A Globally-Asynchronous Locally-Synchronous VLSI Circuit for the SAFER Cryptoalgorithm

T.Villiger, J.Muttersbach, H.Kaeslin, N.Felber, W.Fichtner

Integrated Systems Laboratory, Swiss Federal Institute of Technology

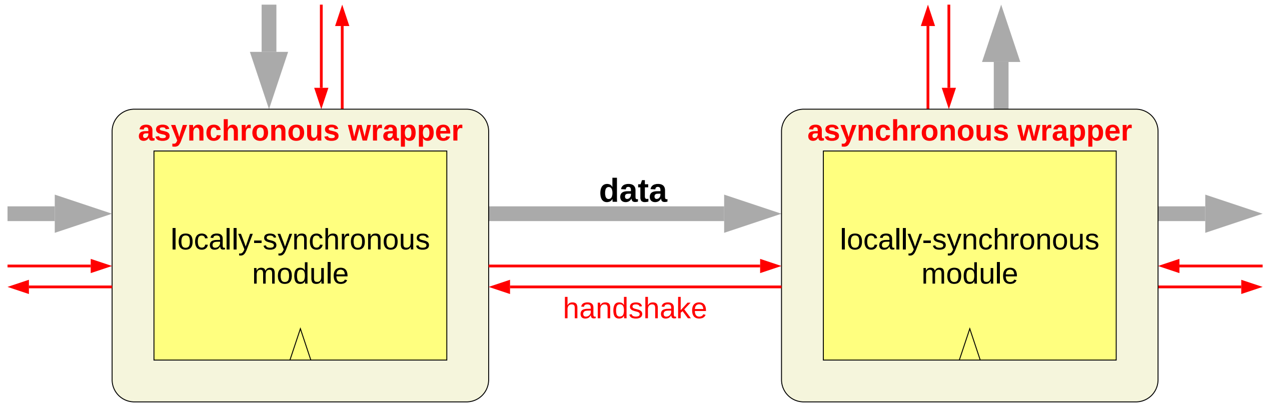
ACiD-WG Workshop, Neuchatel, Switzerland

Funded by: Infineon Technologies, Germany
Philips Semiconductors Zürich, Switzerland
Commission for Technology and Innovation, Switzerland

Outline

GALS Principle

- Locally-synchronous (LS) modules perform all functionality
- Data are transferred in self-timed manner between LS modules



Benefits:

