

# (Phonetic) text search using the FPGA technology

Ondřej Sýkora

June 27, 2009

# Introduction

## What it is all about

- ▶ What is an FPGA
  - ▶ Field-programmable Gate Array
  - ▶ Create circuits by programming in a higher-level language.

# Introduction

## What it is all about

- ▶ What is an FPGA
  - ▶ Field-programmable Gate Array
  - ▶ Create circuits by programming in a higher-level language.
- ▶ Why use an FPGA (people say that ...)

# Introduction

## What it is all about

- ▶ What is an FPGA
  - ▶ Field-programmable Gate Array
  - ▶ Create circuits by programming in a higher-level language.
- ▶ Why use an FPGA (people say that ...)
  - ▶ ... it's all parallel.

# Introduction

## What it is all about

- ▶ What is an FPGA
  - ▶ Field-programmable Gate Array
  - ▶ Create circuits by programming in a higher-level language.
- ▶ Why use an FPGA (people say that ...)
  - ▶ ... it's all parallel.
  - ▶ ... it's extremely fast.

# Introduction

## What it is all about

- ▶ What is an FPGA
  - ▶ Field-programmable Gate Array
  - ▶ Create circuits by programming in a higher-level language.
- ▶ Why use an FPGA (people say that ... )
  - ▶ ... it's all parallel.
  - ▶ ... it's extremely fast.
  - ▶ ... it's fun.
- ▶ Other technologies
  - ▶ ASIC's, CPU's, DSP's, Graphics hardware, ...

# Introduction

## What it is all about

- ▶ Phonetic search
  - ▶ Searching for words by their pronunciation, not using the exact spelling.

# Introduction

## What it is all about

- ▶ Phonetic search
  - ▶ Searching for words by their pronunciation, not using the exact spelling.
  - ▶ More (many) possible transcriptions for each word, not all of them grammatical.

# Introduction

## What it is all about

- ▶ Phonetic search
  - ▶ Searching for words by their pronunciation, not using the exact spelling.
  - ▶ More (many) possible transcriptions for each word, not all of them grammatical.
  - ▶ CZfind - phonetic search based on a transcription rules and a modified Aho-Corasick automaton.

# Introduction

## What it is all about

- ▶ Phonetic search
  - ▶ Searching for words by their pronunciation, not using the exact spelling.
  - ▶ More (many) possible transcriptions for each word, not all of them grammatical.
  - ▶ CZfind - phonetic search based on a transcription rules and a modified Aho-Corasick automaton.
  - ▶ Languages – Arabic and German

# Motivation

Why we used it

- ▶ Complexity of phonetic search. . .

# Motivation

Why we used it

- ▶ Complexity of phonetic search. . .

<b>Word</b>	<b>Transcriptions</b>
kála	33

# Motivation

Why we used it

- ▶ Complexity of phonetic search. . .

<b>Word</b>	<b>Transcriptions</b>
kála	33
bilmáji	944

# Motivation

Why we used it

- ▶ Complexity of phonetic search. . .

<b>Word</b>	<b>Transcriptions</b>
kála	33
bilmáji	944
kysatun	8547

# Motivation

Why we used it

- ▶ Complexity of phonetic search. . .

<b>Word</b>	<b>Transcriptions</b>
kála	33
bilmáji	944
kysatun	8547
kalratun	33973

# Motivation

Why we used it

- ▶ Complexity of phonetic search. . .

<b>Word</b>	<b>Transcriptions</b>
kála	33
bilmáji	944
kysatun	8547
kalratun	33973
machatatun	721709

# Motivation

Why we used it

- ▶ Complexity of phonetic search. . .

<b>Word</b>	<b>Transcriptions</b>
kála	33
bilmáji	944
kysatun	8547
kalratun	33973
machatatun	721709
tazkyratun	1082157

# Motivation

Why we used it

- ▶ Complexity of phonetic search. . .

<b>Word</b>	<b>Transcriptions</b>
kála	33
bilmáji	944
kysatun	8547
kalratun	33973
machatatun	721709
tazkyratun	1082157
alkuránulkarýmu	22154993

# Motivation

Why we used it

- ▶ Complexity of phonetic search. . .

<b>Word</b>	<b>Transcriptions</b>
kála	33
bilmáji	944
kysatun	8547
kalratun	33973
machatatun	721709
tazkyratun	1082157
alkuránulkarýmu	22154993
aliskandrarájatu	430638209

# Motivation

## Why we used it

- ▶ Complexity of phonetic search. . .
  - ▶ . . . exponential growth of the number of possible transcriptions.

# Motivation

## Why we used it

- ▶ Complexity of phonetic search. . .
  - ▶ . . . exponential growth of the number of possible transcriptions.
  - ▶ We get exponential growth of the number of states of the Aho-Corasick automata.

