FPL2012 22nd International Conference on Field Programmable Logic and Applications

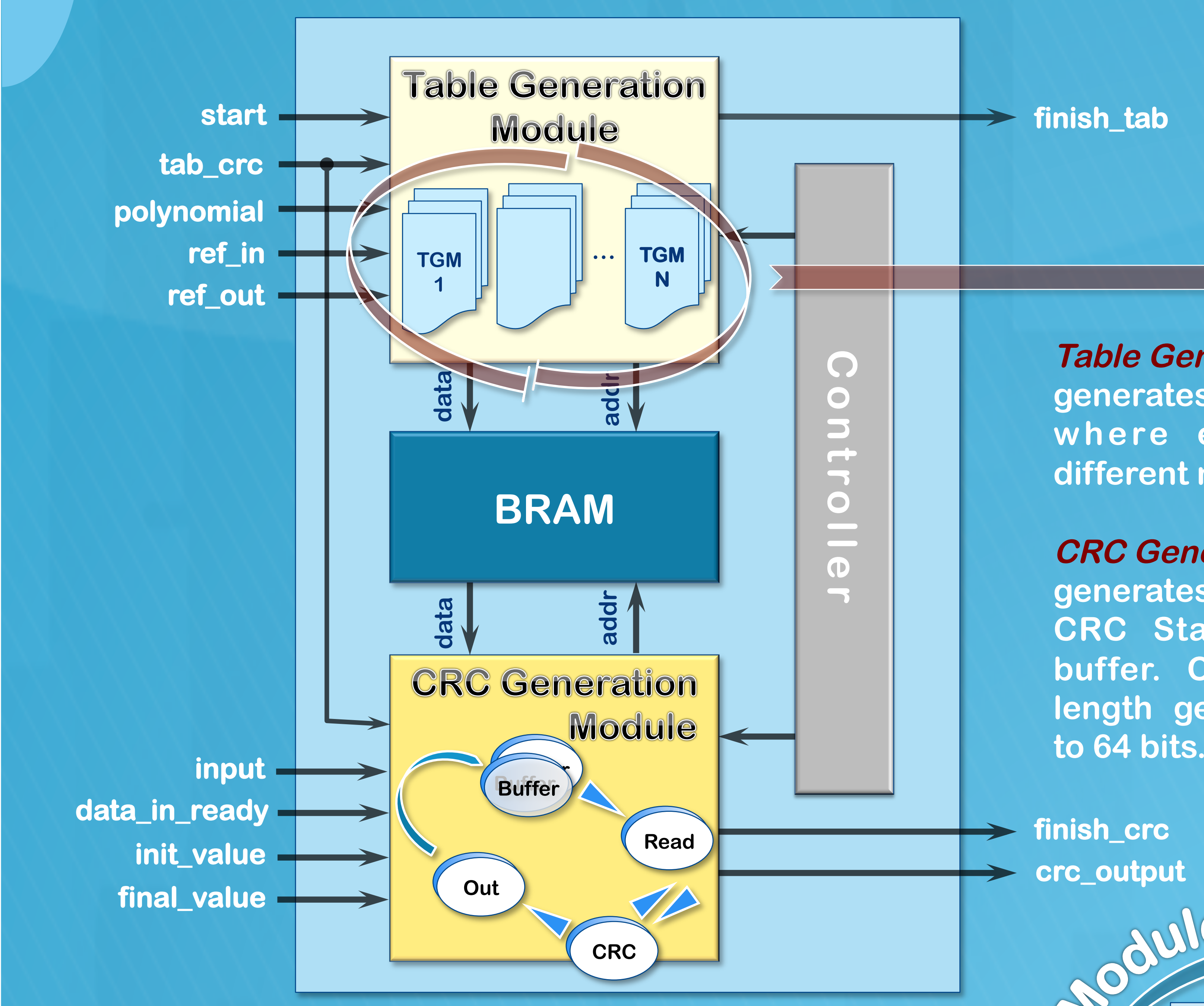
Performance Analysis of Fully-adaptable CRC Accelerators on an FPGA

