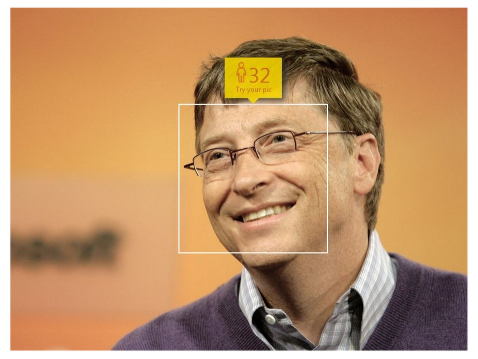
A Case Study of Machine Learning Hardware: Real-Time Source Separation using Markov Random Fields via Sampling-based Inference

Glenn G. Ko and Rob A. Rutenbar

University of Illinois at Urbana-Champaign

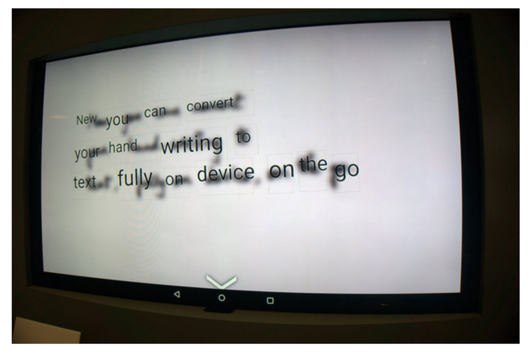
Motivation

Why Machine Learning in Hardware?



Enterprise App: facial recognition

Apps

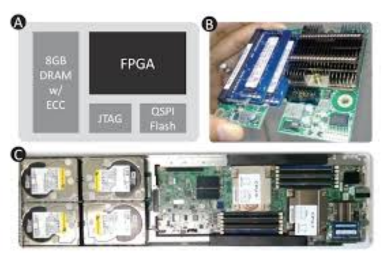


Mobile App: Handwriting Recognition

Big, complex, slow. Slow to run, slower (days/weeks) to train.