# Motivation

## Why we used it

- ▶ Complexity of phonetic search. . .
  - ▶ . . . exponential growth of the number of possible transcriptions.
  - ▶ We get exponential growth of the number of states of the Aho-Corasick automata.
- ▶ But consider non-deterministic automata

# Motivation

## Why we used it

- ▶ Complexity of phonetic search. . .
  - ▶ . . . exponential growth of the number of possible transcriptions.
  - ▶ We get exponential growth of the number of states of the Aho-Corasick automata.
- ▶ But consider non-deterministic automata
  - ▶ The transcription rules are strictly local (transcription of a single letter or a digraph)

# Motivation

## Why we used it

- ▶ Complexity of phonetic search. . .
  - ▶ . . . exponential growth of the number of possible transcriptions.
  - ▶ We get exponential growth of the number of states of the Aho-Corasick automata.
- ▶ But consider non-deterministic automata
  - ▶ The transcription rules are strictly local (transcription of a single letter or a digraph)
  - ▶ In non-determinism, we don't need backwards edges.
    - ▶ No need to handle failures, alternatives “live their own lives”

# Motivation

## Why we used it

- ▶ Complexity of phonetic search. . .
  - ▶ . . . exponential growth of the number of possible transcriptions.
  - ▶ We get exponential growth of the number of states of the Aho-Corasick automata.
- ▶ But consider non-deterministic automata
  - ▶ The transcription rules are strictly local (transcription of a single letter or a digraph)
  - ▶ In non-determinism, we don't need backwards edges.
    - ▶ No need to handle failures, alternatives “live their own lives”
  - ▶ As a result, the number of states of a NFA should be **linear to the length of the transcription**
    - ▶ . . . and the number of applicable rules.

# De-motivation

Exponential growth, but . . .

- ▶ Complexity of phonetic search for gramatic words

<b>Word</b>	<b>Transcriptions</b>
kála	33

# De-motivation

Exponential growth, but ...

- ▶ Complexity of phonetic search for gramatic words

<b>Word</b>	<b>Transcriptions</b>
kála	33
bilmáji	50

# De-motivation

Exponential growth, but ...

- ▶ Complexity of phonetic search for gramatic words

<b>Word</b>	<b>Transcriptions</b>
kála	33
bilmáji	50
kysatun	19

# De-motivation

Exponential growth, but ...

- ▶ Complexity of phonetic search for gramatic words

<b>Word</b>	<b>Transcriptions</b>
kála	33
bilmáji	50
kysatun	19
kalratun	13

# De-motivation

Exponential growth, but ...

- ▶ Complexity of phonetic search for gramatic words

<b>Word</b>	<b>Transcriptions</b>
kála	33
bilmáji	50
kysatun	19
kalratun	13
machatatun	17

# De-motivation

Exponential growth, but ...

- ▶ Complexity of phonetic search for gramatic words

<b>Word</b>	<b>Transcriptions</b>
kála	33
bilmáji	50
kysatun	19
kalratun	13
machatatun	17
tazkyratun	2

# De-motivation

Exponential growth, but ...

- ▶ Complexity of phonetic search for gramatic words

<b>Word</b>	<b>Transcriptions</b>
kála	33
bilmáji	50
kysatun	19
kalratun	13
machatatun	17
tazkyratun	2
alkuránulkarýmu	2

# De-motivation

Exponential growth, but ...

- ▶ Complexity of phonetic search for gramatic words

<b>Word</b>	<b>Transcriptions</b>
kála	33
bilmáji	50
kysatun	19
kalratun	13
machatatun	17
tazkyratun	2
alkuránulkarýmu	2
aliskandrarájatu	3

# De-motivation

Exponential growth, but ...

- ▶ Complexity of phonetic search for gramatic words

<b>Word</b>	<b>Transcriptions</b>
kála	33
bilmáji	50
kysatun	19
kalratun	13
machatatun	17
tazkyratun	2
alkuránulkarýmu	2
aliskandrarájatu	3

- ▶ But what about names of countries, persons and places? Not in a dictionary, but interesting to search for!

# Implementation

How it should be used

An expected workflow of the search automata on FPGA

# Implementation

How it should be used

An expected workflow of the search automata on FPGA

1. Get a phonetic transcription of the search phrase.

# Implementation

How it should be used

An expected workflow of the search automata on FPGA

1. Get a phonetic transcription of the search phrase.
2. Create a search automaton based on the transcription.

# Implementation

How it should be used

An expected workflow of the search automata on FPGA

1. Get a phonetic transcription of the search phrase.
2. Create a search automaton based on the transcription.
3. Generate and compile a program with the automaton.

# Implementation

## How it should be used

An expected workflow of the search automata on FPGA

1. Get a phonetic transcription of the search phrase.
2. Create a search automaton based on the transcription.
3. Generate and compile a program with the automaton.
4. Upload the program to the FPGA.

# Implementation

## How it should be used

An expected workflow of the search automata on FPGA

1. Get a phonetic transcription of the search phrase.
2. Create a search automaton based on the transcription.
3. Generate and compile a program with the automaton.
4. Upload the program to the FPGA.
5. Perform the search.

# Implementation

## How it should be used

An expected workflow of the search automata on FPGA

1. Get a phonetic transcription of the search phrase.
2. Create a search automaton based on the transcription.
3. Generate and compile a program with the automaton.
4. Upload the program to the FPGA.
5. Perform the search.
  - ▶ Important questions:
    - ▶ How to get data to the FPGA?
    - ▶ How to get search results from the FPGA?

