Variable clock periods

Each GALs module can randomly change its own clock period.
This adds even more uncertainty

Chip Photo

Acacia

- UMC 0.25 μm CMOS
- Total area 1.75 mm^2
 - David 0.221 mm^2
 - Goliath 0.687 mm^2
 - Sync. 0.584 mm^2
- Rate 177.7 Mb/s
- Energy 1.232 mJ/Mb



Conclusions

Conclusions

- A novel **GALS based crypto ASIC** implementing the AES algorithm was presented.
- In addition to traditional DPA countermeasures, the chip also includes GALS modules that use **randomly varying clocks** which make known attacks extremely difficult
- The GALS design methodology was refined. The presented design was designed **using mainly standard EDA tools**.
- A combination of functional and scan-chain based testing allows a **stuck-at-coverage** of more than **99.8%**.

Conclusions

Conclusions

- A novel **GALS based crypto ASIC** implementing the AES algorithm was presented.
- In addition to traditional DPA countermeasures, the chip also includes GALS modules that use **randomly varying clocks** which make known attacks extremely difficult
- The GALS design methodology was refined. The presented design was designed **using mainly standard EDA tools**.
- A combination of functional and scan-chain based testing allows a **stuck-at-coverage** of more than **99.8%**.

Is this really secure?

We don't know yet. The security has to be evaluated by cryptanalysts.

QUESTIONS ?

Acknowledgements for GALS

Stephan Oetiker, Thomas Villiger, Hubert Kaeslin, Norbert Felber

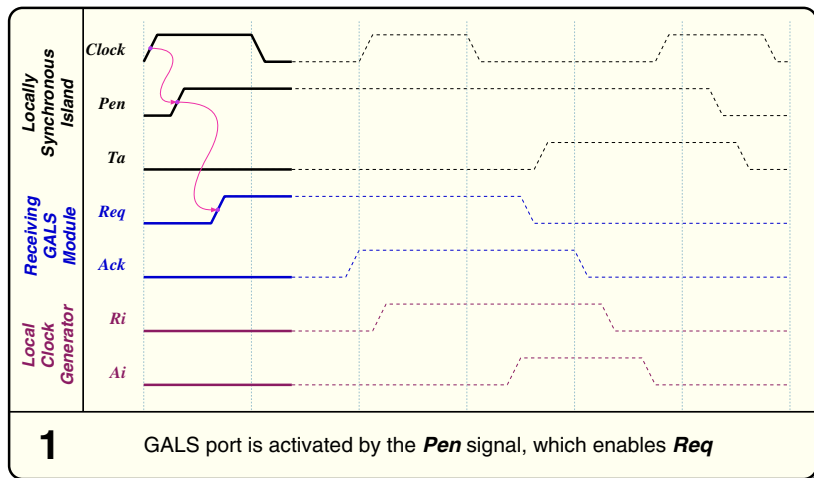
Acknowledgements for Crypto-chips

Stefan Achleitner, Gérard Basler, Andres Erni, Dominique Gasser, Peter Haldi, Franco Hug, Adrian Lutz, Norbert Pramstaller, Stefan Reichmuth, Pieter Rommens, Jürg Treichler, Stefan Zwicky

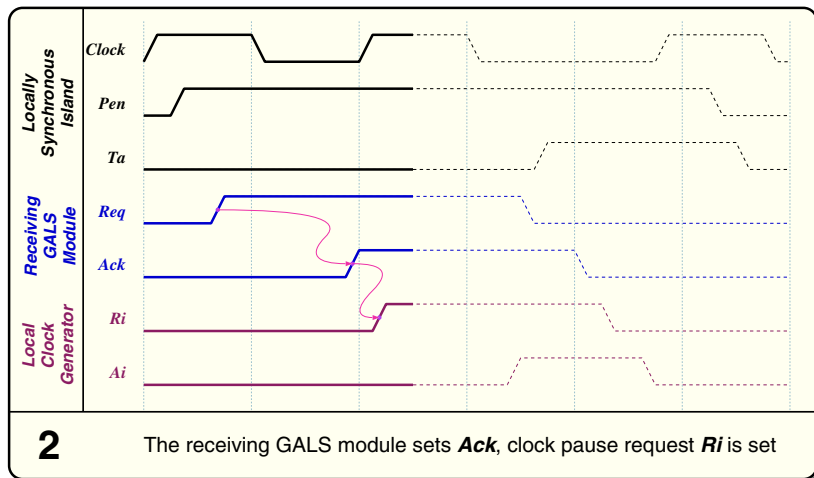
and

Andreas Burg, Matthias Braendli, Stefan Eberli, Simon Haene
Stefan Mangard, Elisabeth Oswald, S. Berna Örs

