
password

SHA1 Decoder

Niklas Hjern, Erik Hogeman & Jonas Vistrand

Lund University Faculty of Engineering

hjern.niklas@gmail.com
Erik.Hogeman@gmail.com
ael09jvi@student.lu.se

October 18, 2013

Introduction

- Decode sha1 hashes with hardware acceleration
- Bruteforce
- Dictionary
- Input: keyboard
- Output: VGA display

System architecture

- Nexys3 board with a Spartan 6 FPGA
- Single microblaze system
- PLB bus
- 32 kB bram
- 88 MHz clock frequency

IP cores

- Xilinx IP cores
  - Ps2 keyboard controller
  - Numonyx pcm
- Custom IP cores
  - SHA1 hasher
  - String generator
  - VGA controller

Architecture overview (old)

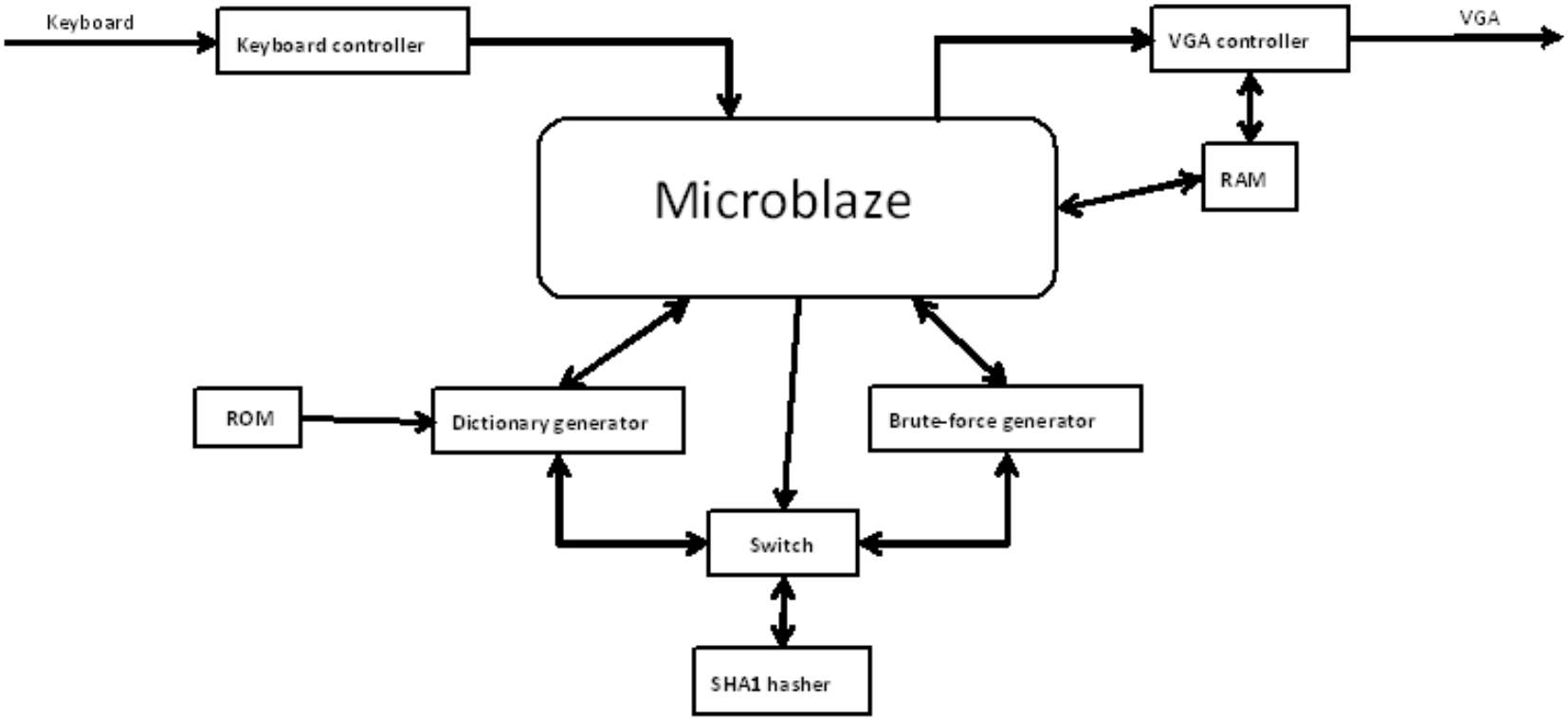

Figure: Old architecture

Architecture overview (new)

