

# Compiler Support for Hardware Accelerators

Jackson Woodruff

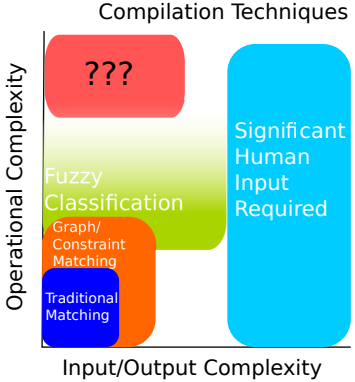
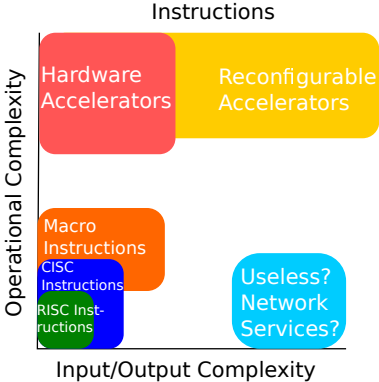
University of Edinburgh

9 September, 2021

# Big-Step Hardware Acceleration

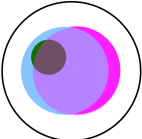
- ▶ Big-Step Accelerators do lots of things in one step:
  - ▶ FFT
  - ▶ CRC32
- ▶ The more an accelerator does, the more overhead can be amortized

# Existing Compiler Technology



# Matching Isn't Enough

## Accelerator is useful



Near or exactly shared behaviour

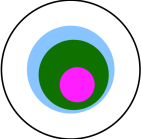


User code supports all values and is subset

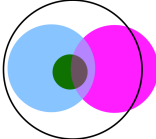


User code supports some values and is subset

## Accelerator may be useful Condition: Size of overlap region



Accelerator is a subset



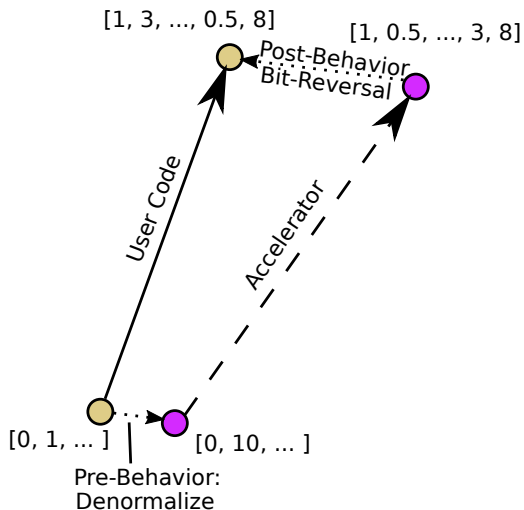
Partial overlap

## Accelerator not useful

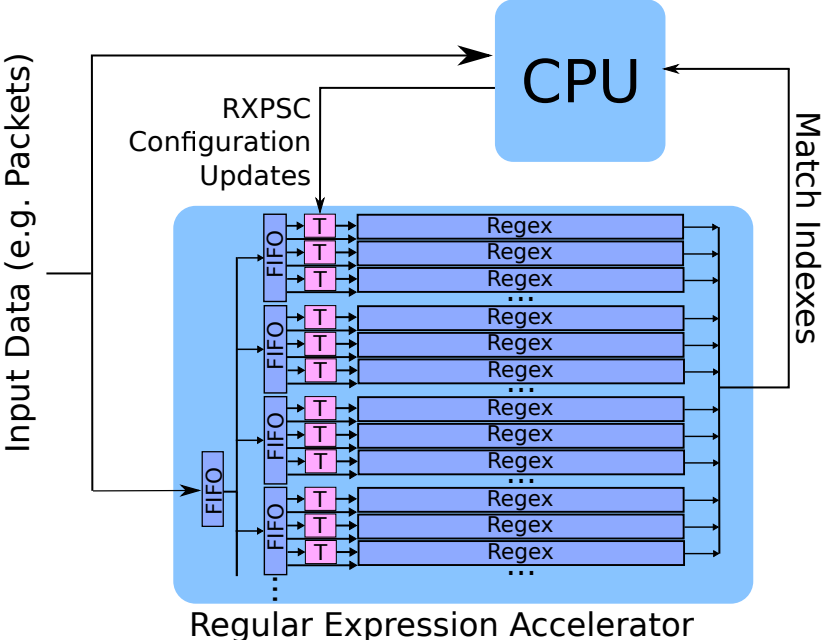


- Set of valid inputs to accelerator
- Set of valid inputs to user code
- Set of possible inputs to user code
- Set of used inputs to user code