# Implementation

## How it should be used

An expected workflow of the search automata on FPGA

1. Get a phonetic transcription of the search phrase.
2. Create a search automaton based on the transcription.
3. Generate and compile a program with the automaton.
4. Upload the program to the FPGA.
5. Perform the search.
  - ▶ Important questions:
    - ▶ How to get data to the FPGA?
    - ▶ How to get search results from the FPGA?
  - ▶ Partial answers:
    - ▶ Network interfaces (ethernet)
    - ▶ PCI (Express)

# Implementation

How did we do it – programming

- ▶ On the FPGA

# Implementation

How did we do it – programming

- ▶ On the FPGA
  - ▶ DFA - a single register for storing the current state.
  - ▶ NFA - a binary register for the activity of each of the states.
  - ▶ Transitions encoded in the circuit.

# Implementation

How did we do it – programming

- ▶ On the FPGA
  - ▶ DFA - a single register for storing the current state.
  - ▶ NFA - a binary register for the activity of each of the states.
  - ▶ Transitions encoded in the circuit.
- ▶ On the PC

# Implementation

How did we do it – programming

- ▶ On the FPGA
  - ▶ DFA - a single register for storing the current state.
  - ▶ NFA - a binary register for the activity of each of the states.
  - ▶ Transitions encoded in the circuit.
- ▶ On the PC
  - ▶ DFA - a single variable for storing the current state.
  - ▶ NFA - a boolean variable for the activity of each of the states.
  - ▶ Transitions provided as parameters of the algorithm.

# Implementation

How did we do it – programming

- ▶ On the FPGA
  - ▶ DFA - a single register for storing the current state.
  - ▶ NFA - a binary register for the activity of each of the states.
  - ▶ Transitions encoded in the circuit.
- ▶ On the PC
  - ▶ DFA - a single variable for storing the current state.
  - ▶ NFA - a boolean variable for the activity of each of the states.
  - ▶ Transitions provided as parameters of the algorithm.
- ▶ Where is the difference?

# Implementation

How did we do it – programming

- ▶ On the FPGA
  - ▶ DFA - a single register for storing the current state.
  - ▶ NFA - a binary register for the activity of each of the states.
  - ▶ Transitions encoded in the circuit.
- ▶ On the PC
  - ▶ DFA - a single variable for storing the current state.
  - ▶ NFA - a boolean variable for the activity of each of the states.
  - ▶ Transitions provided as parameters of the algorithm.
- ▶ Where is the difference?
  - ▶ Parallel updates of the activity of the states on FPGA.

# Implementation

How did we do it – the algorithms

# Implementation

How did we do it – the algorithms

- ▶ Aho-Corasick automata
  - ▶ Well-known algorithm, without modifications. The backwards function  $f$  is encoded into the transitions.

# Implementation

How did we do it – the algorithms

- ▶ Aho-Corasick automata
  - ▶ Well-known algorithm, without modifications. The backwards function  $f$  is encoded into the transitions.
- ▶ Non-deterministic automata
  - ▶ Created from regular expressions (which we get from CZfind).

# Implementation

How did we do it – the algorithms

- ▶ Aho-Corasick automata
  - ▶ Well-known algorithm, without modifications. The backwards function  $f$  is encoded into the transitions.
- ▶ Non-deterministic automata
  - ▶ Created from regular expressions (which we get from CZfind).
  - ▶ The regular expressions from CZfind lead to automata far from the minimal size

# Implementation

How did we do it – the algorithms

- ▶ Aho-Corasick automata
  - ▶ Well-known algorithm, without modifications. The backwards function  $f$  is encoded into the transitions.
- ▶ Non-deterministic automata
  - ▶ Created from regular expressions (which we get from CZfind).
  - ▶ The regular expressions from CZfind lead to automata far from the minimal size
    - ▶ Many branches, almost no merging.

# Implementation

How did we do it – the algorithms

- ▶ Aho-Corasick automata
  - ▶ Well-known algorithm, without modifications. The backwards function  $f$  is encoded into the transitions.
- ▶ Non-deterministic automata
  - ▶ Created from regular expressions (which we get from CZfind).
  - ▶ The regular expressions from CZfind lead to automata far from the minimal size
    - ▶ Many branches, almost no merging.
    - ▶ Addressed by a factorization algorithm – does not produce minimal automata, but we get small-enough ones.

# Implementation

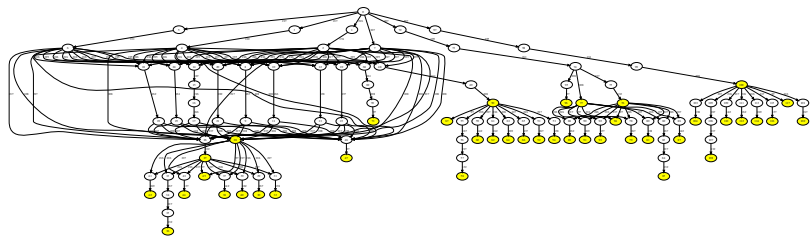
## How did we do it – the algorithms

- ▶ Aho-Corasick automata
  - ▶ Well-known algorithm, without modifications. The backwards function  $f$  is encoded into the transitions.
- ▶ Non-deterministic automata
  - ▶ Created from regular expressions (which we get from CZfind).
  - ▶ The regular expressions from CZfind lead to automata far from the minimal size
    - ▶ Many branches, almost no merging.
    - ▶ Addressed by a factorization algorithm – does not produce minimal automata, but we get small-enough ones.
    - ▶ A possible better solution – create the NFA directly from the phonetic transcription using the rules.