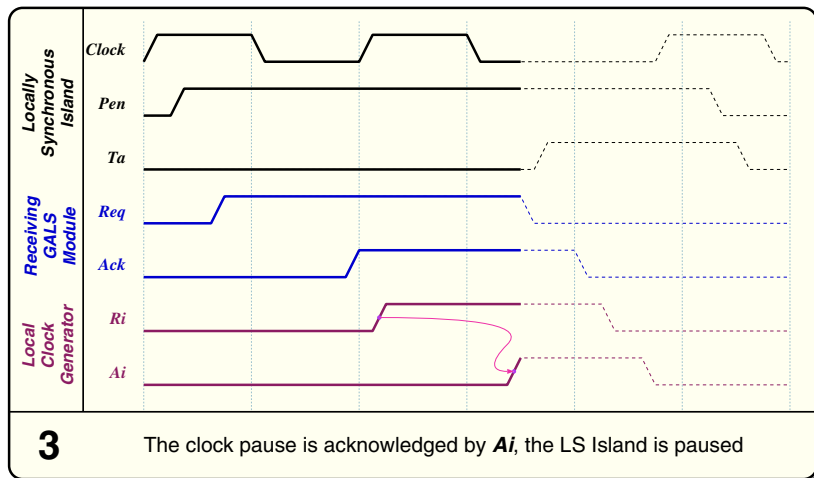
Timing Diagram



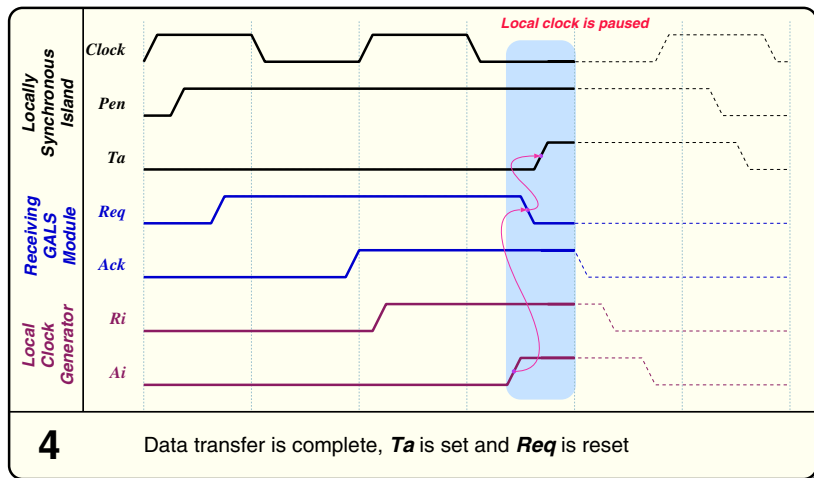
Timing Diagram



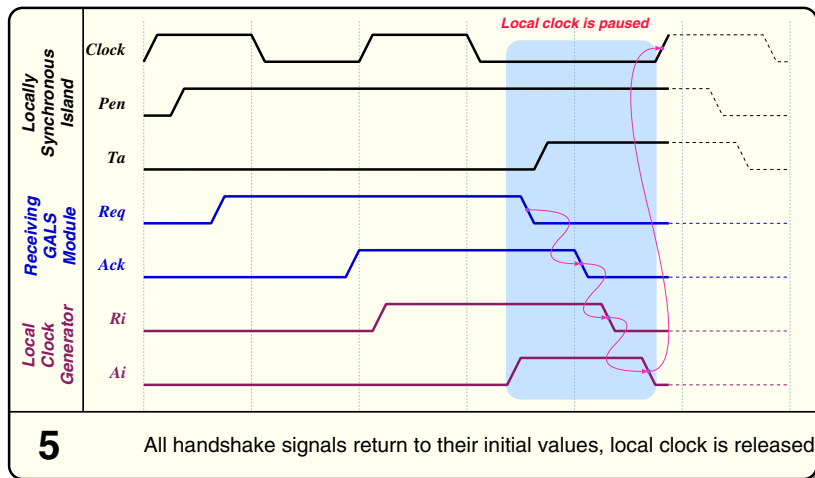
Timing Diagram



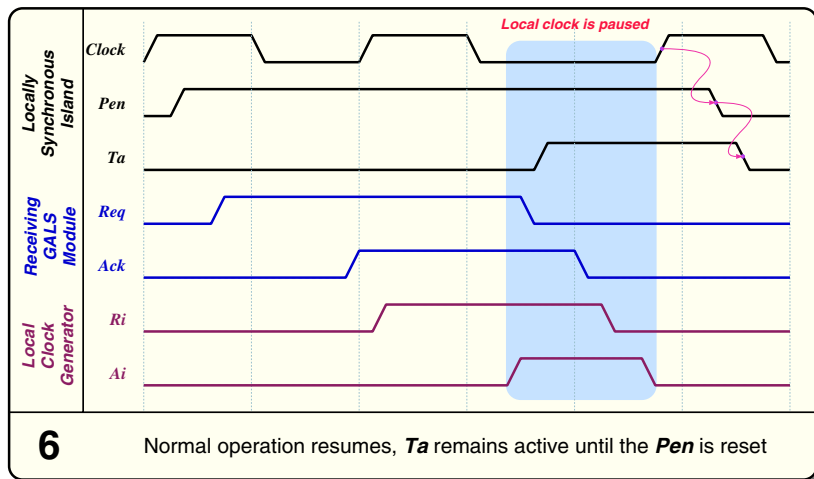
Timing Diagram



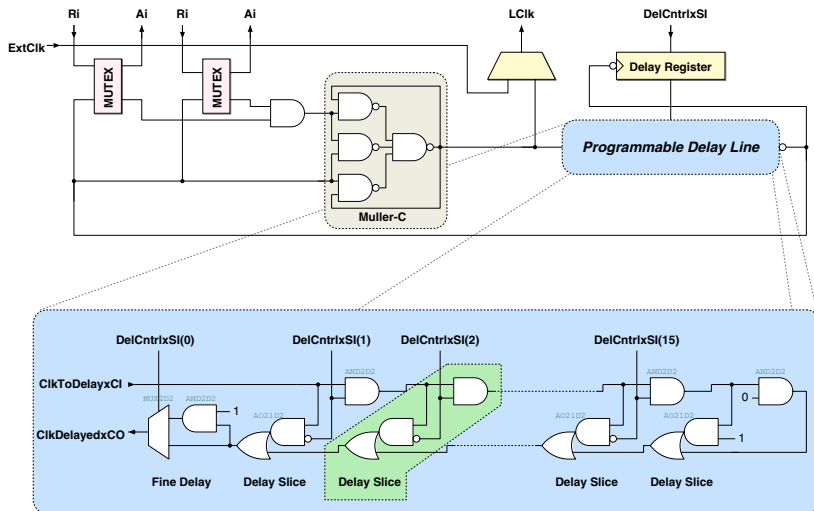