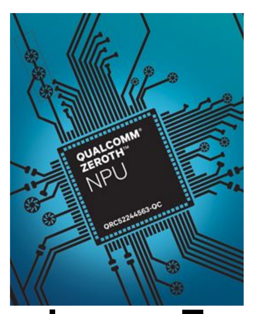
Problems

Challenges involve running in real-time and low power



Microsoft Catapult

Solutions

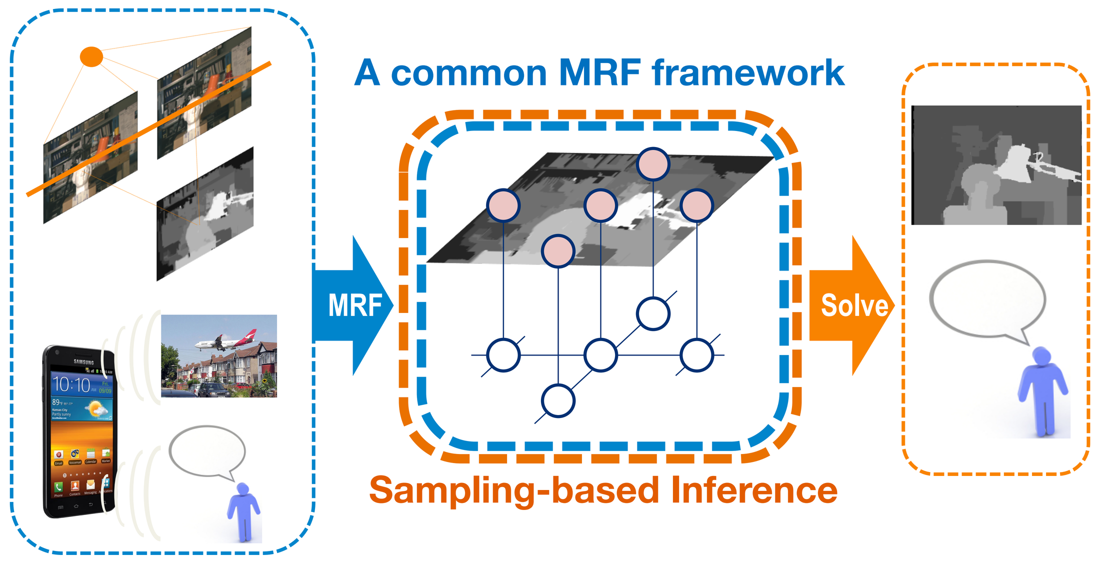


Qualcomm Zeroth

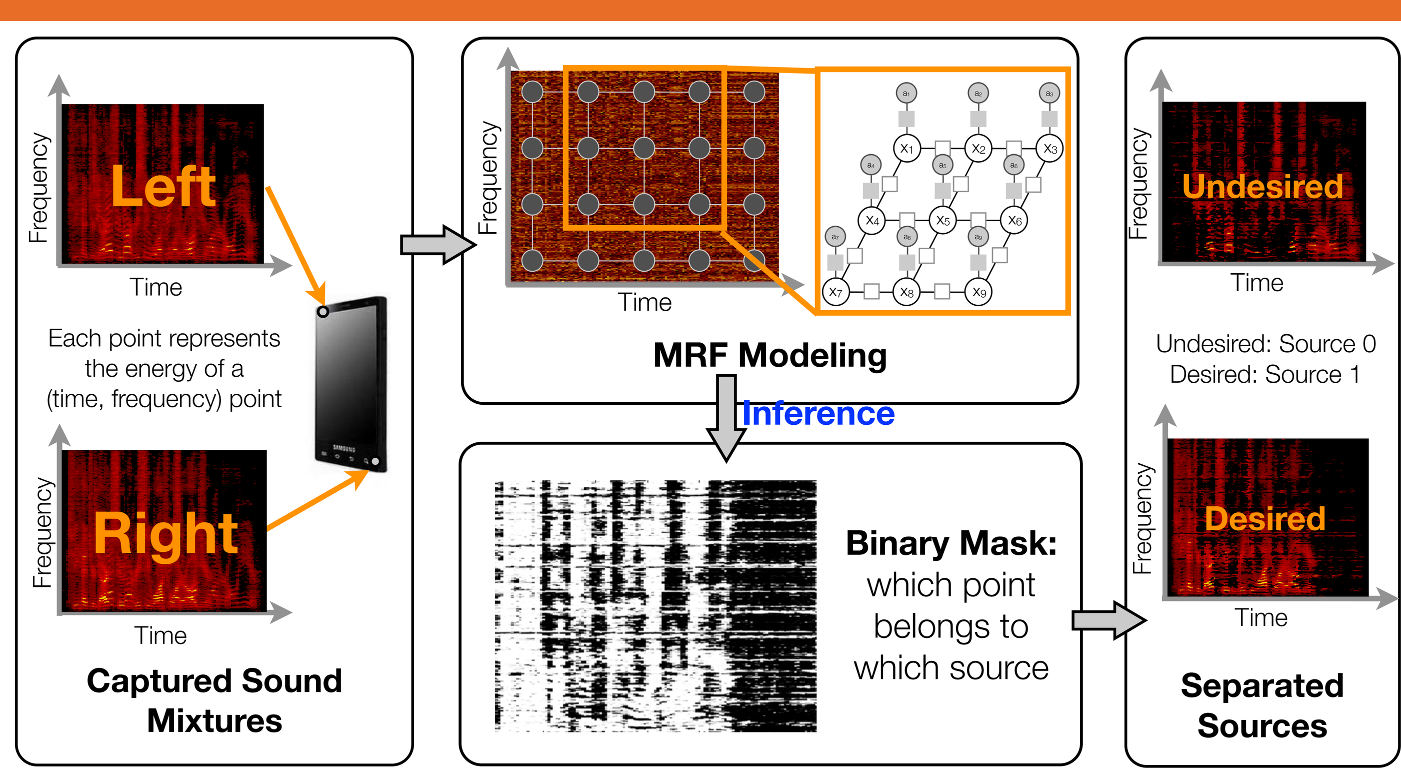
Objective

Develop High Performance and Low-power Architectures for Inference on Probabilistic Graphical Models