# Implementation

How did we do it – the algorithms

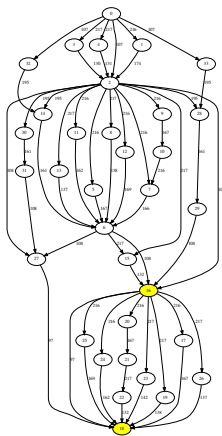
Before factorization...



# Implementation

How did we do it – the algorithms

... after factorization



# FPGA implementations

How did it work

# FPGA implementations

How did it work

- ▶ FPGA implementation of the automata

# FPGA implementations

How did it work

- ▶ FPGA implementation of the automata
  - ▶ (+) ... is possible.

# FPGA implementations

## How did it work

- ▶ FPGA implementation of the automata
  - ▶ (+) ... is possible.
  - ▶ (+) “Constant” time for processing a single character for both NFA's and DFA's.
    - ▶ The time is about 10 ns per character.
    - ▶ Similar for DFA's and NFA's (parallelism!).

# FPGA implementations

## How did it work

- ▶ FPGA implementation of the automata
  - ▶ (+) ... is possible.
  - ▶ (+) “Constant” time for processing a single character for both NFA's and DFA's.
    - ▶ The time is about 10 ns per character.
    - ▶ Similar for DFA's and NFA's (parallelism!).
  - ▶ (–) Compilation may take hours for larger automata.

# FPGA implementations

## How did it work

- ▶ FPGA implementation of the automata
  - ▶ (+) ... is possible.
  - ▶ (+) “Constant” time for processing a single character for both NFA's and DFA's.
    - ▶ The time is about 10 ns per character.
    - ▶ Similar for DFA's and NFA's (parallelism!).
  - ▶ (–) Compilation may take hours for larger automata.
  - ▶ (–) Significant amount of programming to get the data transfer working.

# FPGA implementations

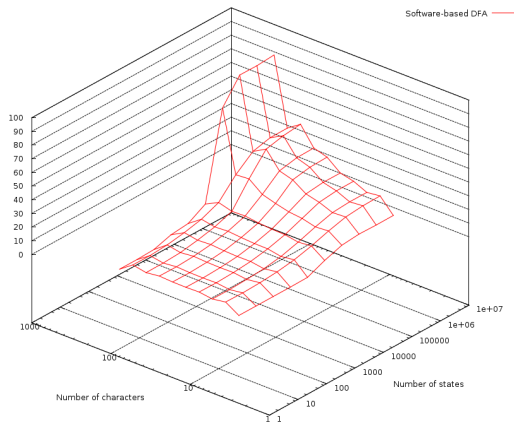
## How did it work

- ▶ FPGA implementation of the automata
  - ▶ (+) ... is possible.
  - ▶ (+) “Constant” time for processing a single character for both NFA's and DFA's.
    - ▶ The time is about 10 ns per character.
    - ▶ Similar for DFA's and NFA's (parallelism!).
  - ▶ (–) Compilation may take hours for larger automata.
  - ▶ (–) Significant amount of programming to get the data transfer working.
  - ▶ (–) Uncertain behaviour for very large words.

# CPU implementations

Can we do it better?

- ▶ What are running times for DFA on a CPU?



# CPU implementations

Can we do it better?

- ▶ And NFA's?

# CPU implementations

Can we do it better?

- ▶ And NFA's?

<b>Word</b>	<b>NFA step time (ns)</b>
rýsá	90.9

# CPU implementations

Can we do it better?

- ▶ And NFA's?

<b>Word</b>	<b>NFA step time (ns)</b>
rýsá	90.9
vadrún	120.22

# CPU implementations

Can we do it better?

- ▶ And NFA's?

<b>Word</b>	<b>NFA step time (ns)</b>
rýsá	90.9
vadrún	120.22
mustafá	150.91

# CPU implementations

Can we do it better?

- ▶ And NFA's?

<b>Word</b>	<b>NFA step time (ns)</b>
rýsá	90.9
vadrún	120.22
mustafá	150.91
sáratun	167.59

# CPU implementations

Can we do it better?

► And NFA's?

<b>Word</b>	<b>NFA step time (ns)</b>
rýsá	90.9
vadrún	120.22
mustafá	150.91
sáratun	167.59
safínatun	205.58

# CPU implementations

Can we do it better?

► And NFA's?

<b>Word</b>	<b>NFA step time (ns)</b>
rýsá	90.9
vadrún	120.22
mustafá	150.91
sáratun	167.59
safínatun	205.58
machatatun	240.27

# CPU implementations

Can we do it better?

► And NFA's?