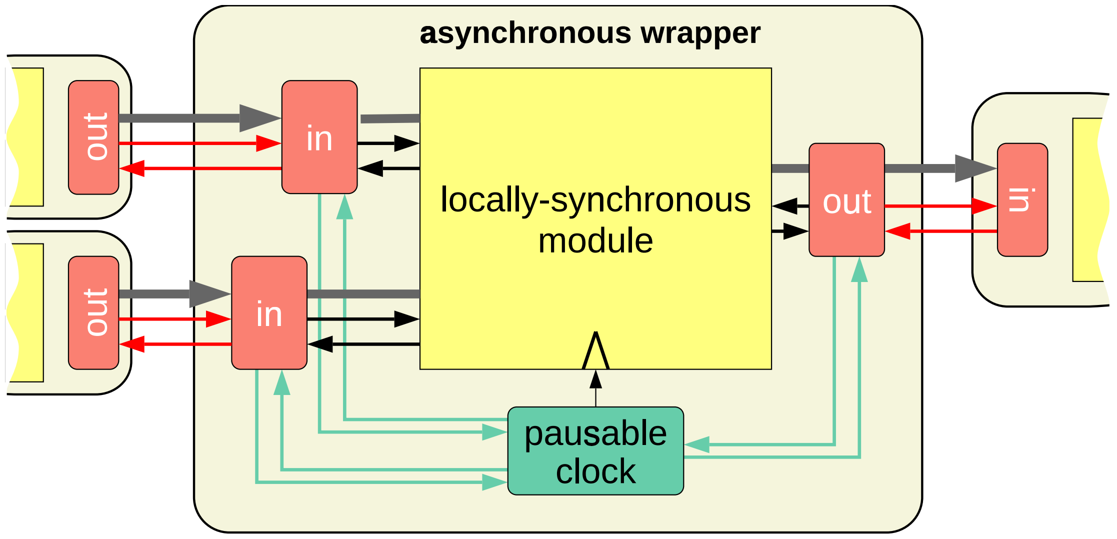
➜ Facilitates clocking of SOCs

➜ Modularity enhances optimization and re-use of LS blocks

➜ Provides a hook for low-power operation

➜ Natural inclusion of totally asynchronous modules

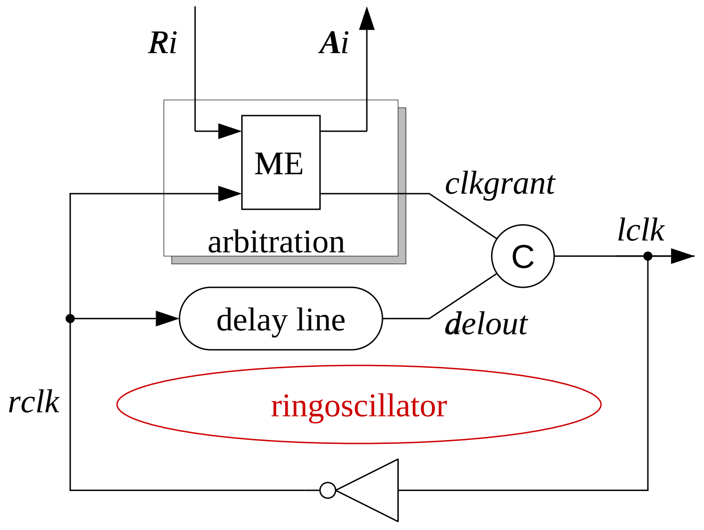
The asynchronous wrapper

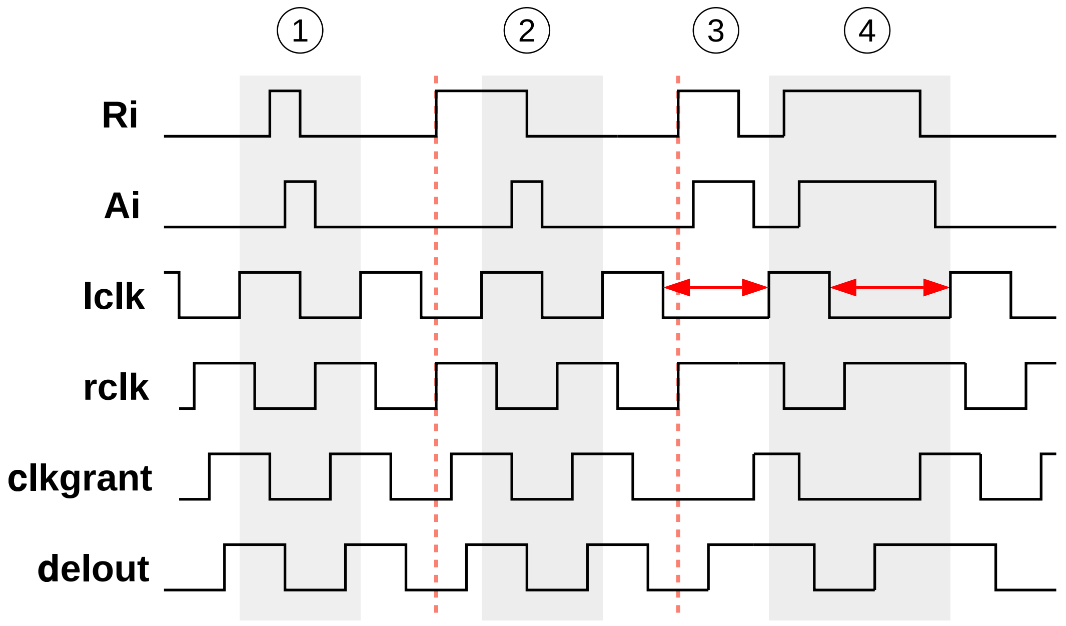


- Asynchronous port controllers allow for fast handshake processing
- Metastability of data is **prevented** by pausing the clock
- No extra latency (synchronizers, FIFOs. . .) introduced
- Wrapper shall be assembled from predesigned elements

Pausable clock generation

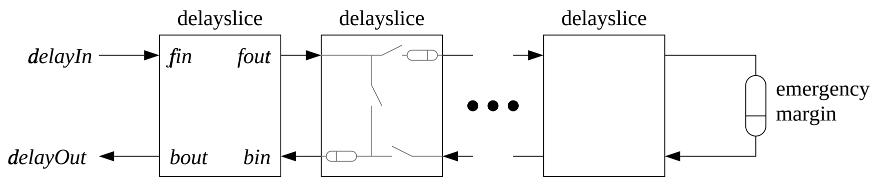
- Ring oscillator for local clock generation
- Arbitration with Mutual Exclusion (ME) elements
- Programmable delayline





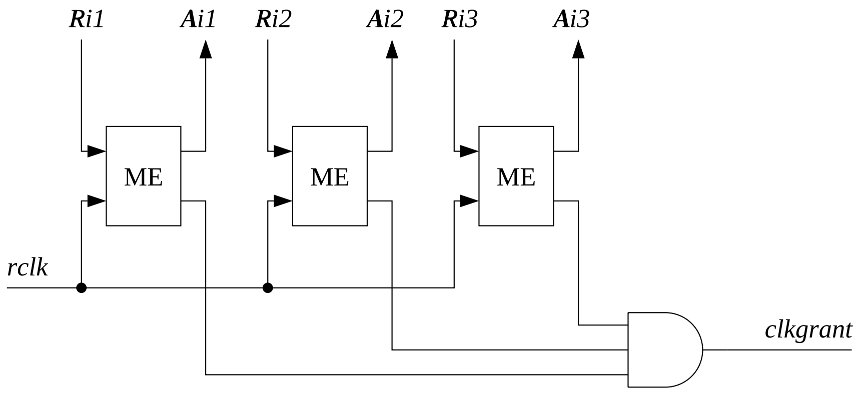
Delayline and Arbitration

Principle of delayline using slices



- every slice can bypass rest of delayline
- delay adjustable over a wide range
- small delay increment ($\approx$ 350ps per slice with $0.25\mu m$ technology)