Amila Akagic and Hideharu Amano

Keio University

Supported by KLL, Keio GCOE

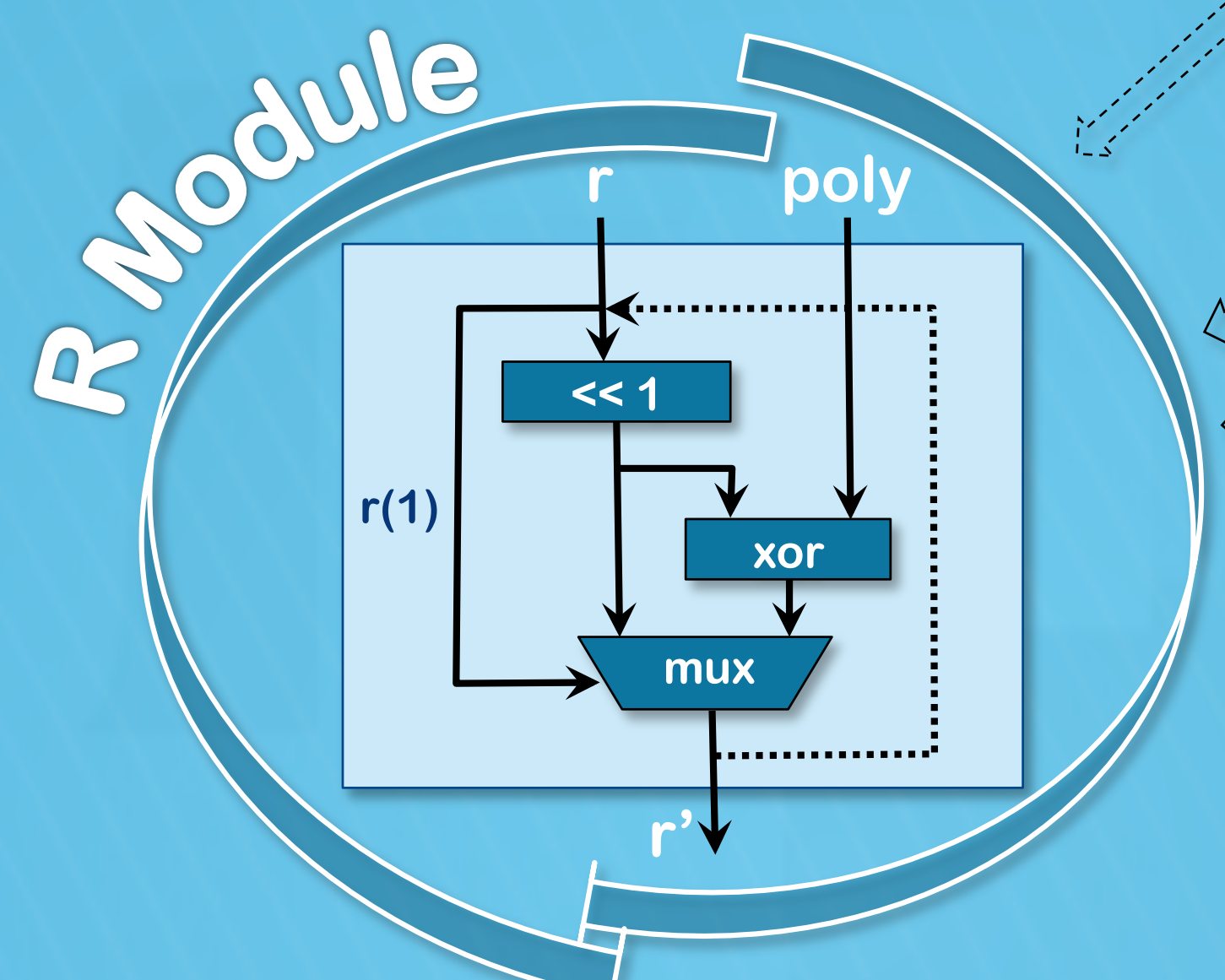
Fully adaptable CRC Accelerator Architecture