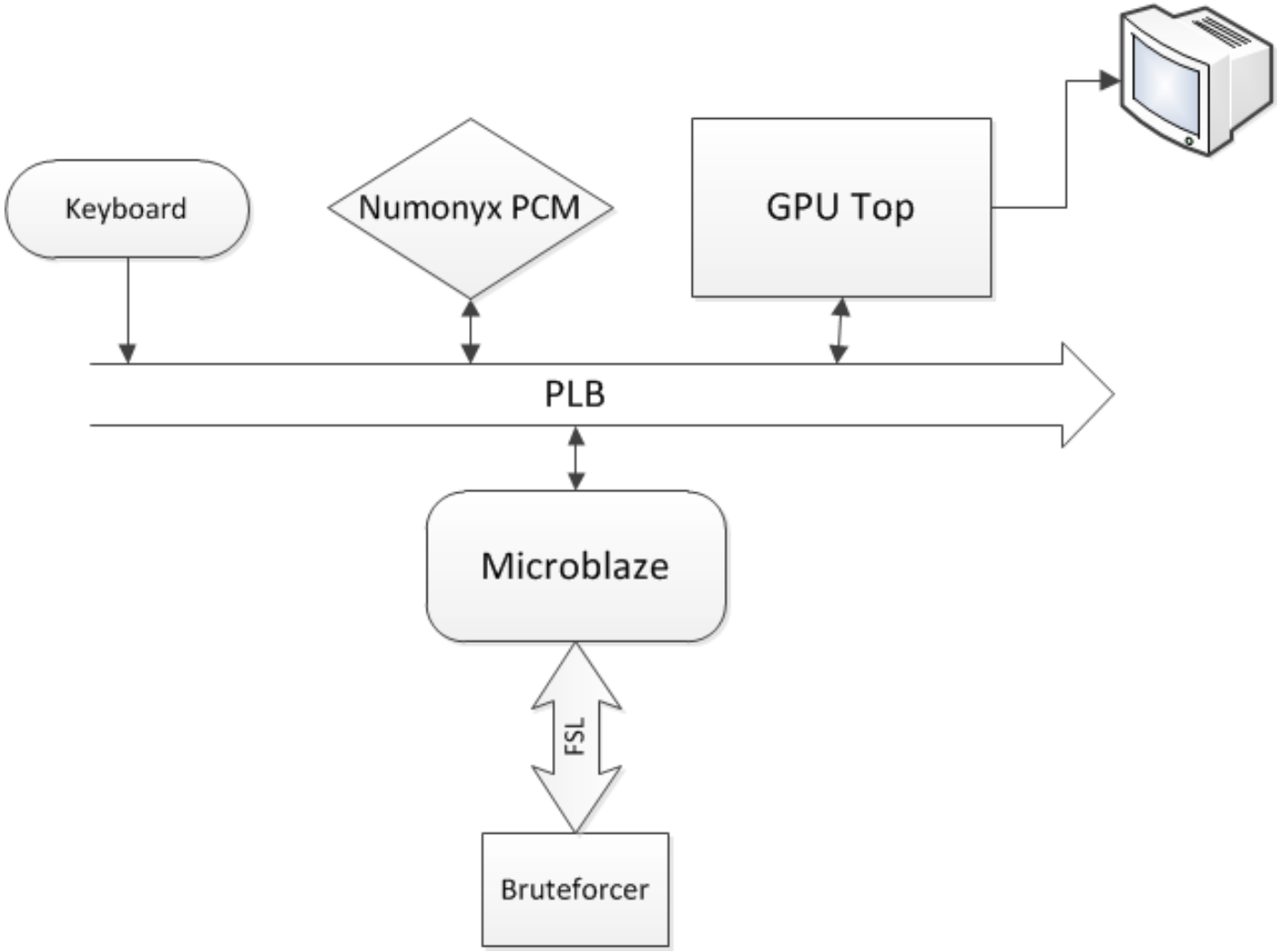

Figure: New architecture

SHA1 algorithm

- Merkle-Damgård construction, runs for 80 rounds.
- Can be split into two distinct parts
  - First part pads the input string to 512 bits and splits it into 16 blocks, then extends these to 80 blocks.
  - Second part is the main loop. Each iteration, one of the 80 blocks is used to compute some new intermediate values.