Multiple Arbitration block



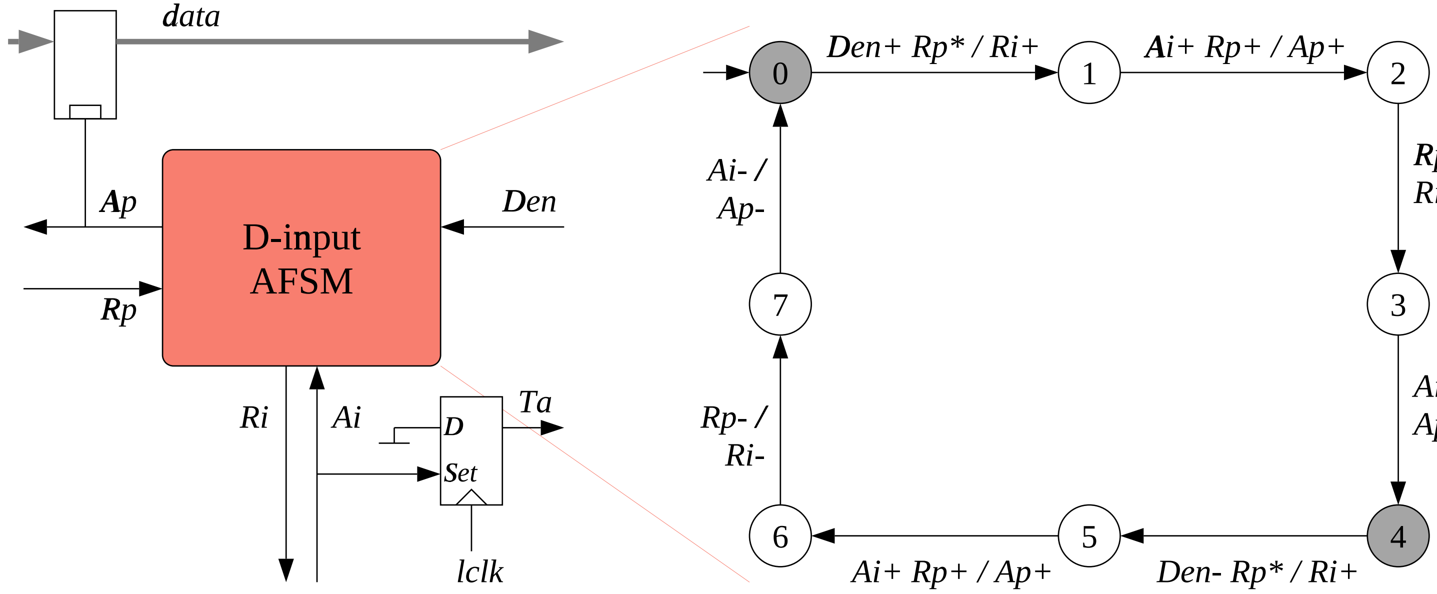
- safely arbitrates between incoming requests and rising edges on rclk
- row of mutual exclusion elements
- scales linearly to multiple ports

GALS building blocks

- Port controller responsible for managing all data transfers
- Each unit is captured by a structural VHDL description
- Asynchronous port controllers achieve cycle times less than 350ps

- A **Poll-type** ports ask for clock stretching only to prevent metastability and ensure data correctness: “proceed while waiting”
- **Demand-type** ports also ensure data integrity but stop the local clock as soon as they are enabled: “sleep while waiting”
- **ROM (LUT) port:** Interface at the ROM port is just a fake delay between *Rp* and *Ap* matching the data delay
- **RAM access:** Essentially a Demand-Out port with bidirectional data transfer (two data vectors in opposite direction controlled by a common handshake pair)

D-input port



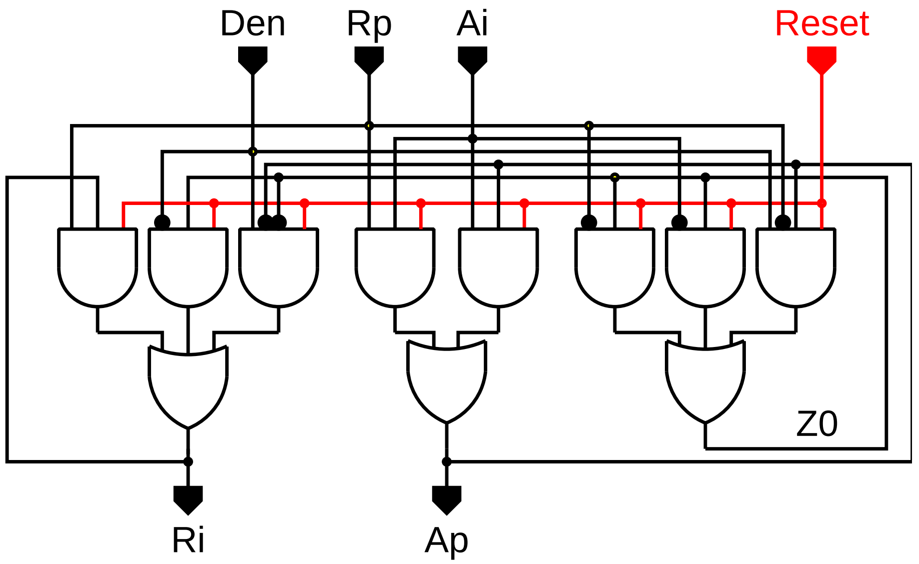
- Performs 2-phase to 4-phase conversion
- Transfer acknowledge (*Ta*) indicates successful transfer
- 3D-tools available for synthesis (by K.Y. Yun)

D-input port (cont.)