- High-speed
- Fully-adaptable
- Up to 64 bits generator polynomial

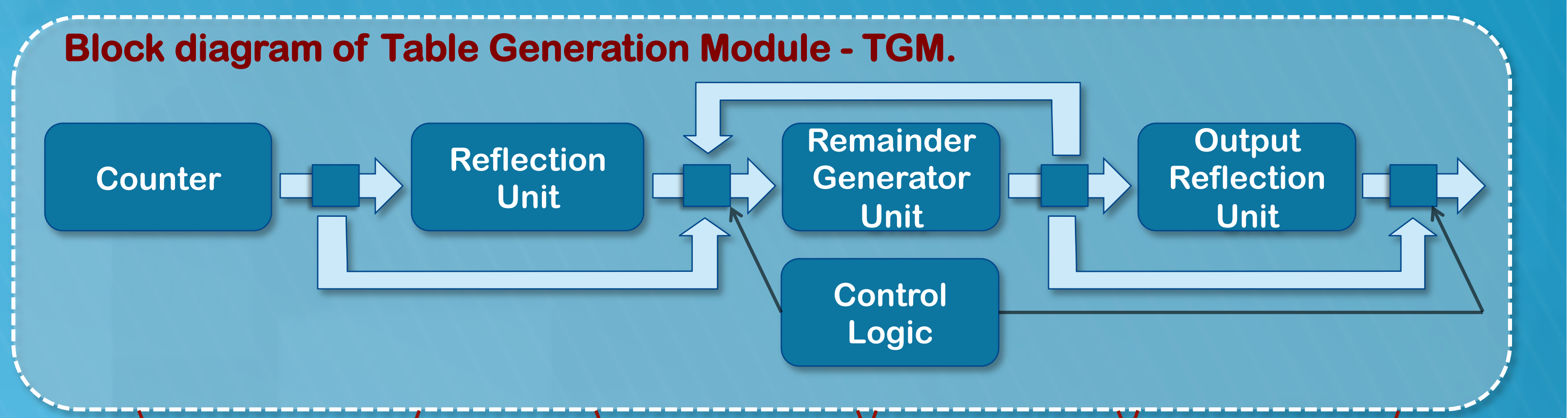
Table Generation Module (TGM) generates remainders for N tables, where each table requires different number of R Modules.

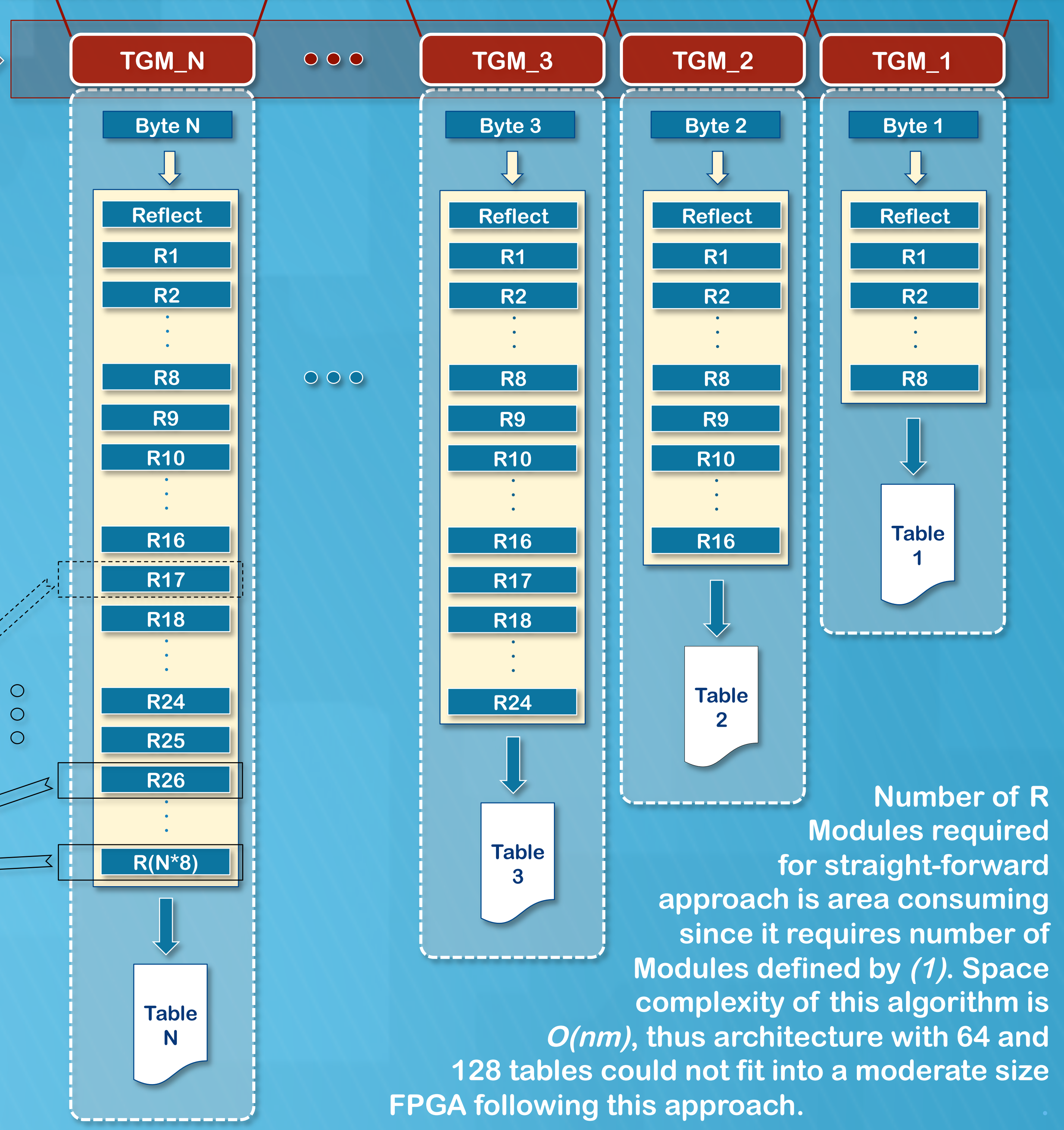
CRC Generation Module (CGM) generates CRC value for any given CRC Standard from the input buffer. CGM supports variable length generator polynomials up to 64 bits.