Timing Diagram



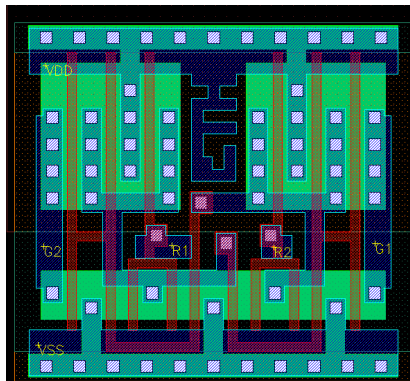
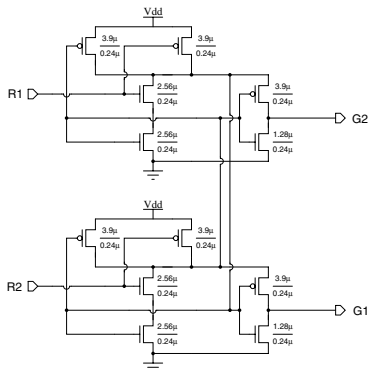
Timing Diagram



Local Clock Generator



Mutual exclusion element

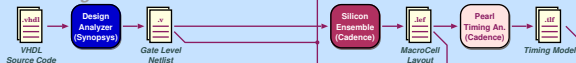


Design flow for GALS (as used in Shir-Khan)