Synthesis results

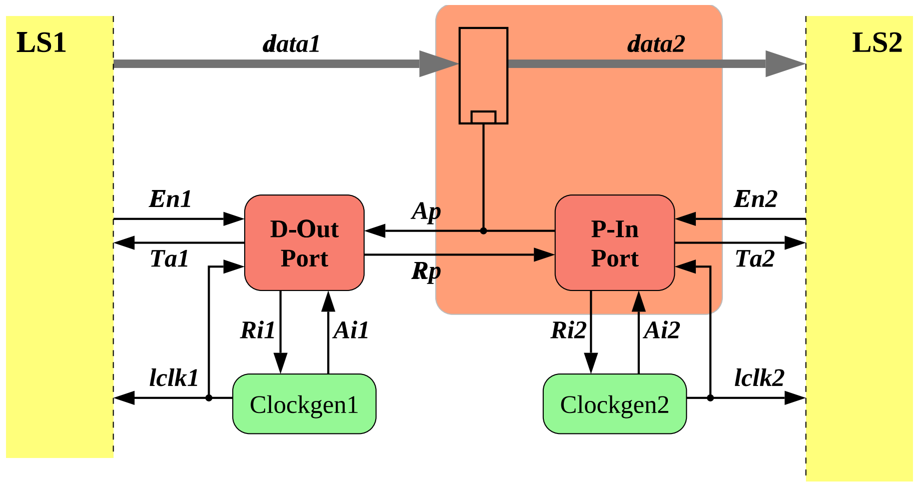
$$Ri = Rp\ Ri + \overline{Den}\ Z0 + Den\ \overline{Ap}\ \overline{Z0}$$
$$Ap = Rp\ Ai + Ai\ Ap$$
$$Z0 = \overline{Rp}\ Z0 + \overline{Ai}\ Z0 + Den\ \overline{Rp}\ Ap$$