R Module performs single operation of Remainder Generator Unit. Operations are interdependent from the results of previous operation, thus it is not possible to execute them in parallel.

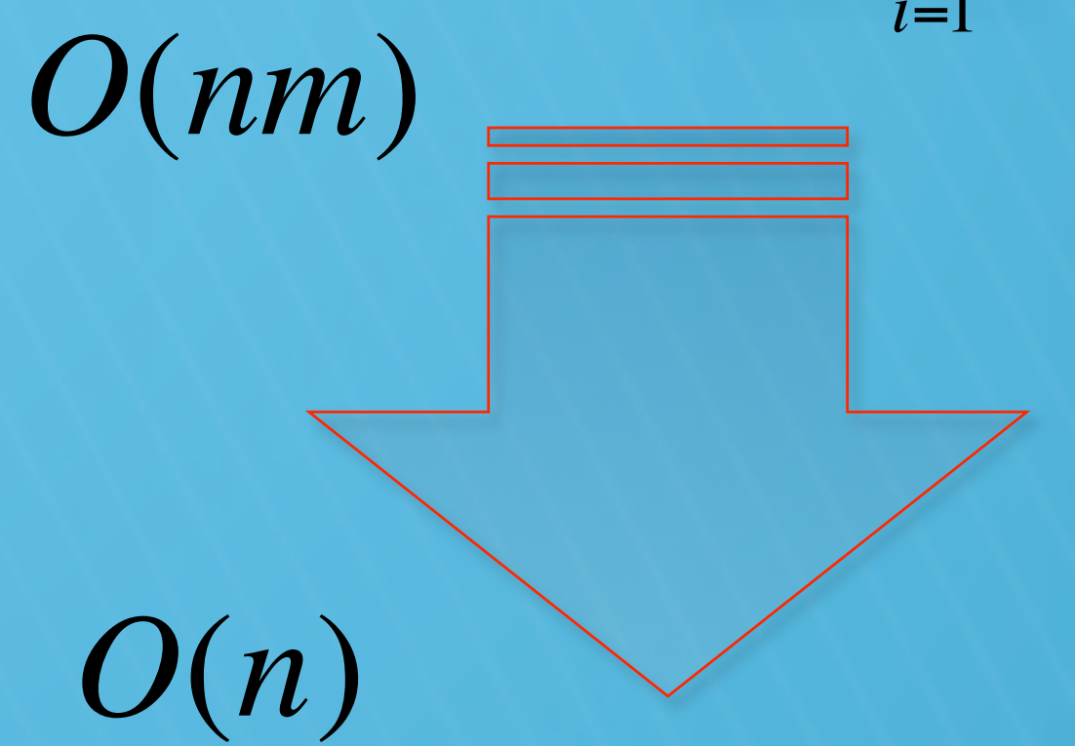
Block diagram of Table Generation Module - TGM.