<b>Word</b>	<b>NFA step time (ns)</b>
rýsá	90.9
vadrún	120.22
mustafá	150.91
sáratun	167.59
safínatun	205.58
machatatun	240.27
aliskandrarájatu	360.02

# CPU implementations

DFA's revisited

# CPU implementations

## DFA's revisited

- ▶ How realistic are random DFA's?

# CPU implementations

## DFA's revisited

- ▶ How realistic are random DFA's?
  - ▶ Not at all...

# CPU implementations

## DFA's revisited

- ▶ How realistic are random DFA's?
  - ▶ Not at all...
- ▶ But then, how realistic are the running times?

# CPU implementations

## DFA's revisited

- ▶ How realistic are random DFA's?
  - ▶ Not at all...
- ▶ But then, how realistic are the running times?
- ▶ The problem with DFA's on a CPU is (probably) the CPU cache. How does it affect the Aho-Corasick automata?

# CPU implementations

## DFA's revisited

- ▶ How realistic are random DFA's?
  - ▶ Not at all...
- ▶ But then, how realistic are the running times?
- ▶ The problem with DFA's on a CPU is (probably) the CPU cache. How does it affect the Aho-Corasick automata?
  - ▶ Based on a trie  $\Rightarrow$  rooted graph, levels

# CPU implementations

## DFA's revisited

- ▶ How realistic are random DFA's?
  - ▶ Not at all...
- ▶ But then, how realistic are the running times?
- ▶ The problem with DFA's on a CPU is (probably) the CPU cache. How does it affect the Aho-Corasick automata?
  - ▶ Based on a trie  $\Rightarrow$  rooted graph, levels
  - ▶ The deeper the level, the longer the match

# CPU implementations

## DFA's revisited

- ▶ How realistic are random DFA's?
  - ▶ Not at all. . .
- ▶ But then, how realistic are the running times?
- ▶ The problem with DFA's on a CPU is (probably) the CPU cache. How does it affect the Aho-Corasick automata?
  - ▶ Based on a trie  $\Rightarrow$  rooted graph, levels
  - ▶ The deeper the level, the longer the match
    - ▶ If matches are rare, then the automaton remains around the first few levels!

# CPU implementations

## DFA's revisited

- ▶ How many times the states are visited (when processing a real 250MB text sample)?

# CPU implementations

## DFA's revisited

- ▶ How many times the states are visited (when processing a real 250MB text sample)?
  - ▶ More than  $200 * 10^6$  – single state (initial state)

# CPU implementations

## DFA's revisited

- ▶ How many times the states are visited (when processing a real 250MB text sample)?
  - ▶ More than  $200 * 10^6$  – single state (initial state)
  - ▶ More than  $10 * 10^6$  – two states

# CPU implementations

## DFA's revisited

- ▶ How many times the states are visited (when processing a real 250MB text sample)?
  - ▶ More than  $200 * 10^6$  – single state (initial state)
  - ▶ More than  $10 * 10^6$  – two states
  - ▶ More than  $5 * 10^6$  – three states

# CPU implementations

## DFA's revisited

- ▶ How many times the states are visited (when processing a real 250MB text sample)?
  - ▶ More than  $200 * 10^6$  – single state (initial state)
  - ▶ More than  $10 * 10^6$  – two states
  - ▶ More than  $5 * 10^6$  – three states
  - ▶ More than  $10^6$  – ten states

# CPU implementations

## DFA's revisited

- ▶ How many times the states are visited (when processing a real 250MB text sample)?
  - ▶ More than  $200 * 10^6$  – single state (initial state)
  - ▶ More than  $10 * 10^6$  – two states
  - ▶ More than  $5 * 10^6$  – three states
  - ▶ More than  $10^6$  – ten states
  - ▶ More than 1000 – 61 states

# CPU implementations

## DFA's revisited

- ▶ How many times the states are visited (when processing a real 250MB text sample)?
  - ▶ More than  $200 * 10^6$  – single state (initial state)
  - ▶ More than  $10 * 10^6$  – two states
  - ▶ More than  $5 * 10^6$  – three states
  - ▶ More than  $10^6$  – ten states
  - ▶ More than 1000 – 61 states
- ▶ 90 % of time spent in three states.
- ▶ 98 % of time spent in up to ten states.

# CPU implementations

## DFA's revisited

- ▶ How many times the states are visited (when processing a real 250MB text sample)?
  - ▶ More than  $200 * 10^6$  – single state (initial state)
  - ▶ More than  $10 * 10^6$  – two states
  - ▶ More than  $5 * 10^6$  – three states
  - ▶ More than  $10^6$  – ten states
  - ▶ More than 1000 – 61 states
- ▶ 90 % of time spent in three states.
- ▶ 98 % of time spent in up to ten states.
  
- ▶ In fact, the Aho-Corasick automata of any size processed a single character in cca 2.45 ns.

# CPU implementations

Are there any limitations?

- ▶ The applicability of the DFA's on a CPU

# CPU implementations

Are there any limitations?

- ▶ The applicability of the DFA's on a CPU
  - ▶ The exponential growth of the number of possible transcriptions can bring severe limitations

# CPU implementations

Are there any limitations?