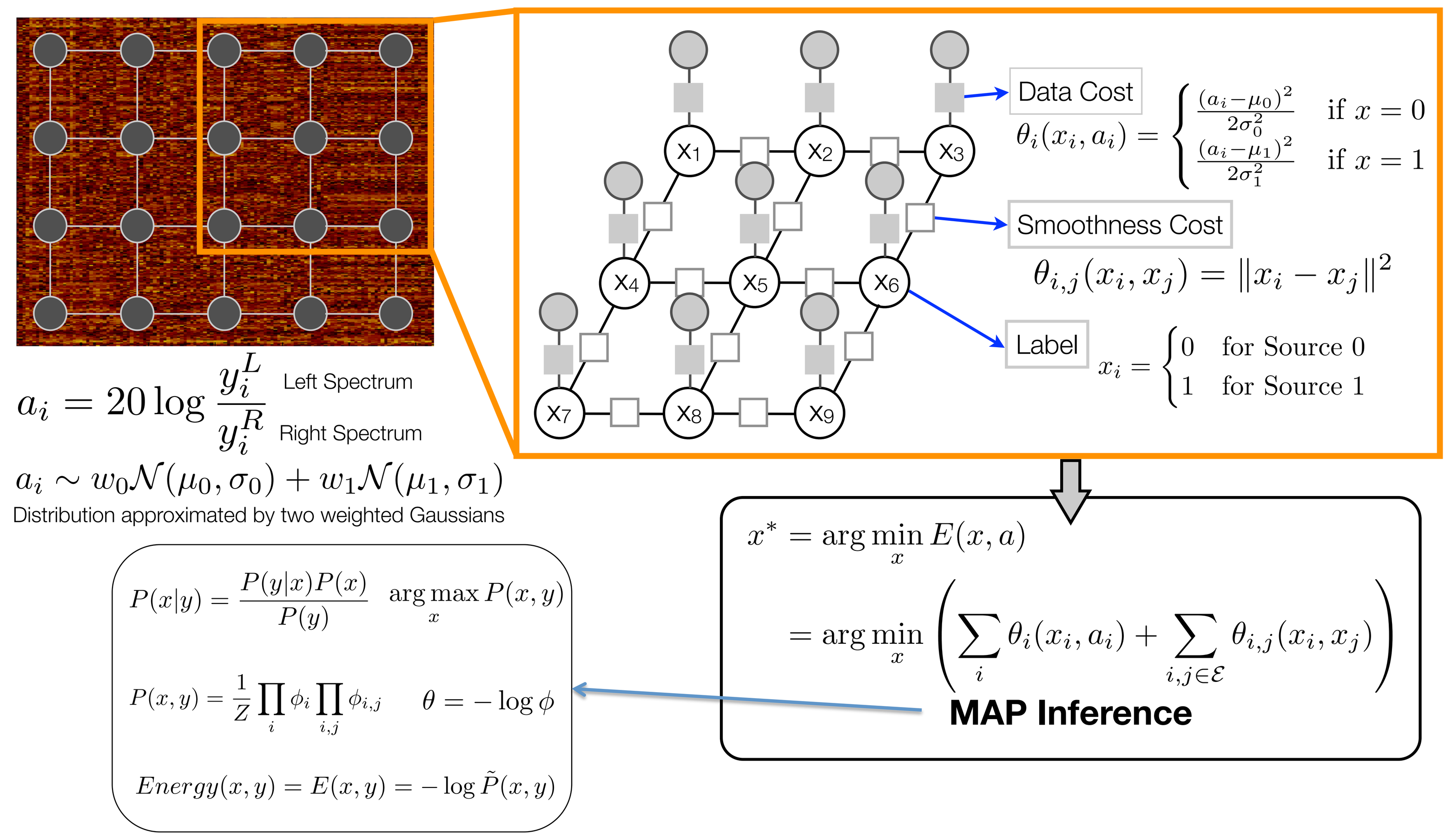
- MRF is a general framework to solve ML inference applications
- An accelerator can possibly boost up all apps in this category



Task: How to Separate Sounds?