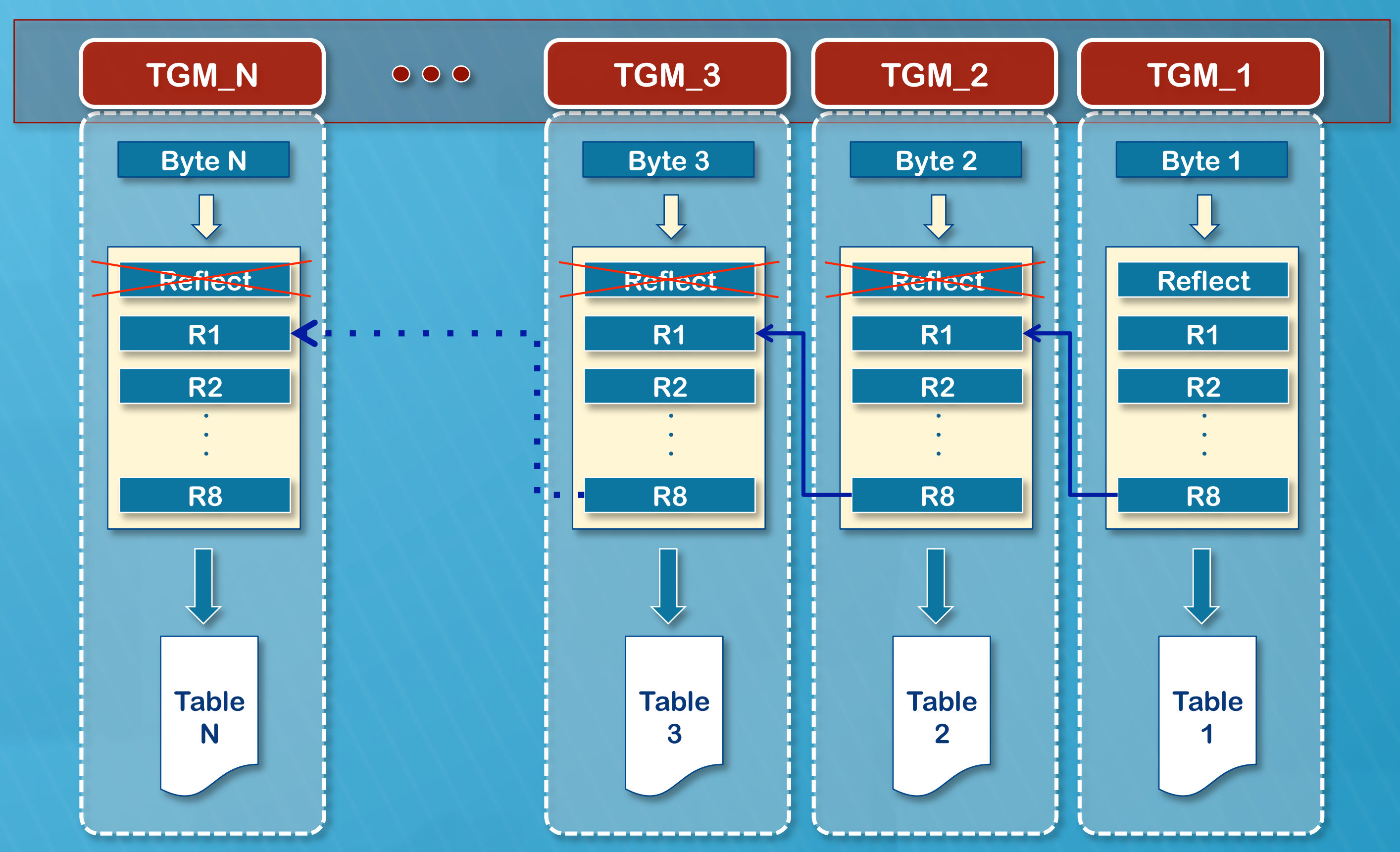
Number of R Modules required for straight-forward approach is area consuming since it requires number of Modules defined by *(1)*. Space complexity of this algorithm is *O(nm)*, thus architecture with 64 and 128 tables could not fit into a moderate size FPGA following this approach.

$$T_{R_N} = \sum_{i=1}^{N} T_{R_i} = slice\_length \times \sum_{i=1}^{N} Offset_i \quad (1)$$



In our ***Overlapped pipelined implementation*** we reduced number of modules to (2) by re-using previous modules. The space complexity is reduced from *O(nm)* to *O(n)*.