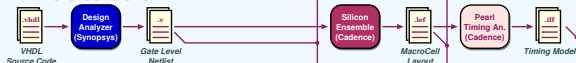
Self-timed library



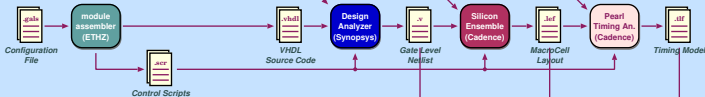
Local clock generator



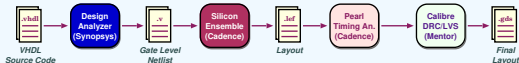
SIMD micro-controller



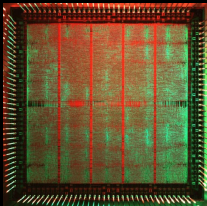
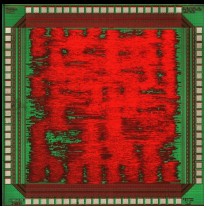
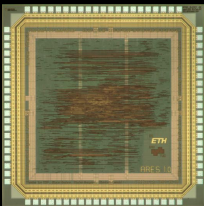
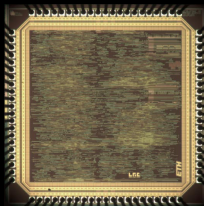
GALS modules



GALS system



AES implementations at IIS

			
Riddler	Fastcore	Ares	Baby / Pampers
2 x 128 bit parallel	128 bit	128 / 32 bit	16 bit
2.16 Gb/s (pipelined)	2.12 Gb/s	1.15 Gb/s (128 bit)	0.285 / 0.230 Gbit/s
37.8 mm² (0.6 μm)	3.56 mm² (0.25 μm)	1.2 mm² (0.25 μm)	0.35 / 0.58 mm² (0.25 μm)
En/Decryption (ECB)	En/Decryption (all)	Encryption (ECB/OFB)	Encryption (ECB/OFB)
Parallel Datapath	Independent Enc/Dec	Includes masking	Plain / Countermeasure

SubBytes determines AES performance

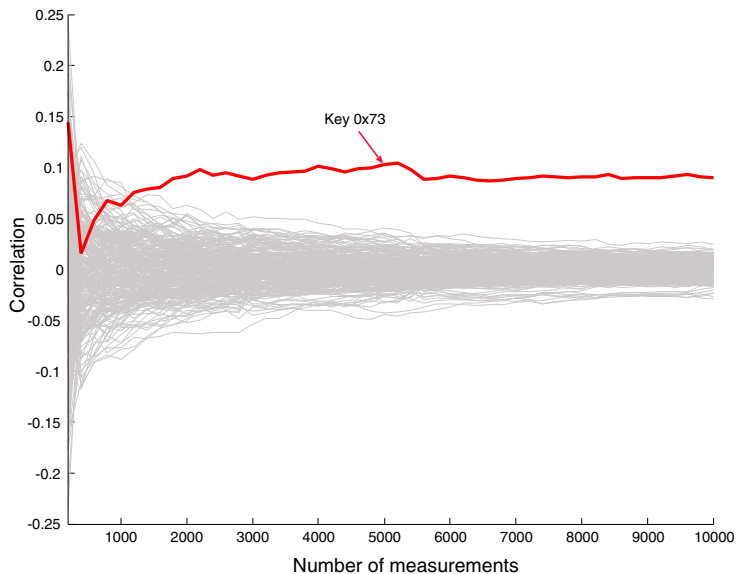
Datapath width	8-bit	16-bit	32-bit	64-bit	128-bit
Parallel SubBytes units	1	2	4	8	16
Complexity (gate eq)	5,052	6,281	7,155	11,628	20,410
Area (normalized)	1	1.266	1.472	2.432	4.269
Clock cycles for AES-128	160	80	40	20	10
Critical path (normalized)	1.349	1.341	1.206	1.133	1
Total time (normalized)	21.580	10.729	4.825	2.227	1

Countermeasures against DPA attacks

Protect your weak spots

- **DPA measures power consumption**
Add **Noise** (unrelated switching activity) to confuse the measurements
- **DPA targets a specific operation**
Change the operation order by inserting **Random Operations**
- **Power consumption of CMOS is data dependent**
Use **Alternative Logic Styles**
- **There are direct operations between input and output**
Prevent direct operations by **Masking** the key with random data