- ▶ The applicability of the DFA's on a CPU
  - ▶ The exponential growth of the number of possible transcriptions can bring severe limitations

<b>Word</b>	Transcriptions	States
machatatun	721709	1468758

# CPU implementations

Are there any limitations?

- ▶ The applicability of the DFA's on a CPU
  - ▶ The exponential growth of the number of possible transcriptions can bring severe limitations

<b>Word</b>	Transcriptions	States
machatatun	721709	1468758
tazkyratun	1082157	2214458

# CPU implementations

Are there any limitations?

- ▶ The applicability of the DFA's on a CPU
  - ▶ The exponential growth of the number of possible transcriptions can bring severe limitations

<b>Word</b>	Transcriptions	States
machatatun	721709	1468758
tazkyratun	1082157	2214458
sajjáratun	1690413	3390610

# CPU implementations

Are there any limitations?

- ▶ The applicability of the DFA's on a CPU
  - ▶ The exponential growth of the number of possible transcriptions can bring severe limitations

<b>Word</b>	Transcriptions	States
machatatun	721709	1468758
tazkyratun	1082157	2214458
sajjáratun	1690413	3390610
muntasaulajli	1415197	5125516

# CPU implementations

Are there any limitations?

- ▶ The applicability of the DFA's on a CPU
  - ▶ The exponential growth of the number of possible transcriptions can bring severe limitations

<b>Word</b>	Transcriptions	States
machatatun	721709	1468758
tazkyratun	1082157	2214458
sajjáratun	1690413	3390610
muntasaulajli	1415197	5125516
fûlun súdánijun	2787840	5747819

# CPU implementations

Are there any limitations?

- ▶ The applicability of the DFA's on a CPU
  - ▶ The exponential growth of the number of possible transcriptions can bring severe limitations

<b>Word</b>	Transcriptions	States
machatatun	721709	1468758
tazkyratun	1082157	2214458
sajjáratun	1690413	3390610
muntasaulajli	1415197	5125516
fûlun súdánijun	2787840	5747819
aliskandrarájatu	430638209	?

# CPU implementations

Are there any limitations?

- ▶ The applicability of the DFA's on a CPU
  - ▶ The exponential growth of the number of possible transcriptions can bring severe limitations

<b>Word</b>	Transcriptions	States
machatatun	721709	1468758
tazkyratun	1082157	2214458
sajjáratun	1690413	3390610
muntasaulajli	1415197	5125516
fûlun súdánijun	2787840	5747819
aliskandrarájatu	430638209	?
alžámirulazharu	32804738	?

# CPU implementations

Are there any limitations?

- ▶ The applicability of the DFA's on a CPU
  - ▶ The exponential growth of the number of possible transcriptions can bring severe limitations

<b>Word</b>	Transcriptions	States
machatatun	721709	1468758
tazkyratun	1082157	2214458
sajjáratun	1690413	3390610
muntasaulajli	1415197	5125516
fūlun sūdánijun	2787840	5747819
aliskandrarájatu	430638209	?
alžámirulazharu	32804738	?
sálatulistykbáli	19739628	?

# CPU implementations

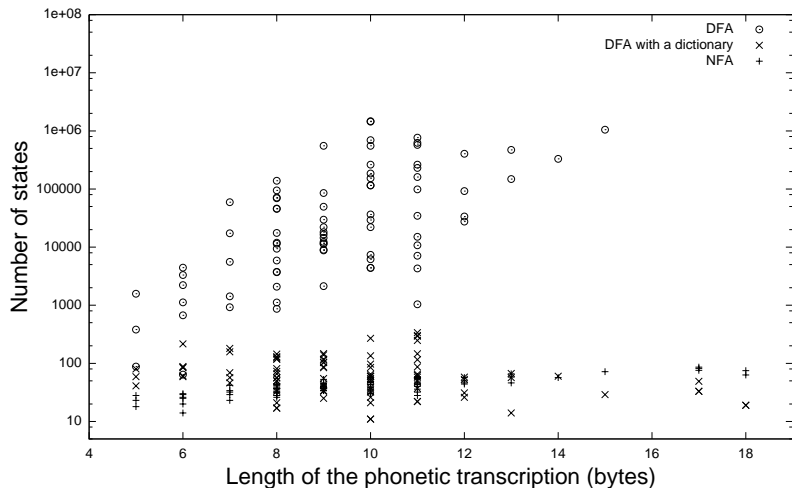
Are there any limitations?

- ▶ The applicability of the DFA's on a CPU
  - ▶ The exponential growth of the number of possible transcriptions can bring severe limitations

<b>Word</b>	Transcriptions	States
machatatun	721709	1468758
tazkyratun	1082157	2214458
sajjáratun	1690413	3390610
muntasaulajli	1415197	5125516
fûlun súdánijun	2787840	5747819
aliskandrarájatu	430638209	?
alžámirulazharu	32804738	?
sálatulistykbáli	19739628	?
alkuránulkarýmu	22154993	?