$$T_{R_N} = N \times 8 \quad (2)$$

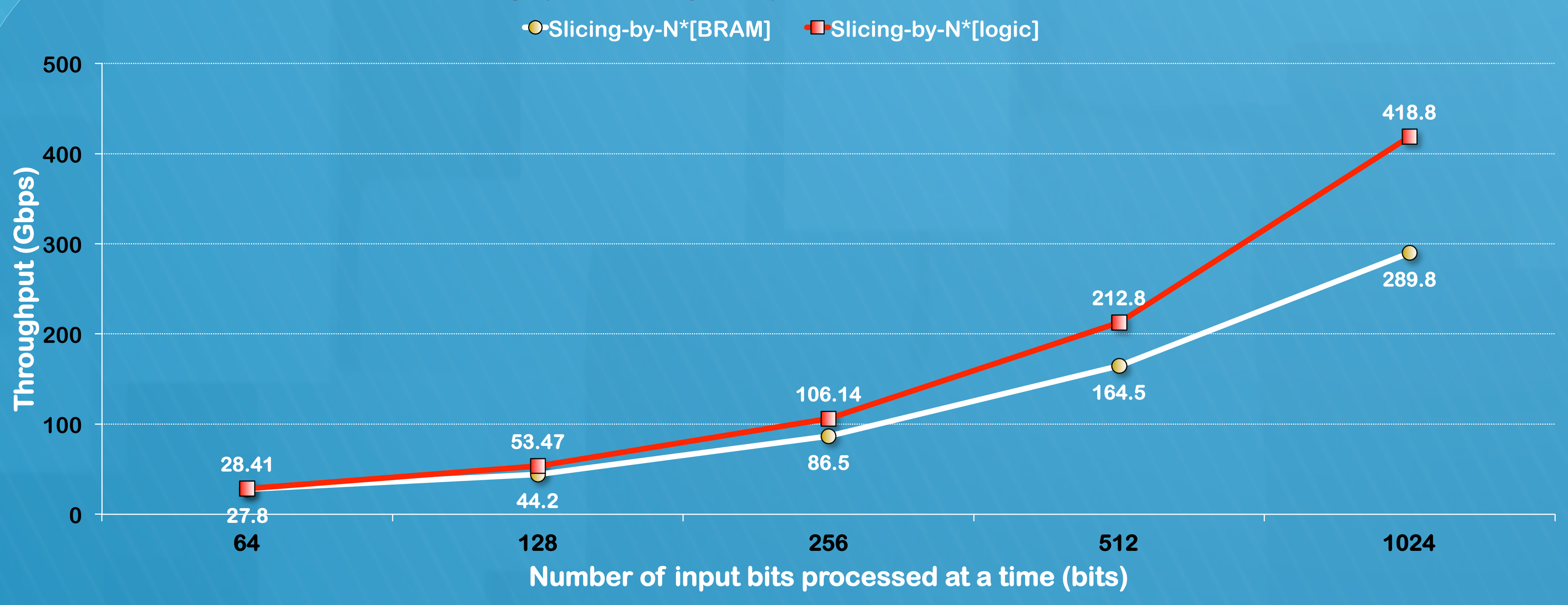


Cyclic Redundancy Checks (CRC)

is an error detection scheme that detects corruption of digital content during data transmission, processing and storage.

Our main contribution is a methodology for building new fully-adaptable and high-throughput architecture for CRC on FPGAs. The implementation was based on table-based algorithm for generating CRCs. The adaptability was achieved with additional circuit for generating remainder values for variable width generator polynomials up to 64 bits.

Throughput of fully adaptable CRC Accelerators



In order to show scalability of the accelerator we implemented five versions of the accelerator, capable of processing different input data widths, ranging from 64 to 1024 bits at a time. Our accelerators are 1.65 to 31.64x faster than related work, depending on the data-path's width.

On the Xilinx Virtex 6 LX550T FPGA board **they occupy between 1-2% area** to produce maximum of 289.8Gbps at 283.1MHz if BRAM is deployed, **or between 1.6 - 14%** of area for 418.8Gbps at 408.9MHz if tables are implemented in logic.