Mapping to Markov Random Field



Inference using Gibbs Sampler

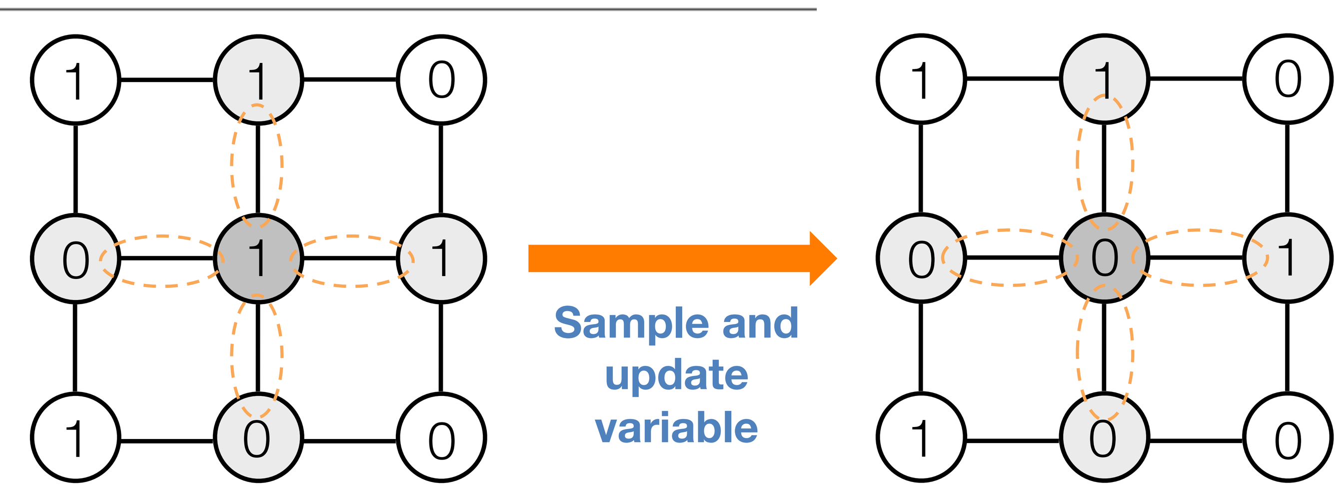
Algorithm 1 Gibbs sampling algorithm

procedure GIBBSSAMPLING(x)
  for $t = 1$ to T **do**
    for $i = 1$ to n **do**
      $x_i^{(t+1)} \sim P(x_i | x_1^{(t+1)}, ..., x_{i-1}^{t+1}, x_{i+1}^{t}, ..., x_n^{(t)})$
    end for
  end for
  return *samples*
end procedure

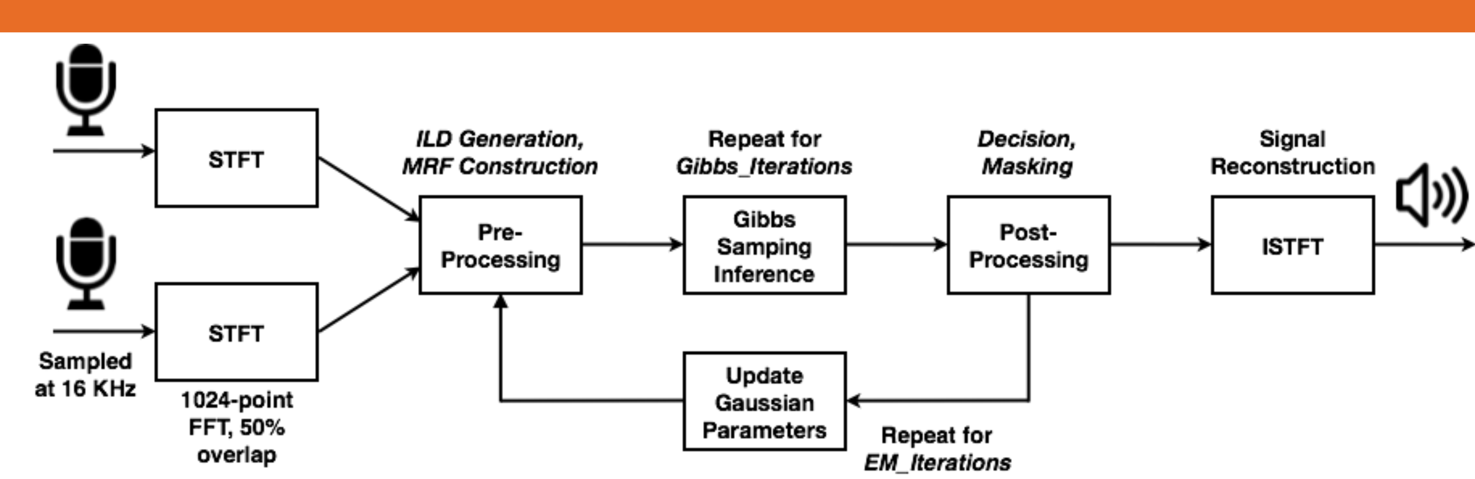
Conditional Independence via Local Markov Property