SHA1 hasher

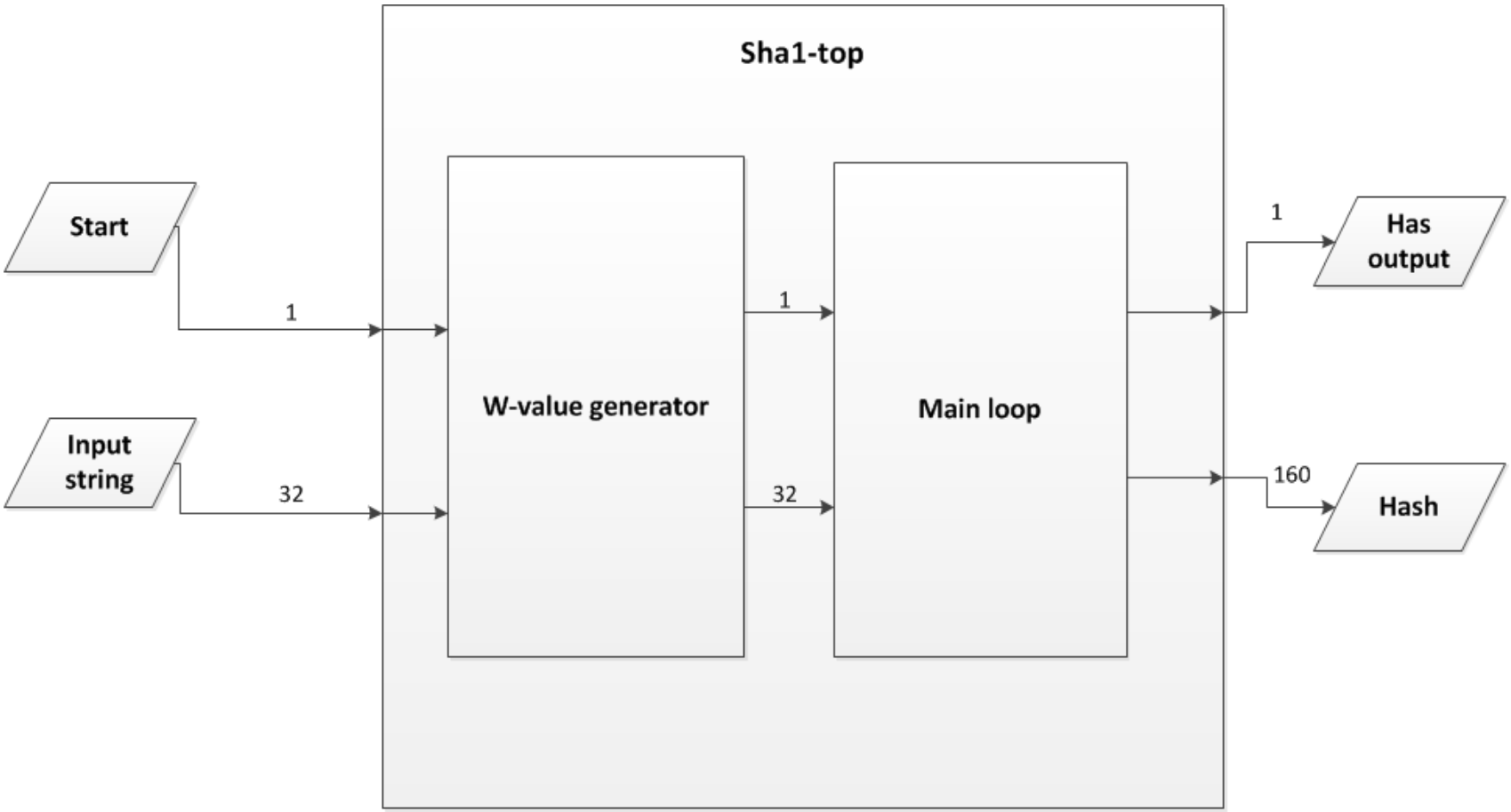

Figure: SHA1 hasher, first version

SHA1 hasher

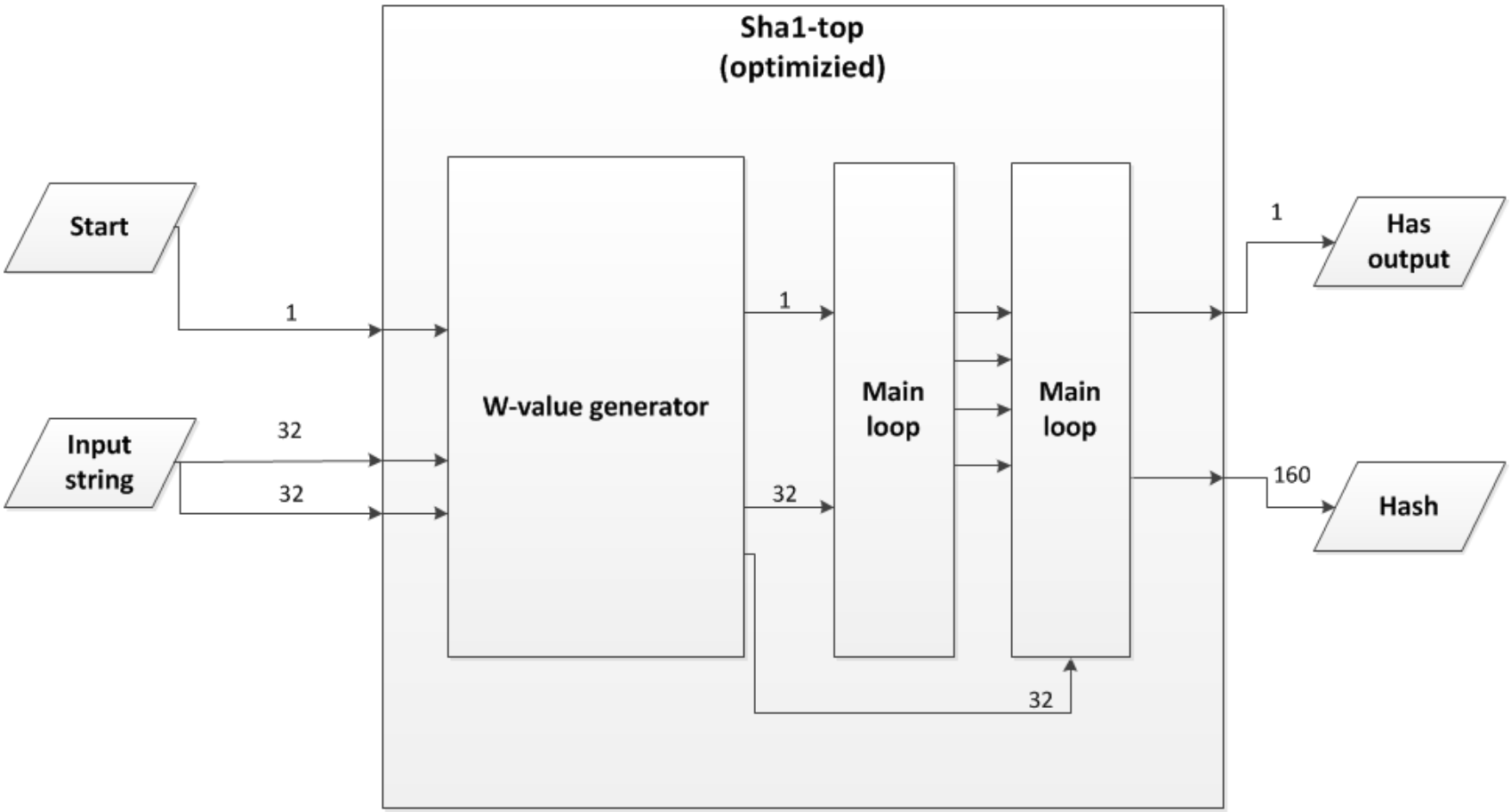

Figure: SHA1 hasher, second version

Bruteforcer

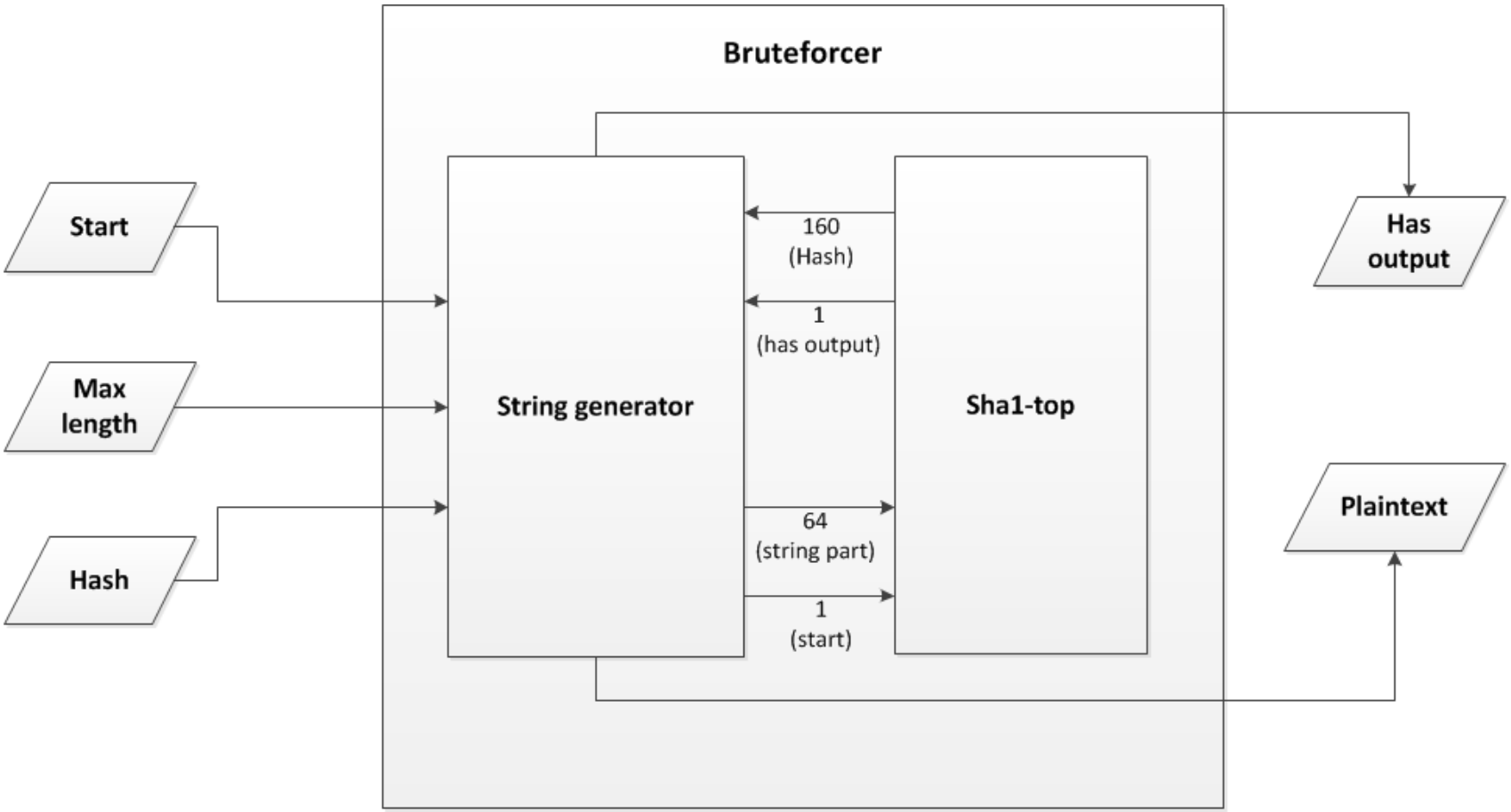

