Selection of an Error-Correcting Code for FPGA-based Physical Unclonable Functions

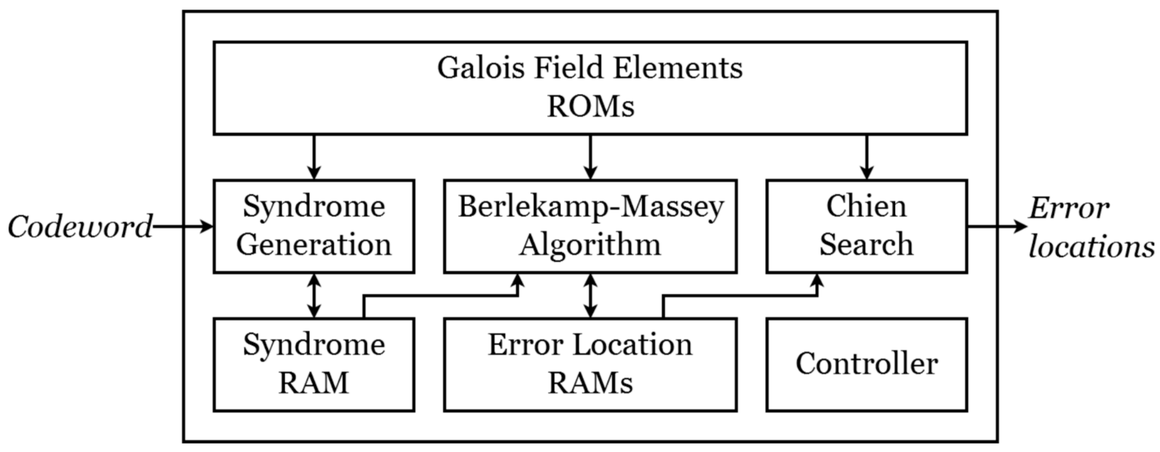
Brian Jarvis and Kris Gaj
Cryptographic Engineering Research Group (CERG), ECE Department, George Mason University, Fairfax, VA USA
http://cryptography.gmu.edu

Motivation

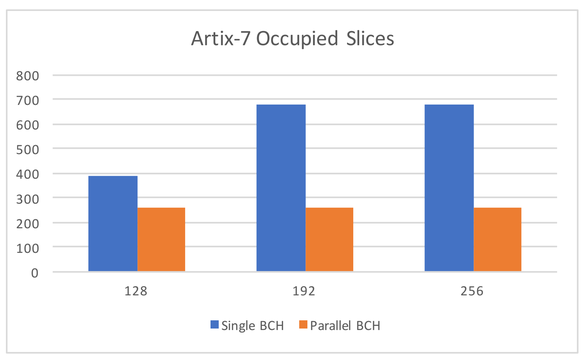
- Fuzzy Extractors are used to improve reproducibility of a PUF
- An error correcting code is needed for the fuzzy extractor to work
- Single codes which yield sufficient entropy and probability of failure have high area utilization when implemented in an FPGA
- Explore parallel codes and convolutional codes as an alternative