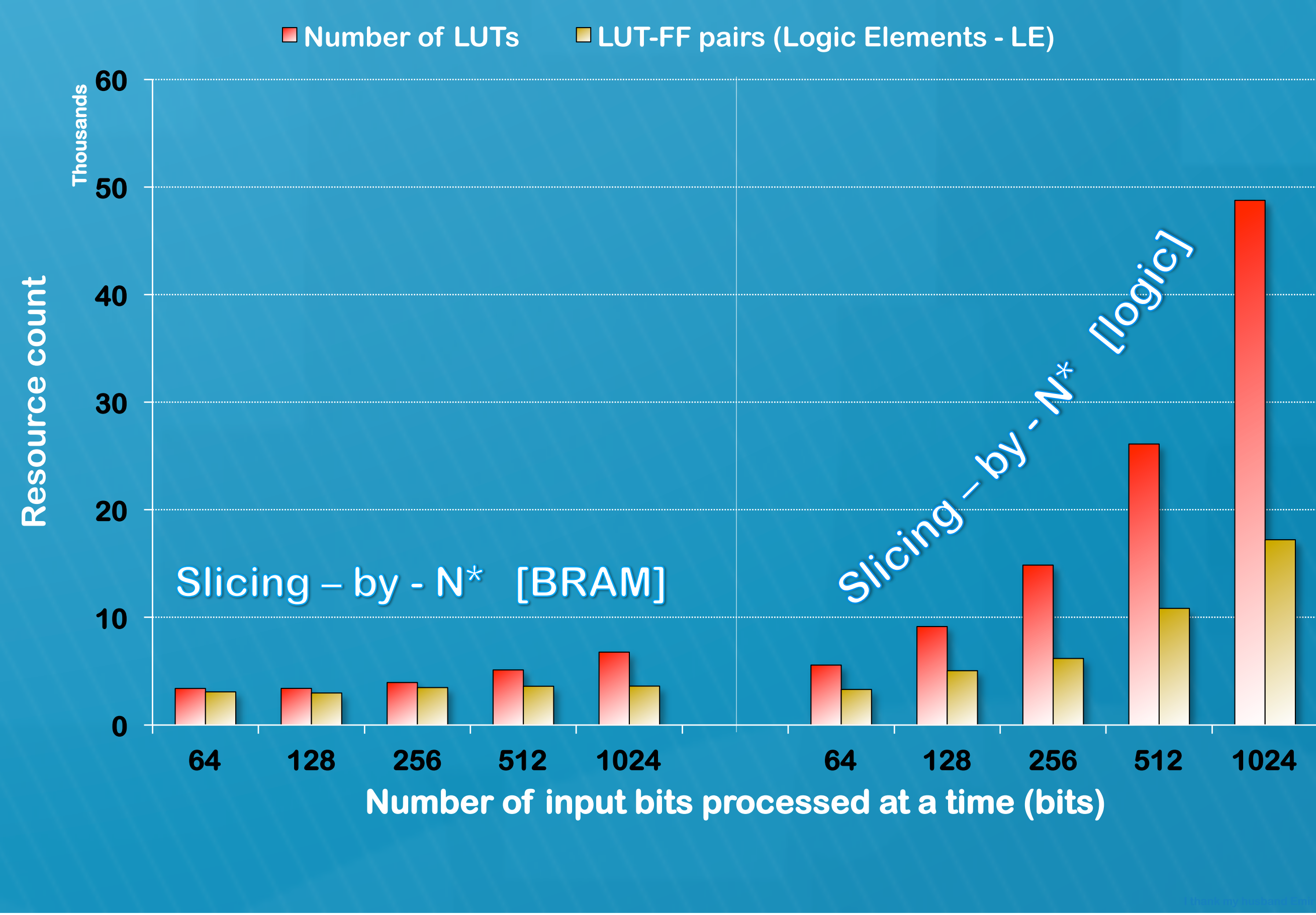
Algorithm	Tables	Clock Cycles	BRAM (μs)	Logic (μs)
Slicing-by-8*	8	320	.9	.72
Slicing-by-16*	16	384	1.11	.92
Slicing-by-32*	32	512	1.52	1.23
Slicing-by-64*	64	768	2.40	1.85
Slicing-by-128*	128	1280	4.52	3.13

Time required for re-generation of a specific number of tables. Re-generation is required only when CRC standard is changed.

The results of the implementation in logic show significant increase in resource utilization but critical path is also significantly decreased since BRAM is not part of the critical path.

Throughput of implementation in logic is up to 31% higher than BRAM implementation, with maximum throughput reaching 418.8Gbps.

Resource utilization of Slicing-by-N* algorithms on Xilinx Virtex 6 LX550T


Number of LUTs
LUT-FF pairs (Logic Elements - LE)
Thousands
Resource count
Slicing – by - N* [BRAM]
Slicing – by - N* [logic]
Number of input bits processed at a time (bits)