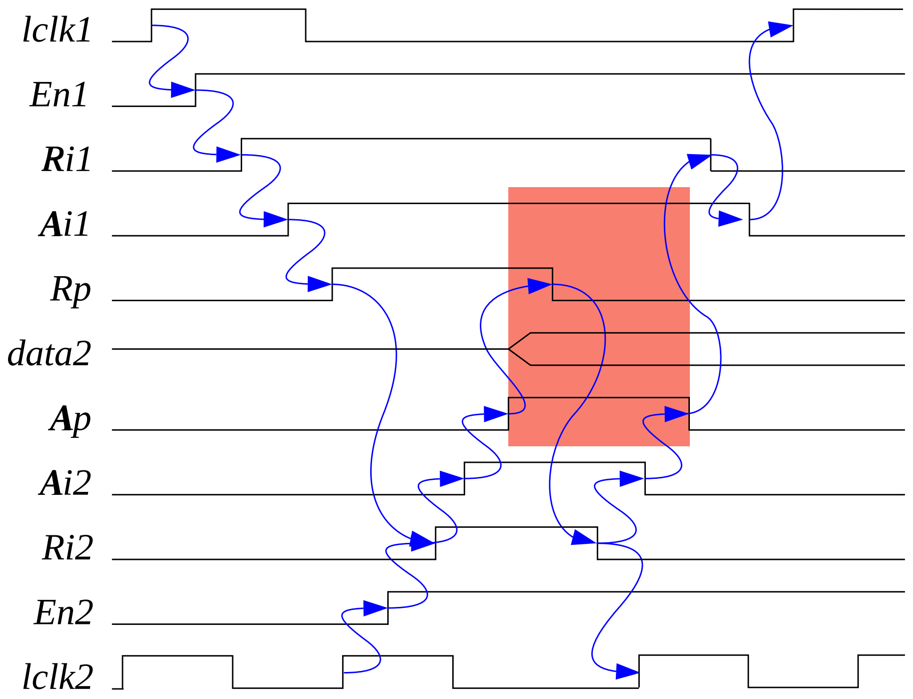
2-level AND-OR Implementation



Compact, fast and hazard-free implementation of async FSM

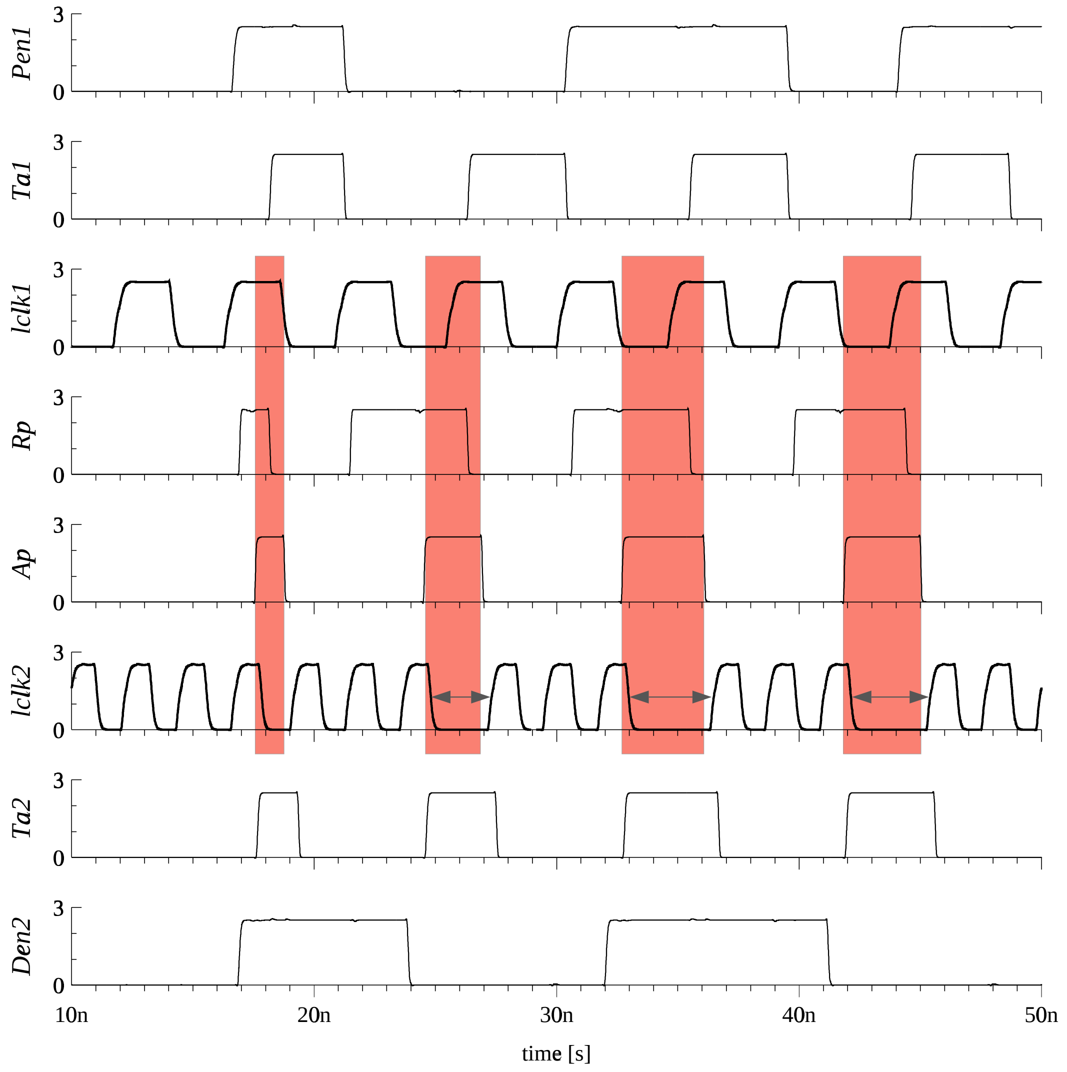
Data transfer mechanism





- The transfer channels all work with rendezvous scheme
- Push channels only, but pull channels would also be possible
- Data latches needed due to undetermined clock relation between the modules

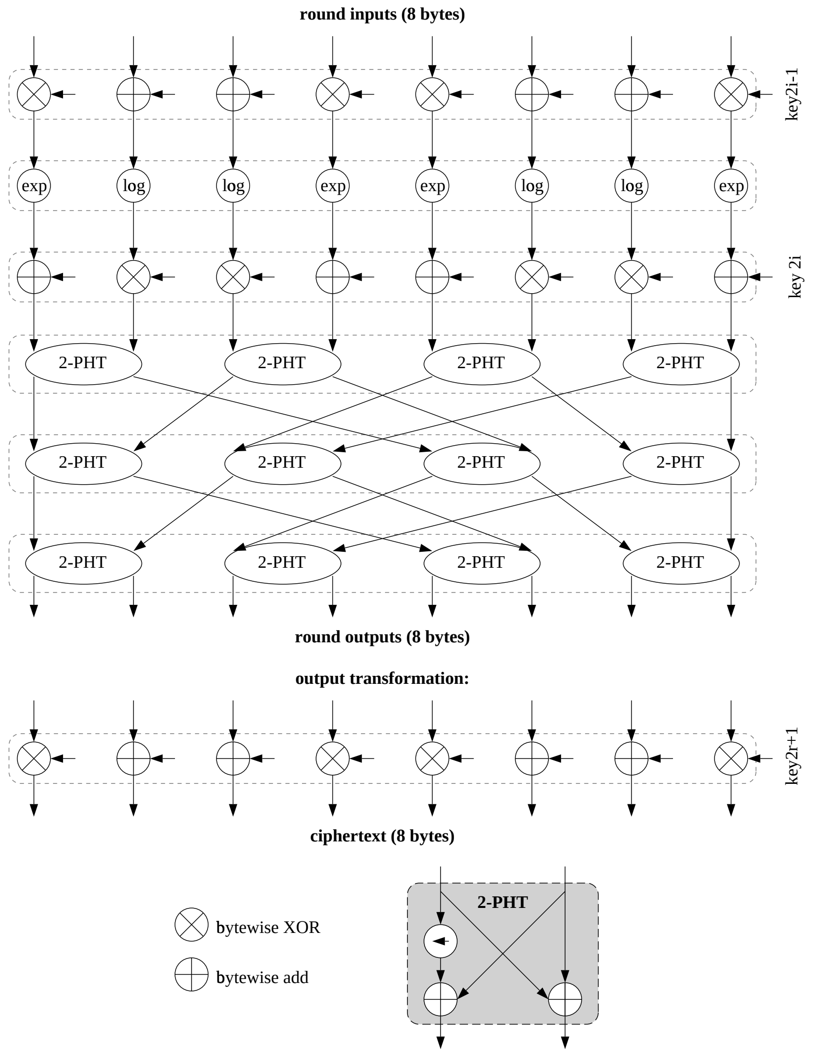
Data channel simulation



P-out to D-in channel:

- Clock stretching infrequent for P-ports
- Fast transfer processing
- Local clocks restart in phase with data