DPA attacks work



DPA attack setup



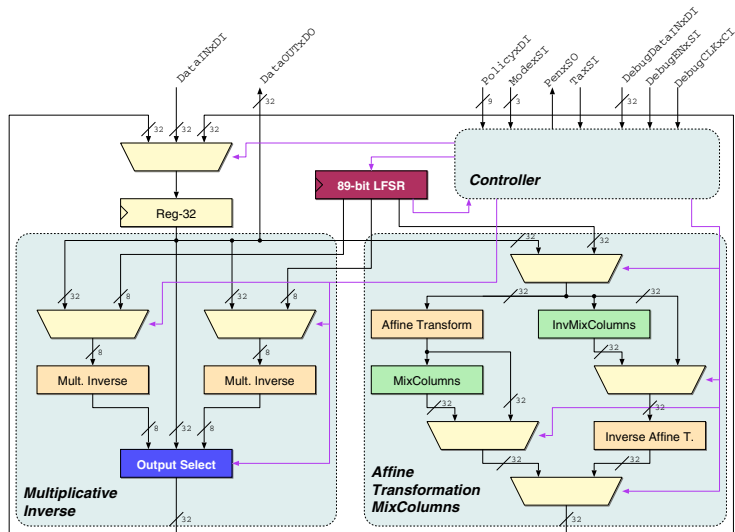
Area overhead of GALS

	David		Goliath	
Area μm^2	183,007	92.98%	551,194	96.66%
Area μm^2 -LSFRs	26,928	13.68%	73,512	12.89%
Area μm^2 -ClockGen	7,579	3.85%	7,626	1.34%
Area μm^2 -Ports	6,225	3.16%	11,412	2.00%
Area μm^2 -GALS	196,811	100.00%	570,233	100.00%
Area μm^2 -TOTAL	963,855			

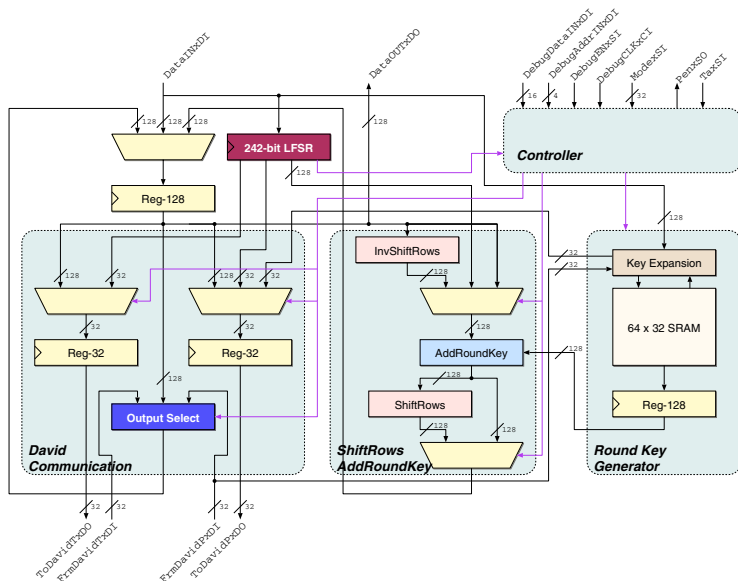
Latency overhead of GALS

	Synchronous		GALS+DPA	
	David	Goliath	David	Goliath
Critical path (ns)	5.43	5.84	3.98	5.27
Latency (cycles)	3	1	4	2
Clock freq. (MHz)	170.96		250.8	189.6
Enc(clock cycles)	7		8	2
Enc time (ns)	40.88		42.38	

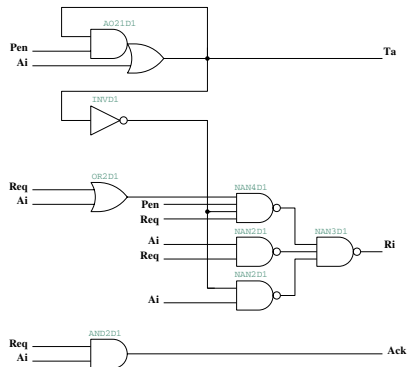
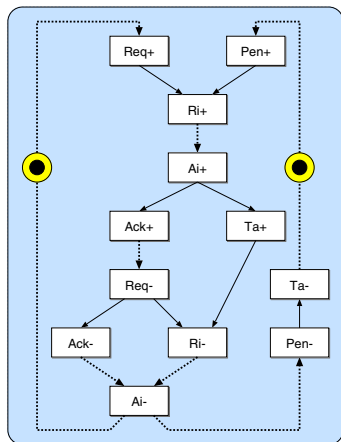
Block diagram of David