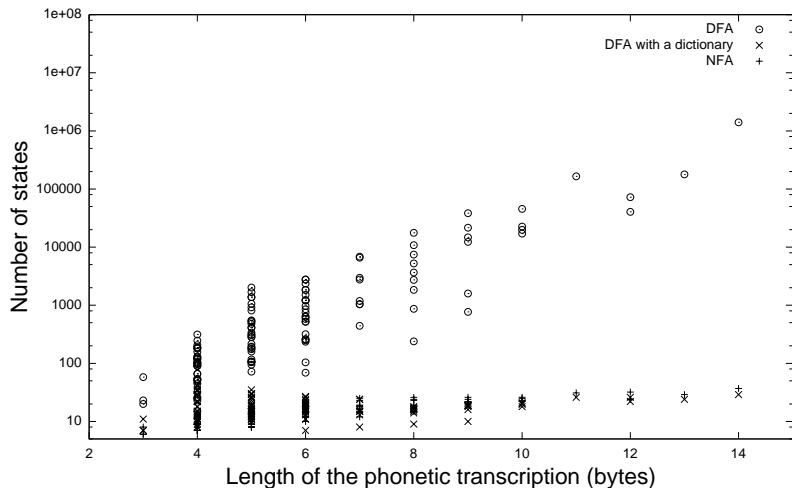
# Limitations and applicability

How do the DFA's and NFA's compare?



# Limitations and applicability

And what about other languages?



# Limitations and applicability

Are there any limitations of the NFA's?

# Limitations and applicability

Are there any limitations of the NFA's?

- ▶ NFA's are significantly smaller than the corresponding DFA's

# Limitations and applicability

Are there any limitations of the NFA's?

- ▶ NFA's are significantly smaller than the corresponding DFA's
  - ▶ Allows us to process even very long search phrases (consisting of several words).

# Limitations and applicability

Are there any limitations of the NFA's?

- ▶ NFA's are significantly smaller than the corresponding DFA's
  - ▶ Allows us to process even very long search phrases (consisting of several words).
- ▶ The NFA's can be limited by the capacity of the FPGA's (logic blocks used by the automata).

# Limitations and applicability

Are there any limitations of the NFA's?

- ▶ NFA's are significantly smaller than the corresponding DFA's
  - ▶ Allows us to process even very long search phrases (consisting of several words).
- ▶ The NFA's can be limited by the capacity of the FPGA's (logic blocks used by the automata).
  - ▶ The FPGA board Altera DE2 has about 33000 logic blocks.

# Limitations and applicability

Are there any limitations of the NFA's?

- ▶ NFA's are significantly smaller than the corresponding DFA's
  - ▶ Allows us to process even very long search phrases (consisting of several words).
- ▶ The NFA's can be limited by the capacity of the FPGA's (logic blocks used by the automata).
  - ▶ The FPGA board Altera DE2 has about 33000 logic blocks.
  - ▶ More problems – a very limited number of input/output pins (cca 500).

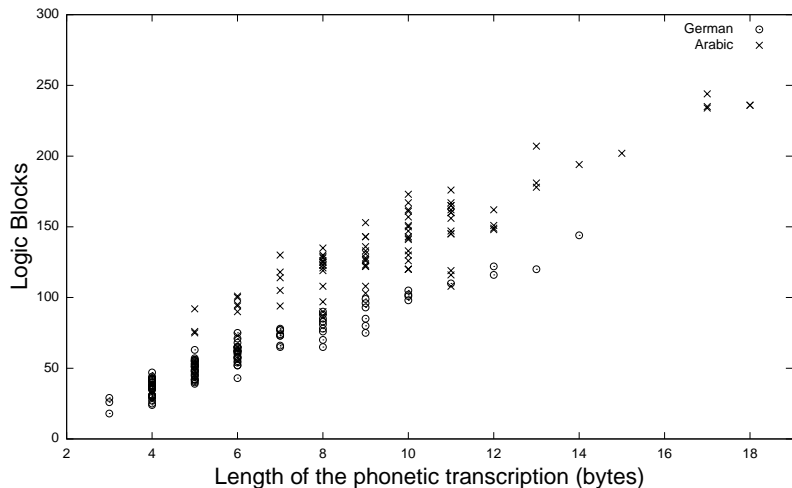
# Limitations and applicability

Are there any limitations of the NFA's?

- ▶ NFA's are significantly smaller than the corresponding DFA's
  - ▶ Allows us to process even very long search phrases (consisting of several words).
- ▶ The NFA's can be limited by the capacity of the FPGA's (logic blocks used by the automata).
  - ▶ The FPGA board Altera DE2 has about 33000 logic blocks.
  - ▶ More problems – a very limited number of input/output pins (cca 500).
    - ▶ Can be solved by using a single output pin (which is also an implication of the “minimization” of the automata).

# Limitations and applicability

## Limitations of the NFA's



# Conclusions & Further research

What did we learn?

- ▶ Unexpected performance of the Aho-Corasick automata.

# Conclusions & Further research

What did we learn?

- ▶ Unexpected performance of the Aho-Corasick automata.
  - ▶ If it is possible to build the Aho-Corasick automaton and run it on a CPU, it does not pay to use an FPGA.

# Conclusions & Further research

What did we learn?

- ▶ Unexpected performance of the Aho-Corasick automata.
  - ▶ If it is possible to build the Aho-Corasick automaton and run it on a CPU, it does not pay to use an FPGA.
  - ▶ Applicable for transcriptions filtered by a dictionary.