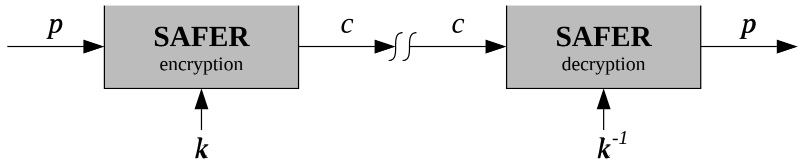
SAFER cryptoalgorithm



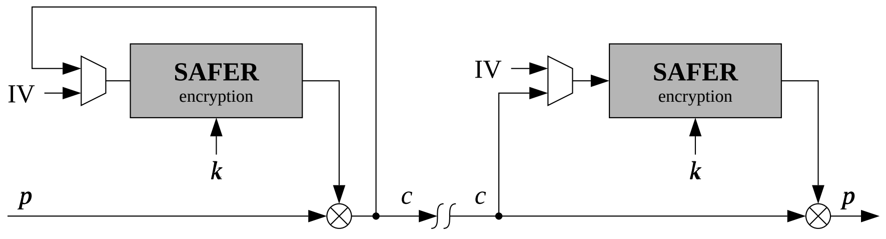
- Secret-key iterated block cipher
- Encryption and decryption slightly different
- Byte oriented: blocks of 8 bytes
- Recommended number of rounds 10 to 12
- Additional input/output transformation
- Comes with variable key lengths (Implemented version: SK-128)