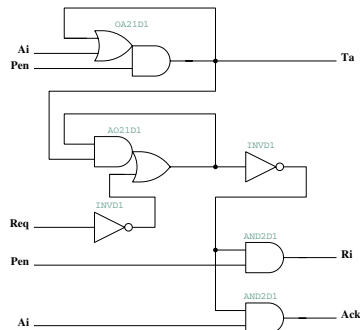
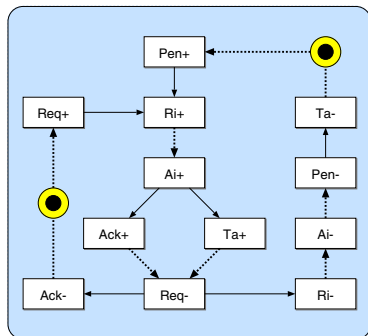
Block diagram of Goliath



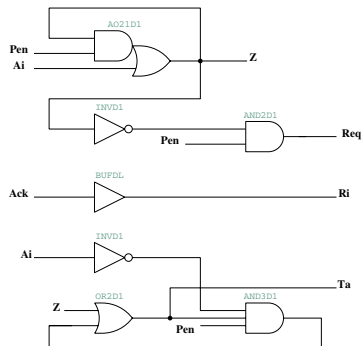
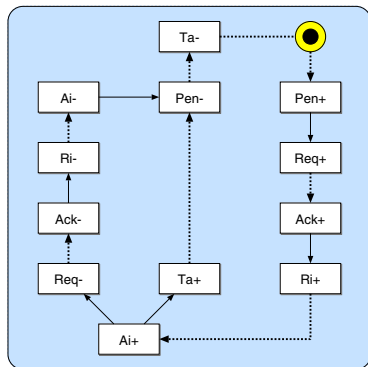
Goliath to Synchronous Interface