# Conclusions & Further research

What did we learn?

- ▶ Unexpected performance of the Aho-Corasick automata.
  - ▶ If it is possible to build the Aho-Corasick automaton and run it on a CPU, it does not pay to use an FPGA.
  - ▶ Applicable for transcriptions filtered by a dictionary.
  - ▶ Applicable for shorter words without a dictionary (up to cca 10-14 characters).

# Conclusions & Further research

What did we learn?

- ▶ Unexpected performance of the Aho-Corasick automata.
  - ▶ If it is possible to build the Aho-Corasick automaton and run it on a CPU, it does not pay to use an FPGA.
  - ▶ Applicable for transcriptions filtered by a dictionary.
  - ▶ Applicable for shorter words without a dictionary (up to cca 10-14 characters).
- ▶ Good performance of NFA's on the FPGA's

# Conclusions & Further research

What did we learn?

- ▶ Unexpected performance of the Aho-Corasick automata.
  - ▶ If it is possible to build the Aho-Corasick automaton and run it on a CPU, it does not pay to use an FPGA.
  - ▶ Applicable for transcriptions filtered by a dictionary.
  - ▶ Applicable for shorter words without a dictionary (up to cca 10-14 characters).
- ▶ Good performance of NFA's on the FPGA's
  - ▶ (almost) a constant time for processing a single character

# Conclusions & Further research

What did we learn?

- ▶ Unexpected performance of the Aho-Corasick automata.
  - ▶ If it is possible to build the Aho-Corasick automaton and run it on a CPU, it does not pay to use an FPGA.
  - ▶ Applicable for transcriptions filtered by a dictionary.
  - ▶ Applicable for shorter words without a dictionary (up to cca 10-14 characters).
- ▶ Good performance of NFA's on the FPGA's
  - ▶ (almost) a constant time for processing a single character
- ▶ Very small NFA's even for large words

# Conclusions & Further research

What did we learn?

- ▶ Unexpected performance of the Aho-Corasick automata.
  - ▶ If it is possible to build the Aho-Corasick automaton and run it on a CPU, it does not pay to use an FPGA.
  - ▶ Applicable for transcriptions filtered by a dictionary.
  - ▶ Applicable for shorter words without a dictionary (up to cca 10-14 characters).
- ▶ Good performance of NFA's on the FPGA's
  - ▶ (almost) a constant time for processing a single character
- ▶ Very small NFA's even for large words
  - ▶ Parallel searches on the FPGA

# Conclusions & Further research

What did we learn?

- ▶ Unexpected performance of the Aho-Corasick automata.
  - ▶ If it is possible to build the Aho-Corasick automaton and run it on a CPU, it does not pay to use an FPGA.
  - ▶ Applicable for transcriptions filtered by a dictionary.
  - ▶ Applicable for shorter words without a dictionary (up to cca 10-14 characters).
- ▶ Good performance of NFA's on the FPGA's
  - ▶ (almost) a constant time for processing a single character
- ▶ Very small NFA's even for large words
  - ▶ Parallel searches on the FPGA
  - ▶ A memory-effective representation, even when not using an FPGA.

# Conclusions & Further research

What did we learn?

- ▶ Unexpected performance of the Aho-Corasick automata.
  - ▶ If it is possible to build the Aho-Corasick automaton and run it on a CPU, it does not pay to use an FPGA.
  - ▶ Applicable for transcriptions filtered by a dictionary.
  - ▶ Applicable for shorter words without a dictionary (up to cca 10-14 characters).
- ▶ Good performance of NFA's on the FPGA's
  - ▶ (almost) a constant time for processing a single character
- ▶ Very small NFA's even for large words
  - ▶ Parallel searches on the FPGA
  - ▶ A memory-effective representation, even when not using an FPGA.
- ▶ Relatively low performance of the FPGA (“not a consumer hardware”).

# Conclusions & Further research

What can we do next?

# Conclusions & Further research

What can we do next?

- ▶ A better construction of the NFA's (without using the regular expressions)

# Conclusions & Further research

What can we do next?

- ▶ A better construction of the NFA's (without using the regular expressions)
- ▶ Multi-layer NFA's to speed up processing (avoid registers).

# Conclusions & Further research

What can we do next?

- ▶ A better construction of the NFA's (without using the regular expressions)
- ▶ Multi-layer NFA's to speed up processing (avoid registers).
- ▶ Approximate phonetic string matching.

# Conclusions & Further research

What can we do next?

- ▶ A better construction of the NFA's (without using the regular expressions)
- ▶ Multi-layer NFA's to speed up processing (avoid registers).
- ▶ Approximate phonetic string matching.
- ▶ Change the hardware platform for NFA's.

# Conclusions & Further research

What can we do next?

- ▶ A better construction of the NFA's (without using the regular expressions)
- ▶ Multi-layer NFA's to speed up processing (avoid registers).
- ▶ Approximate phonetic string matching.
- ▶ Change the hardware platform for NFA's.
- ▶ On-demand DFA loading.

# Conclusions

This is the last one

- ▶ Thank you for your attention.
- ▶ Questions, Comments?