Cryptographic operation modes

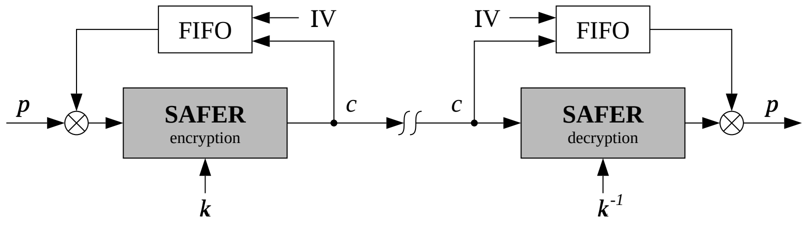
ECB Mode



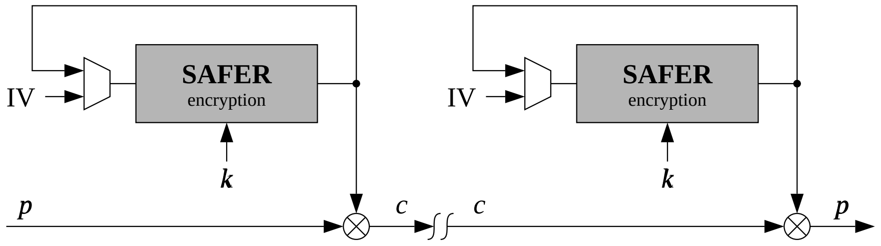
CFB mode



CBC mode

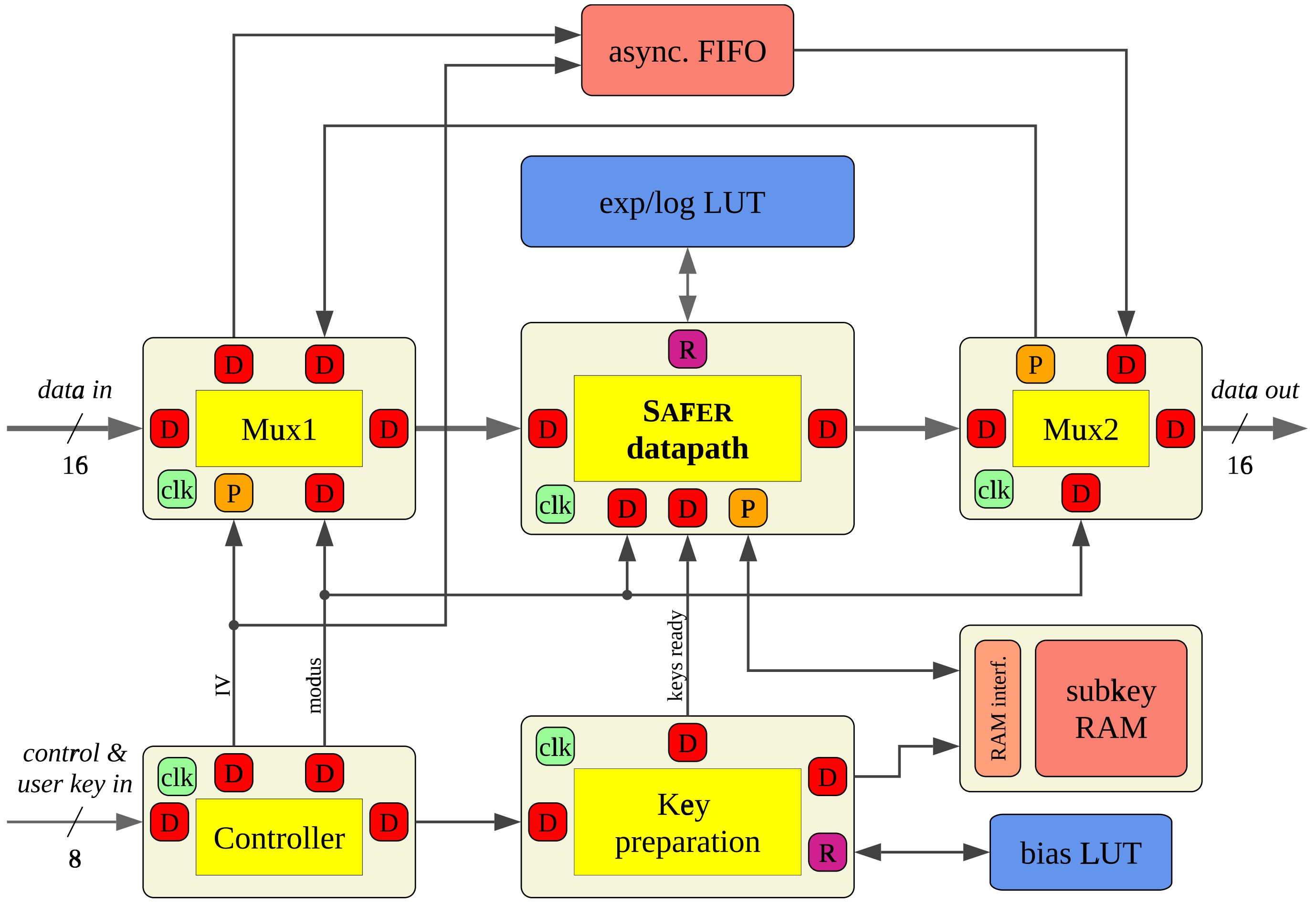


OFB mode



p : Plaintext
c : Ciphertext
k : Key
IV: Initialization Vector

Design Example: MARILYN SAFER cryptochip

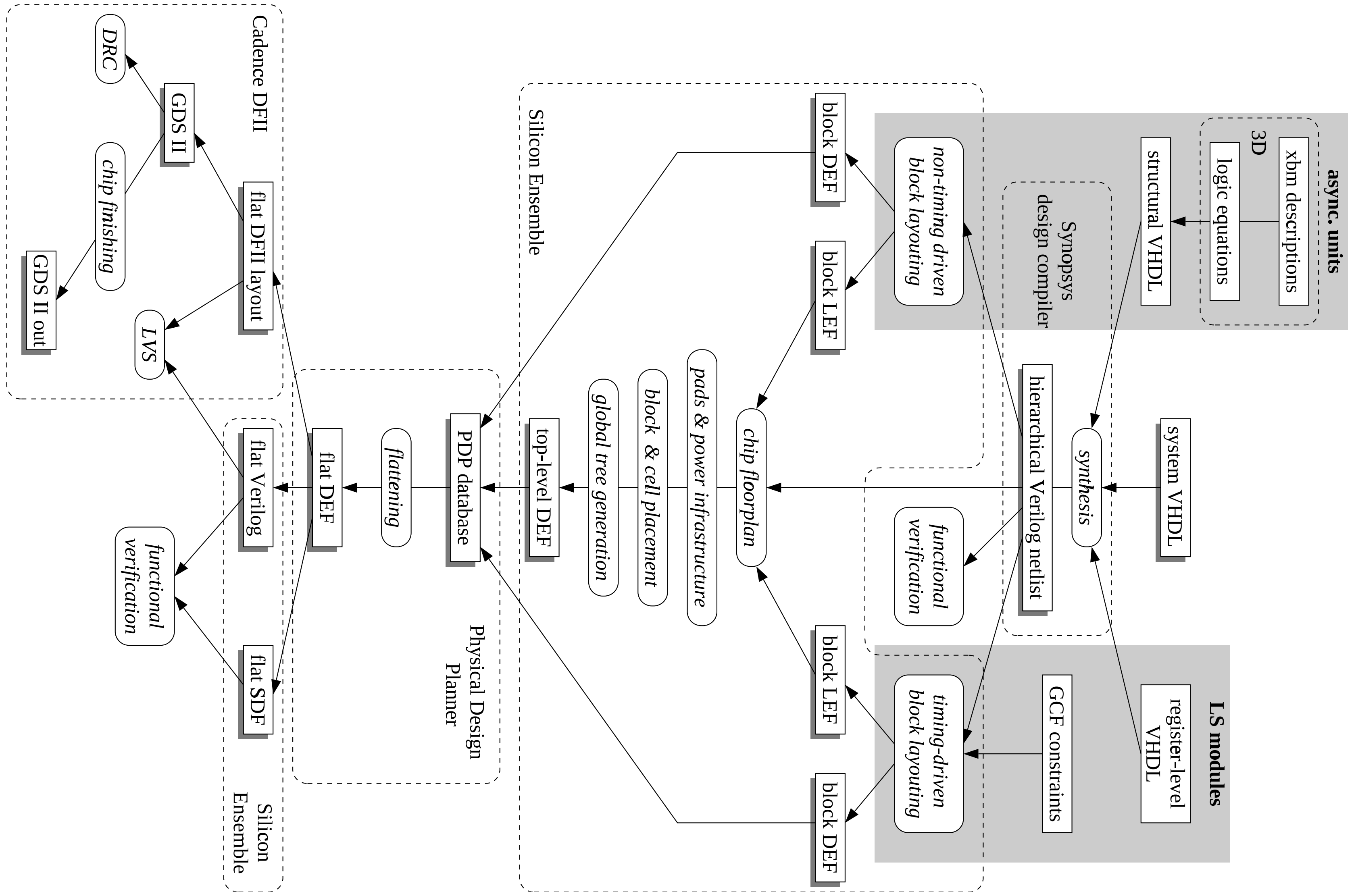


- SAFER block cipher algorithm
- Supports ECB, CBC, CFB, OFB
- Implements both encryption and decryption
- 5 “true” clock domains
- Synchronous counterpart named “MERLIN”