$\forall x \in V : x \perp V \backslash x \mid neighbors(x)$

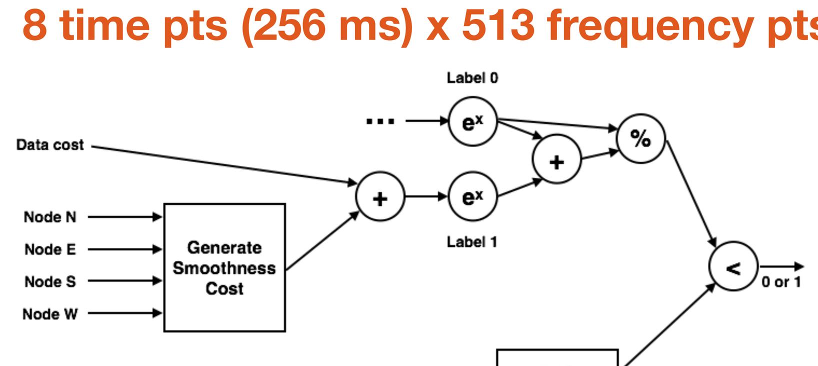
Local computation: Only 4 data access for adjacent nodes



MCMC-EM Parameter Estimation



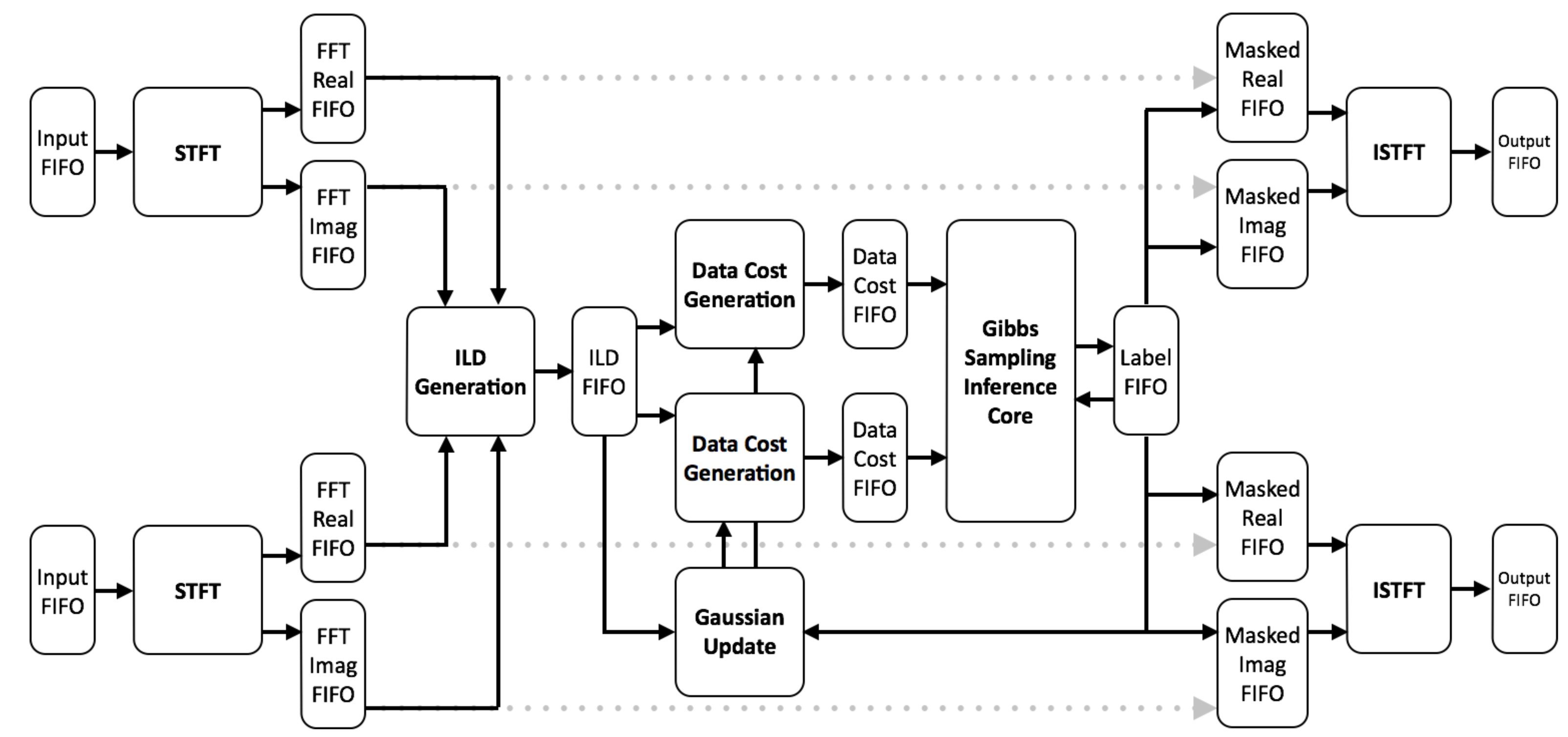
4 iterations of EM estimation and Gibbs sampling inference for MRF of 8 time pts (256 ms) x 513 frequency pts