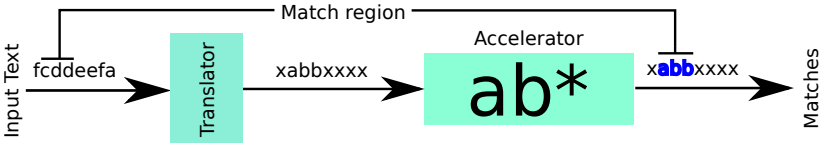
# Supporting Hardware Accelerators



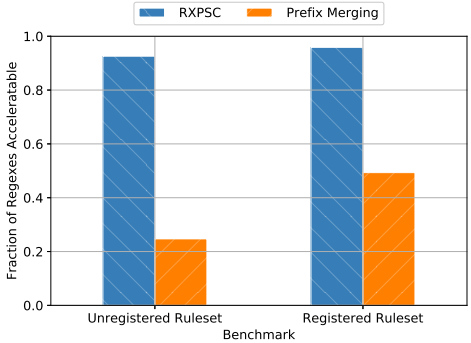
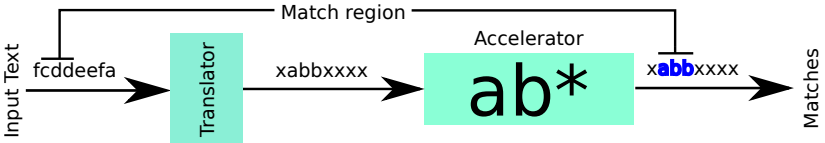
# Regular Expression Acceleration



# Regular Expression Acceleration

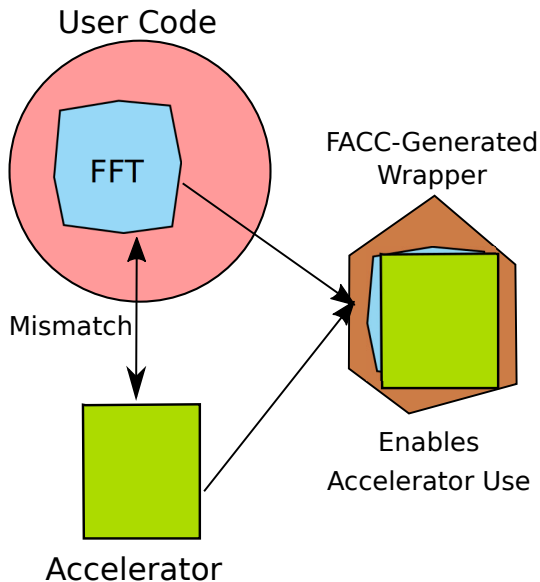


# Regular Expression Acceleration

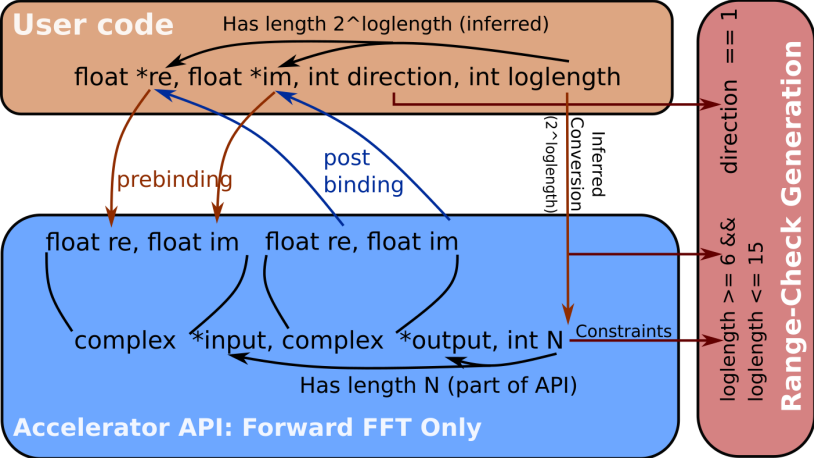




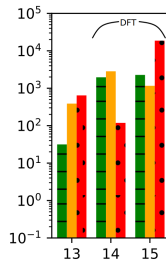
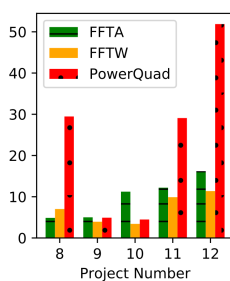
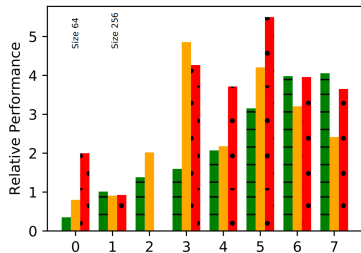
## Fourier Transform Acceleration



# Fourier Transform Acceleration: Method



# Fourier Transform Acceleration: Results



# Conclusion

- ▶ Hardware accelerators lack:
  - ▶ Portability
  - ▶ Ease-of-use
  - ▶ Integration into software-based development pipelines
- ▶ Solutions to close the gap between software programming and hardware accelerators required