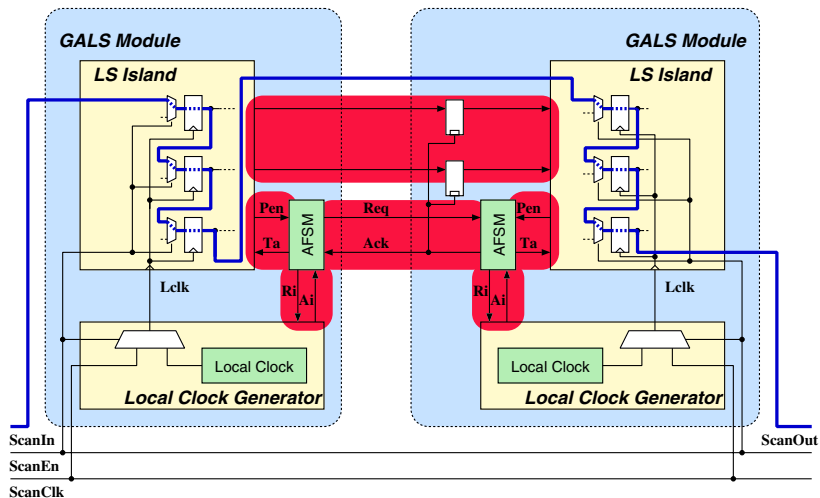
Goliath to David



David to Goliath



Scan-test configuration

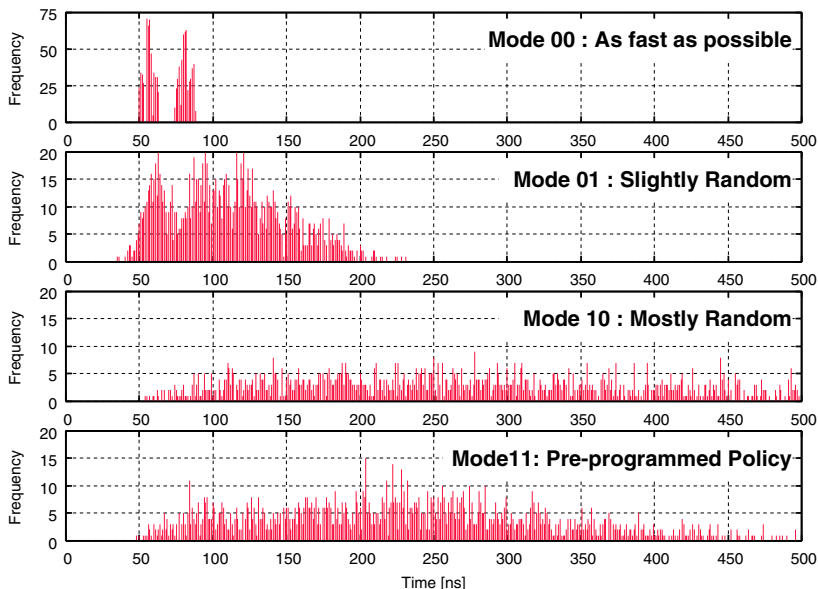


Stuck-at-fault test coverage

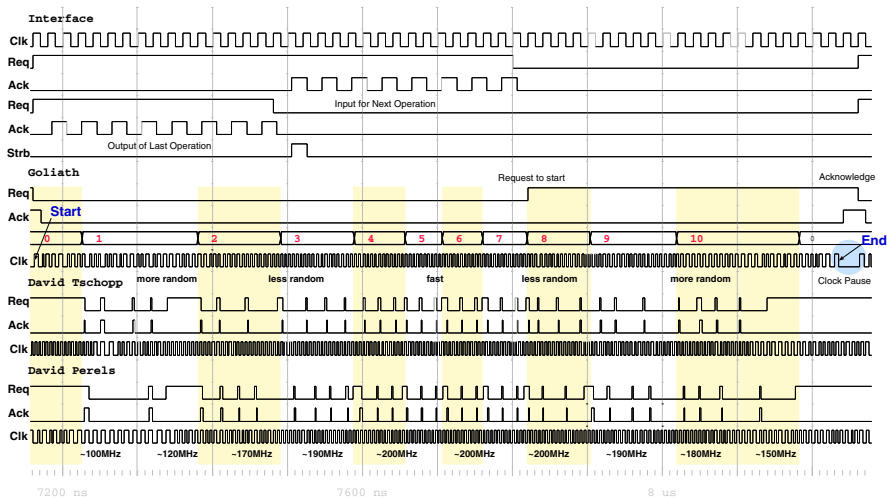
Stuck-at-fault testing

- There are a **total** of 154.604 stuck-at faults in the entire circuit
- **Only 182** of these faults are within the asynchronous finite state machines
- A straightforward test vector generation using TetraMax **fails** to detect **3.089** faults
- Using a simple encryption/decryption operation **2.796** of these faults were **detected by simulation**.
- The **total test coverage** obtained by combining these two methods exceeds **99.8%**.