Algorithm 1 Creating masks for source separation via MCMC-EM

procedure SOURCESEPARATIONMASK(A_i, x)
  for each EM iteration **do**
    ConstructMRF
    (E-Step) GibbsSampling:
    for $t = 1$ to max iteration T **do**
      for each node $i = 1$ to I **do**
        $x_i^{(t+1)} \sim P(x_i | x_1^{(t+1)}, ..., x_{i-1}^{t+1}, x_{i+1}^{t}, ..., x_n^{(t)})$
      end for
    end for
    (M-Step) UpdateGaussians $\mu_{0,1}$ and $\sigma_{0,1}$
  end for
  return x
end procedure

Scalable Iterative Architecture



- Gibbs sampling inference core is a parameterized pipeline that can be easily extended to multiple parallel pipelines
- Uses inference results from previous frame for faster convergence

Hardware Results

Latency Requirements

- ITU requirement 200 ms – LTE latency 160ms = 40ms: need < 40ms

Software References

- Intel Xeon X6550: ~64ms latency = too slow!
- ARM Cortex-A9 takes 23.32s to run 4s of audio with est. peak power of 3.67 Watts = slower and too power much for mobile!

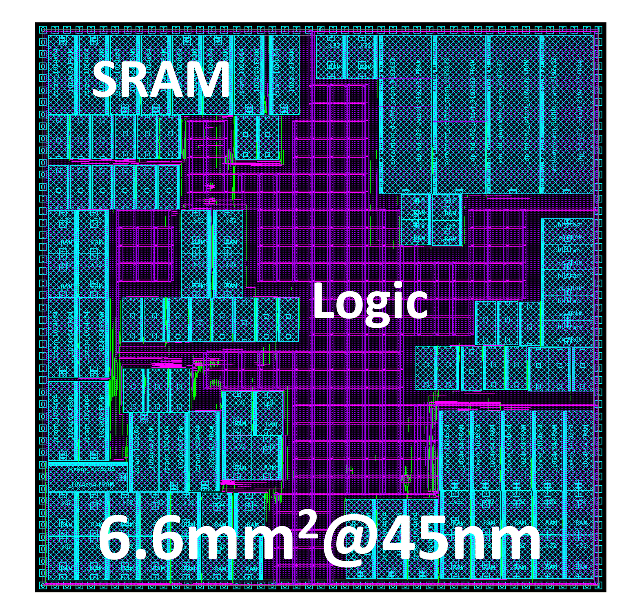
FPGA Resource	FPGA Utilization
Slice Register	101314 / 207360 (48%)
Slice LUT	90019 / 207360 (43%)
Slice LUT FF	119418 / 207360 (57%)
BRAM	115 / 288 (40%)
DSP	36 / 192 (18%)

FPGA Platform

- Convey HC-1 w/ Xilinx Virtex5 FPGAs
- 150 MHz, 207 KB SRAM
- 6.7 dB Signal-to-Distortion-Ratio
- **1.6 ms real-time latency, 20X speedup**

Virtual ASIC Design @ 45nm

- Quite small < 10 M gates
- 469 mW at 150 MHz (404 mW from SRAM)
- 70 mW at 20 MHz (meets latency)
- **52X reduced power vs. ARM Cortex-A9**



Summary

- A working HW implementation of source separation in a mobile form-factor using MRFs and Gibbs inference
- FPGA implementation running at **150 MHz** with **207 KB of RAM**, 64-bit memory width and 96 Kb/s of bandwidth requirement which is mobile-friendly
- Good **Signal-to-Distortion Ratio of 6.7 dB** with better auditory performance and very low **1.601ms latency**, a **20X speedup** from input to output sound stream, well below latency req for human hearing on mobile phone
- Virtual ASIC design (45 nm) **power estimate 70 mW running at 20 MHz to meet latency req, a 52X power reduction** vs. ARM Cortex-A9 mobile software reference

NSF
STARnet
SONIC