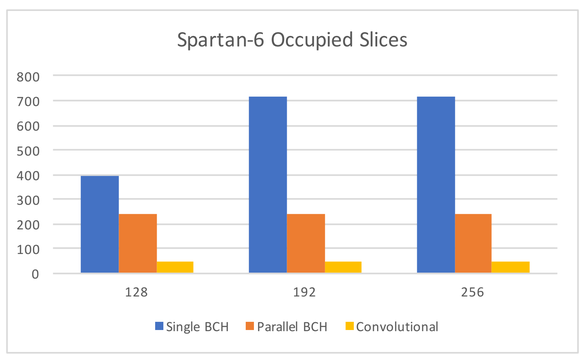
Design Methodology

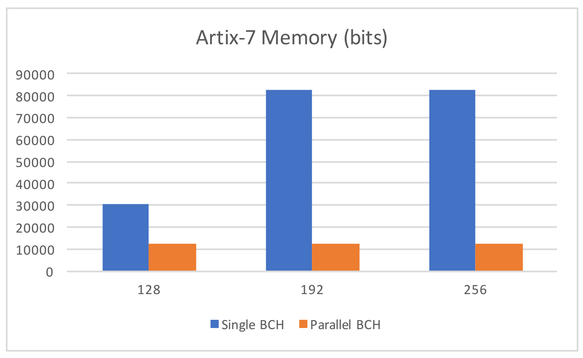
- BCH decoder implemented for Spartan-6 and Artix-7 FPGAs
- Generic design allows any BCH parameters to be configured
- BCH decoding algorithm consists of three high level subcomponents: syndrome generation, Berlekamp-Massey algorithm, Chien search.
- Circuit produces a stream of error locations from an input codeword

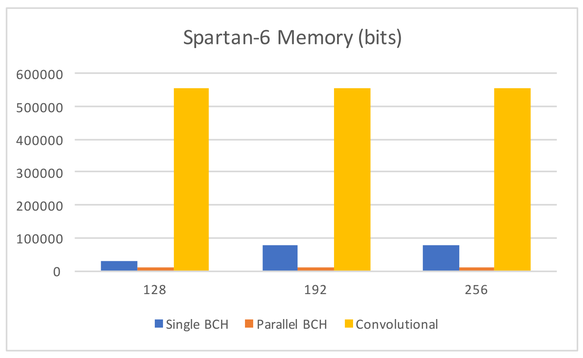


- Convolutional decoder referenced from Hiller et al. SeeSaw Viterbi decoder









- Parallel codes offer attractive alternative, however require a large number of PUF bits
- Convolutional codes are area efficient and have small PUF size requirements
- Convolutional code memory requirements are substantial, but could be realized using block RAMs where available

ECC Selection

- All BCH codes with codeword size 1023 or less were analyzed
- Remaining entropy for each code calculated according to the following equation:

$$H = \min[\rho * n - (n - k), 0]$$

ρ = entropy density = 0.85
n = ECC codeword size
k = ECC message size

- For all codes which produce at least 128 bits of remaining entropy, probability of failure evaluated using the following equation:

$$p_{fail} = 1 - \sum_{k=0}^{t} \binom{n}{k} {p_e}^k (1 - p_e)^{n-k}$$

p_e = probability of input bit error = 0.03
t = error correcting capability

- Codes selected which produce sufficient leftover entropy and have a probability of failure less than 1E-6

ECC REQUIREMENTS USING A SINGLE BCH CODE

ν	n_B	k_B	d_{min_B}	t_B	H_B	p_{fail_B}	n
128	511	238	75	37	161.35	4.6E-7	511
192	1023	483	121	60	329.55	5.8E-7	1023
256	1023	483	121	60	329.55	5.8E-7	1023

ECC REQUIREMENTS USING PARALLEL BCH CODE WITH $n_P = 127$, $k_P = 22$, $d_{min_P} = 47$, $t_P = 23$

ν	x_P	H_P	p_{fail_P}	n
128	44	129.8	3.03E-11	5,588
192	66	194.7	4.54E-11	8,382
256	87	256.7	5.99E-11	11,049

ECC REQUIREMENTS USING CONVOLUTIONAL CODE WITH $n_C = 2$, $k_C = 1$, $K_C = 12$, $A_{d_{free_C}} = 14$

ν	y_C	H	p_{fail_C}	n
128	183	128.1	6.02E-7	366
192	275	192.5	6.02E-7	550
256	366	256.2	6.02E-7	732

Conclusions

- PUF designer need not settle for a single monolithic block code
- Parallel codes and convolutional codes offer an attractive alternative due to both requiring significantly fewer area resources
- Parallel codes may place too large a burden on PUF size requirements
- Convolutional codes require far fewer PUF bits, however have a large area requirement
- If block RAM can be used, convolutional codes offer the most attractive solution