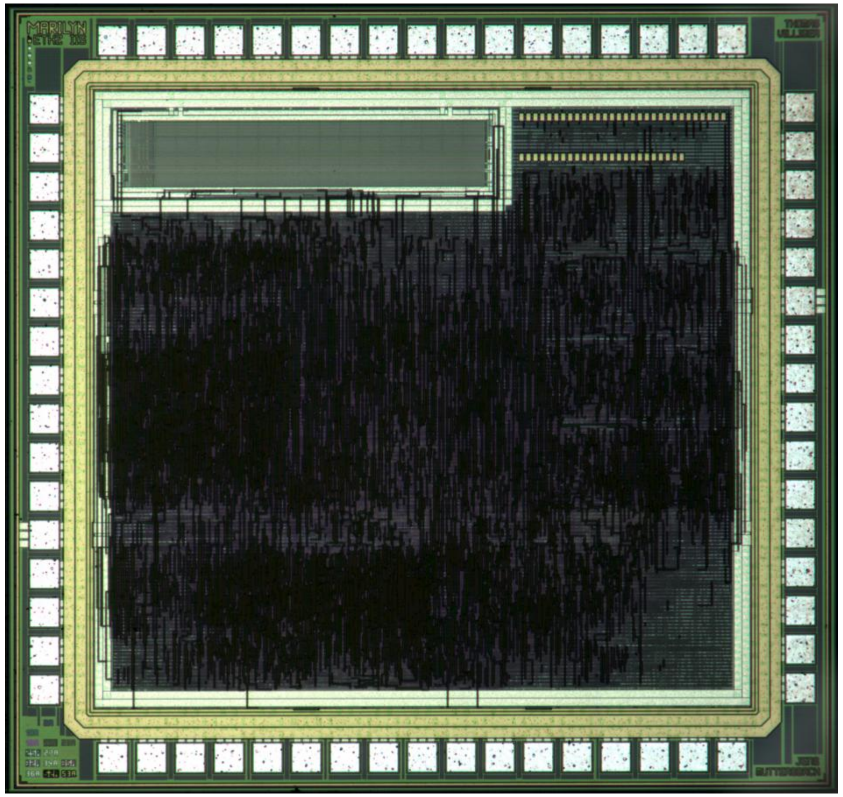
Area figures

	nominal cycle time	(sync.) module $[\mu m^2]$	wrapper area w/o sync $[\mu m^2]$	wrapper area with sync $[\mu m^2]$
controller	2.1ns	38 457	12 501	18 315
key prep	3.4ns	250 299	13 059	18 531
bias ROM	1.2ns	32 805	4 104	4 104
datapath	3.3ns	214 956	28 629	37 863
mux1	2.3ns	150 831	29 628	39 204
mux2	2.2ns	126 747	26 361	34 227
exp/log ROM	1.7ns	291 672	12 708	12 708
subkey RAM	–	237 415	17 784	17 784
async FIFO	–	34 182	–	–
		1 377 364	144 774	182 736
	area	100%	10.5%	13.2%

Design flow


async. units
xbm descriptions
3D
logic equations
structural VHDL
Synopsys
design compiler
synthesis
system VHDL
hierarchical Verilog netlist
functional verification
LS modules
register-level VHDL
GCF constraints
non-timing driven block layouting
timing-driven block layouting
block DEF
block LEF
block LEF
block DEF
Silicon Ensemble
chip floorplan
pads & power infrastructure
block & cell placement
global tree generation
top-level DEF
PDP database
Physical Design Planner
flattening
flat DEF
flat Verilog
flat SDF
Silicon Ensemble
functional verification
LVS
flat DFII layout
GDS II
DRC
chip finishing
GDS II out
Cadence DFII

MARILYN



- Technology:
  0.25μm, 5 metal, CMOS
- Core: 1.7 x 1.7 mm^2
- Die: 2 x 2 mm^2
- 66 pads ($\rightarrow$ JLCC68 package)
- Contains extra testblocks for GALS components
- GALS overhead $\approx$ 10%
- Throuhgput 232Mbit/s at 10 rounds ECB
- Max. throuhgput up to 780Mbit/s (feedthrough)

Results

			MARILYN (GALS)	MERLIN w/ clk-gating	MERLIN w/o clk-gating	GALS benefit
throughput	ECB enc.	10 rounds	232 MBit/s	303 MBit/s		−23%
		12 rounds	194 MBit/s	255 MBit/s		−24%
	CBC dec.	10 rounds	227 MBit/s	303 MBit/s		−25%
		12 rounds	191 MBit/s	255 MBit/s		−25%
energy / MBit		ECB enc. 10 rounds	555 nJ	737 nJ	973 nJ	25%
		CBC dec. 10 rounds	577 nJ	733 nJ	965 nJ	21%

Conclusion

- Complete methodology for GALS architectures developed
- Over 25% energy reduction (energy per MBit throughput)
- Area overhead below 10%
- Throughput up to 25% lower than synchronous version
  (due to datapath not running at full speed, reason currently being investigated)
- GALS building blocks:
  - Consist mainly of technology independent structural VHDL
  - Only mutual exclusion (ME) element requires cell design
- And most important: GALS works on silicon!

Future work

- Improve testability
- Extend the communication schemes beyond point to point links
- Adress system level aspects like deadlock analysis and system partitioning
- Improve tool flow support (hierarchical flow, timing verification, integration of asynchronous tools)
- Possibility to withdraw a data transfer request (bus deferral)
- Power estimation theory
- Finer time-slice resolution of ring-oscillator