Figure: Complete bruteforcer, first version

Bruteforcer

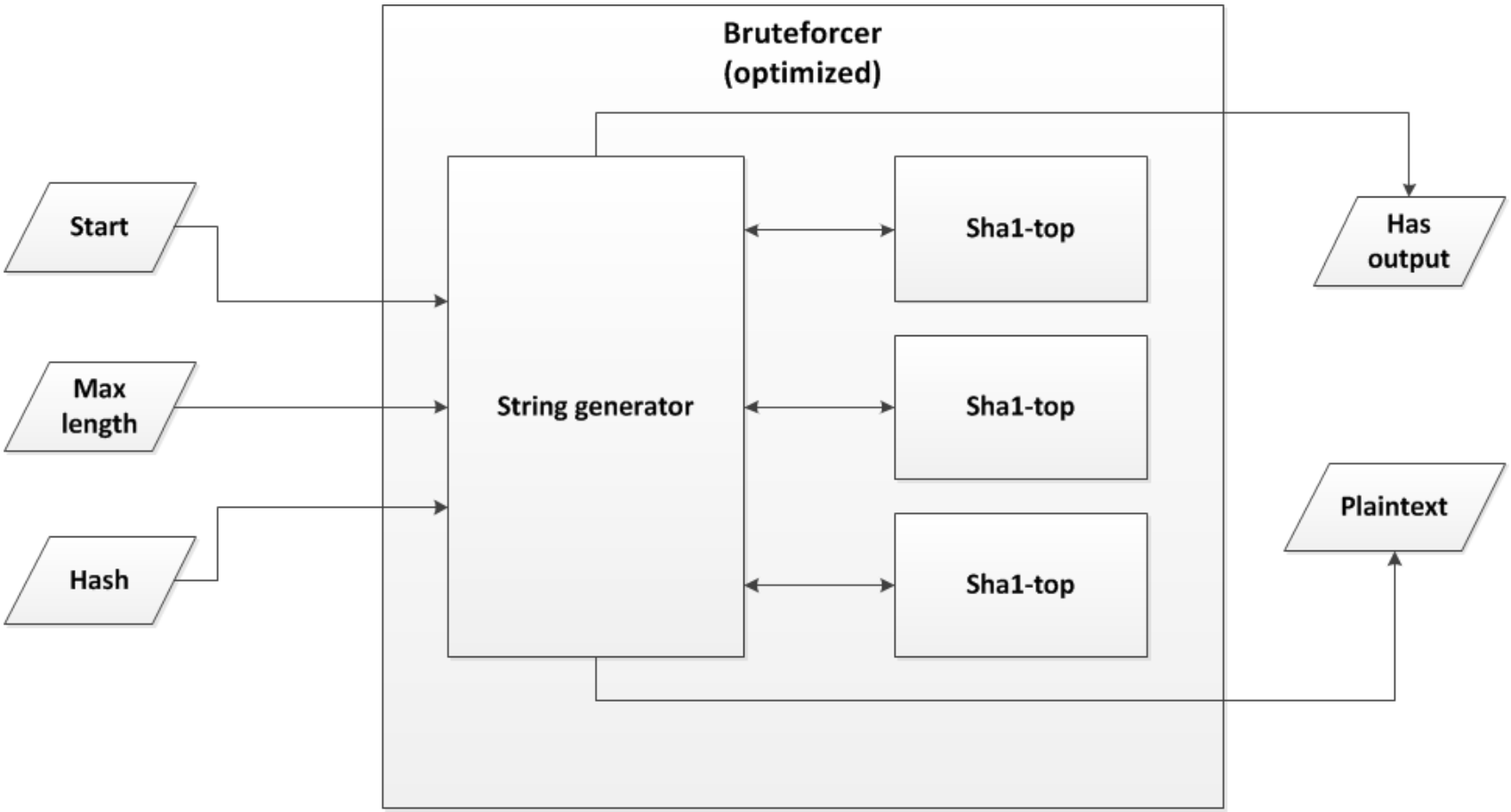

Figure: Complete bruteforcer, second version

Hasher stats

- 88 MHz frequency
- Each hasher approx $\frac{88 \cdot 1024 \cdot 1024}{40}$ hashes per second
- Three hashes in parallell equals $\frac{3 \cdot 88 \cdot 1024 \cdot 102}{40} = 6920602$ each second
- This equals 6920602 hashes per second
- (Real performance is close to this)

VGA

- 640x480 60 Hz
- Divided into 80x30 squares, 8x16 pixels large.
- Each square holds one character.
- Characters stored in 128x32 byte memory, one character = 1 byte.
- Microblaze writes to this memory to change screen content, VGA controller reads from this memory to display content.
- Monochrome display, only shows green.

