NORX v2.0
Hardware Architecture

Michael Muehlberghuber

January 26, 2016

Abstract. This document contains a brief summary of the NORX hardware implementation.[1] The goal in mind during the development of the architecture was to achieve a throughput of 100 Gbps on a 65 nm ASIC technology. Our architecture solely comprises NORX64-4-1, since this is the recommended version of the authors. Version 2.0 of the NORX specification was used as a reference for this design.

1 Architecture Overview

Figure 1 provides an overview of the developed NORX64-4-1 hardware architecture. Its major components are the eight instances of the NORX G function. Originally, the

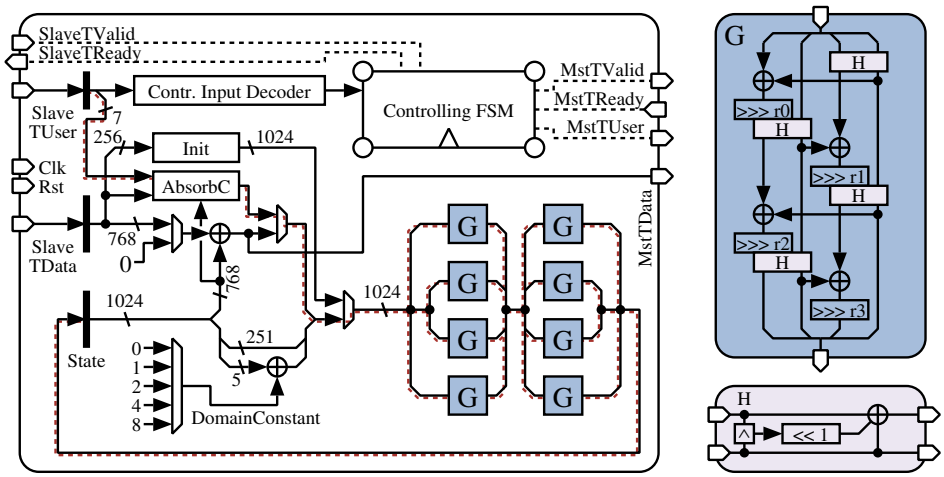

Figure 1: NORX64-4-1 architecture overview

[1] https://github.com/norx/norx-hw

architecture was developed for NORX v1.1. For the submission of the second CAESAR competition round, the authors of NORX adapted the rate/capacity ratio. Therefore, the hardware architecture was adapted accordingly.

2 Features

The NORX hardware architecture ...

- ... supports both encryption and decryption of the recommended version of NORX v2.0 (i.e., NORX64-4-1).
- ... takes as input blocks, each of which has the same size as the bitrate (i.e., 768 bits for NORX64-4-1 v2.0).
- ... operates on the byte level, i.e., it can process both full as well as non-full input blocks.
- ... can read inputs and write outputs with the use of two AXI-4 Stream Protocols.

3 I/O Interface

Basically, two AXI4-Stream protocols are used to communicate with the NORX architecture. For the input data, the `SlaveTUser_SI` signal is used to describe the incoming data (provided via the `SlaveTData_DI` signal). While bits `14 downto 8` of the `SlaveTUser_SI` signal are used to describe the number of valid input bytes, the lower eight bits of the `SlaveTUser_SI` signal are used to actually determine what type of input data is provided. Table 1 describes the possible values the lower byte of the `SlaveTUser_SI` signal can have, encoded as an integer value.

Table 1: Values of the lower byte of the signal `SlaveTUser_SI` and its meanings to the NORX architecture.

Value	Description
0	Nothing to be done at all.
1	Indicates that both a new nonce (`TDataSrc_(767:640)`) and an new cipherkey (`TDataSrc_(639:384)`) have been applied at the data input and the next incoming data block will be a header block (i.e., associated data).
2	Indicates that both a new nonce (`TDataSrc_(767:640)`) and an new cipherkey (`TDataSrc_(639:384)`) have been applied at the data input and the next incoming data block will be a payload block.
3	Indicates that a new AD block is applied and the next input block will also be an AD block.

4	Indicates that a new AD block is applied and the next input block will be a payload block (i.e., plaintext or ciphertext).
5	Indicates that a new AD block is applied and next the authentication tag should be computed (i.e., no data can be obtained from the source as the NORX architecture will be busy with computing the tag).
6	Indicates that a new plaintext block is applied and that the next input block will also be a plaintext block.
7	Indicates that a new plaintext block is applied and that the next input block will be a trailer block.
8	Indicates that a new plaintext block is applied and that next the authentication tag should be computed.
9	Indicates that a new ciphertext block is applied and that the next input block will also be a ciphertext block.
10	Indicates that a new ciphertext block is applied and that the next input block will be a trailer block.
11	Indicates that a new ciphertext block is applied and that next the authentication tag should be computed.
12	Indicates that a new trailer block is applied and that the next input block is another trailer block.
13	Indicates that a new trailer block is applied and that next that authentication tag should be computed.

Table 2 summarizes the descriptions of Table 1 in brief. For the control signal of the output interface (`MasterTUser_SO`) there are only two possible valid values. While a decimal '1' indicates that there is a valid message block (i.e., ciphertext for encryption respectively plaintext for decryption), a '2' indicates that the provided block on the output data signal (`MasterTData_DO`) contains the authentication tag. As long as there is no valid data on the output data signal, `MasterTUser_SO` remains zero.

4 Potential Architecture Changes

- If bad I/O timings are not considered to be a problem, the input registers may be removed. Note that with such a modification the area can be decreased a little bit (i.e., by saving appr. 800 bits of memory). However, with such a modification the input-to-state delay will run through the G functions and may violate timing constraints when putting the architecture into a larger system.
- If data is only provided on the block-level (i.e., 768 bits), the `absorbCiphertext` module can be removed, which should save another 5–10% of area and also shortens the critical path.

Table 2: Summary of the values of the input control signal `TUserSlave_SI` and its meanings to the NORX architecture.

Value	**Current Phase**		**Next Phase**
	Data Type[†]	**Data Width**[‡]	
0	-	-	-
1	N/CK	384	AD
2	N/CK	384	PT/CT
3	AD	768	AD
4	AD	768	PT/CT
5	AD	768	Tag Gen.
6	PT	768	PT
7	PT	768	TR
8	PT	768	Tag Gen.
9	CT	768	CT
10	CT	768	TR
11	CT	768	Tag Gen.
12	TR	768	TR
13	TR	768	Tag Gen.

[†] N... Nonce, CK... Cipherkey, AD... Associated data (header), PT... Plaintext, CT... Ciphertext, TR... Trailer data

[‡] The actual width of data transferred via the input data signal. Note that for non-full blocks, this width becomes respectively smaller.

- Currently, no countermeasures against any kind of side-channel attacks have been considered. Especially processing the nonce and the cipherkey using a single input block might lead to problems since an attacker could choose the nonce. However, as for this evaluation the high throughput was our major goal, side-channel countermeasures must be considered as future work.
- With the current design, the user applying data to the NORX architecture, must know in advance what type of input data will be applied next (respectively if the authentication tag should be computed). In order to get around this prerequisite, a second input register stage could be added. Since thereby the area of the design would increase accordingly, we consider this approach as another requirement.