Distribution of the first SubBytes operation



Simulation result

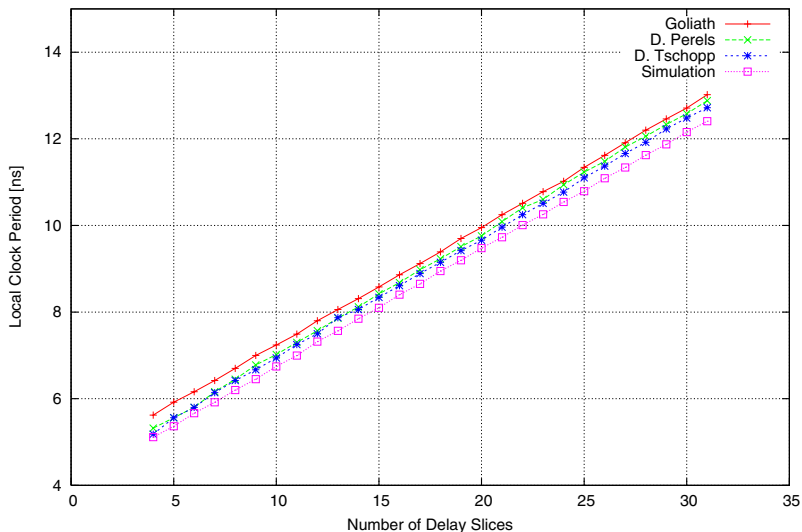


Operation modes of Acacia

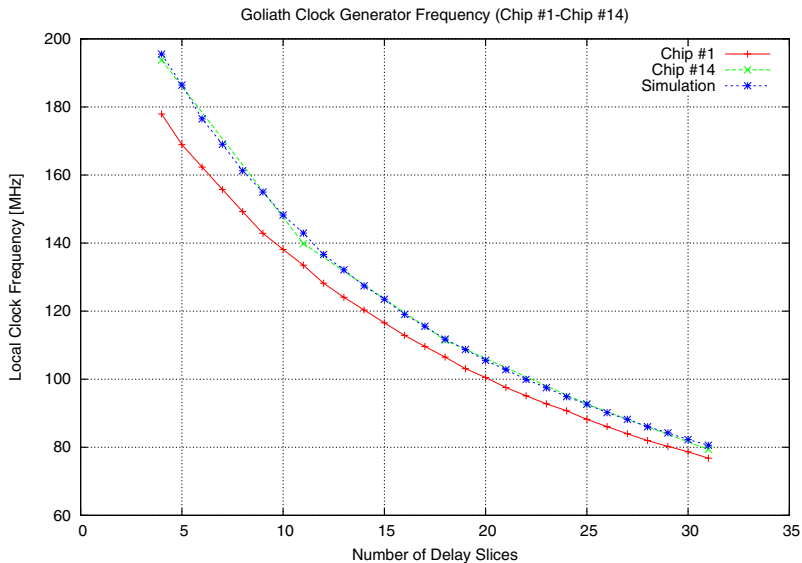
Operation Mode	I/O Clock [MHz]	Encr. [ns]	Throughput [Mb/s]	Energy [mJ/Mb]
Acacia - 00	50	720.0	177.7	1.232
Acacia - 01	50	880.0	145.4	1.362
Acacia - 10	50	2,440.0	57.1	2.704
Acacia - 11	50	920.0	139.1	1.198
Synchronous	150	779.2	164.2	0.976

Clock period versus delay-line settings

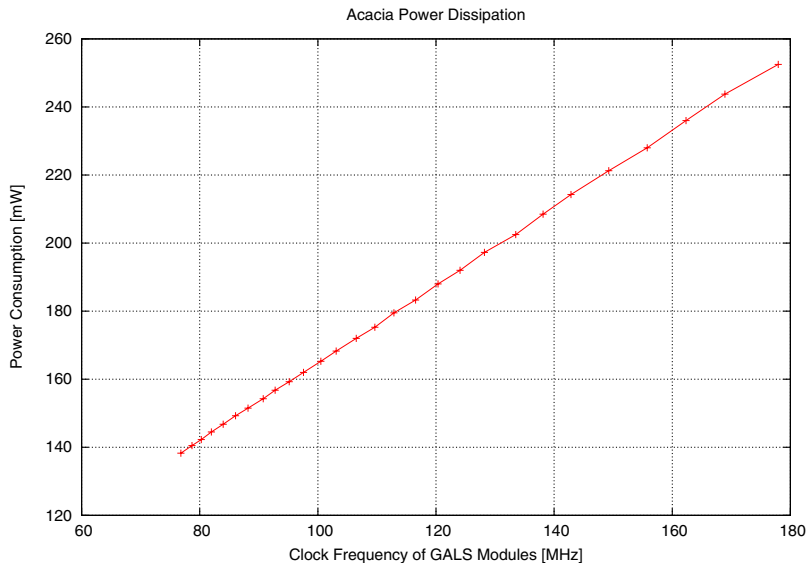
Acacia Local Clock Generator Period (Chip #1)



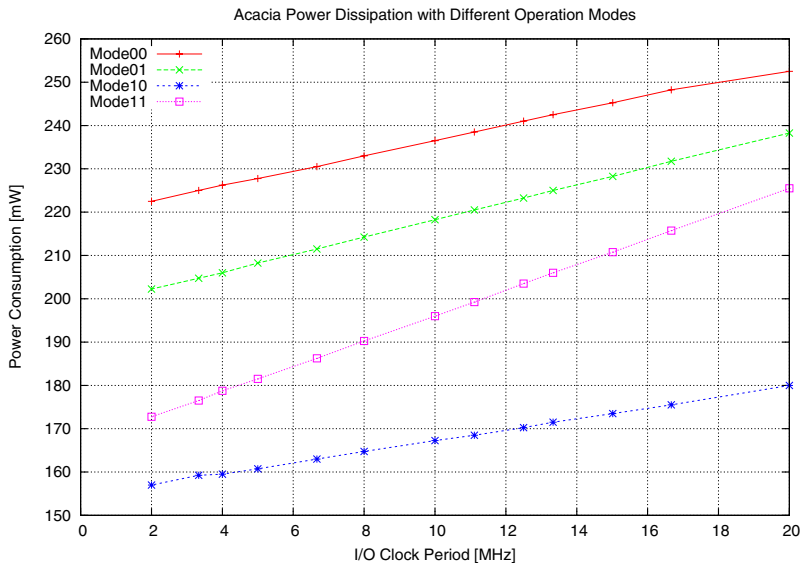
Clock frequency versus delay-line settings



Power consumption vs maximum GALS module frequency



Power consumption of different operation modes



F. Kağan Gürkaynak in KG