VGA schematics

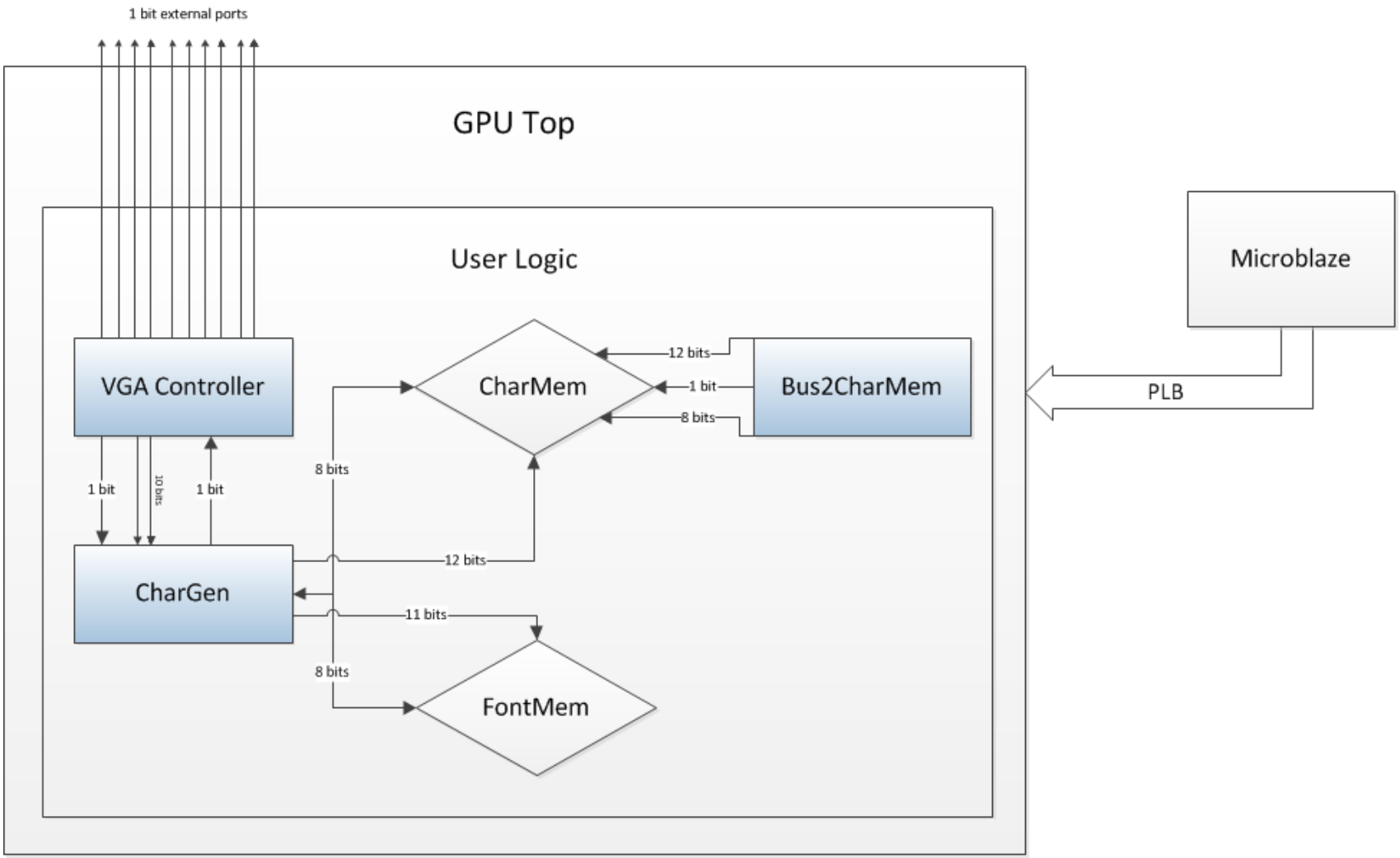

Figure: Overview of the VGA hardware

Dictionary attack

- Uses a prestored dictionary in non-volatile (pcm) memory.
- Includes common passwords and a normal English dictionary.
- Both the plaintext and the precomputed hash values are stored with a specified offset between them in memory.
- When a matching hash has been found, this offset is used to compute the address with the plaintext.
- Prestored hashes allow less hardware, which means the brute-force can be more optimized instead.

Software

- Implements the general communication between all the hardware components.
- Polls keys from the keyboard and process this input.
- Starts the brute force or dictionary attack depending on input.
- Updates the VGA character memory when new information should be displayed on the screen.

Problems

- Spent almost a week trying to make our custom keyboard controller communicate with the Microblaze, then we realized there was pre-constructed one available.
- Needed to include an if-statement in the VHDL code to fix a bug, this caused area overflow... (We were already at over 97% hardware)
- Obscure bug that required the user to press a key twice at some places for the input to be processed.
- These problems were solved with creative code in C.

Future improvements

- Optimize hardware usage further to make room for another hasher.
- Implement support for more hash functions (would require more hardware though).
- Implement a timer functionality for statistical purposes.
- Implement a larger and/or smarter dictionary attack.

The End