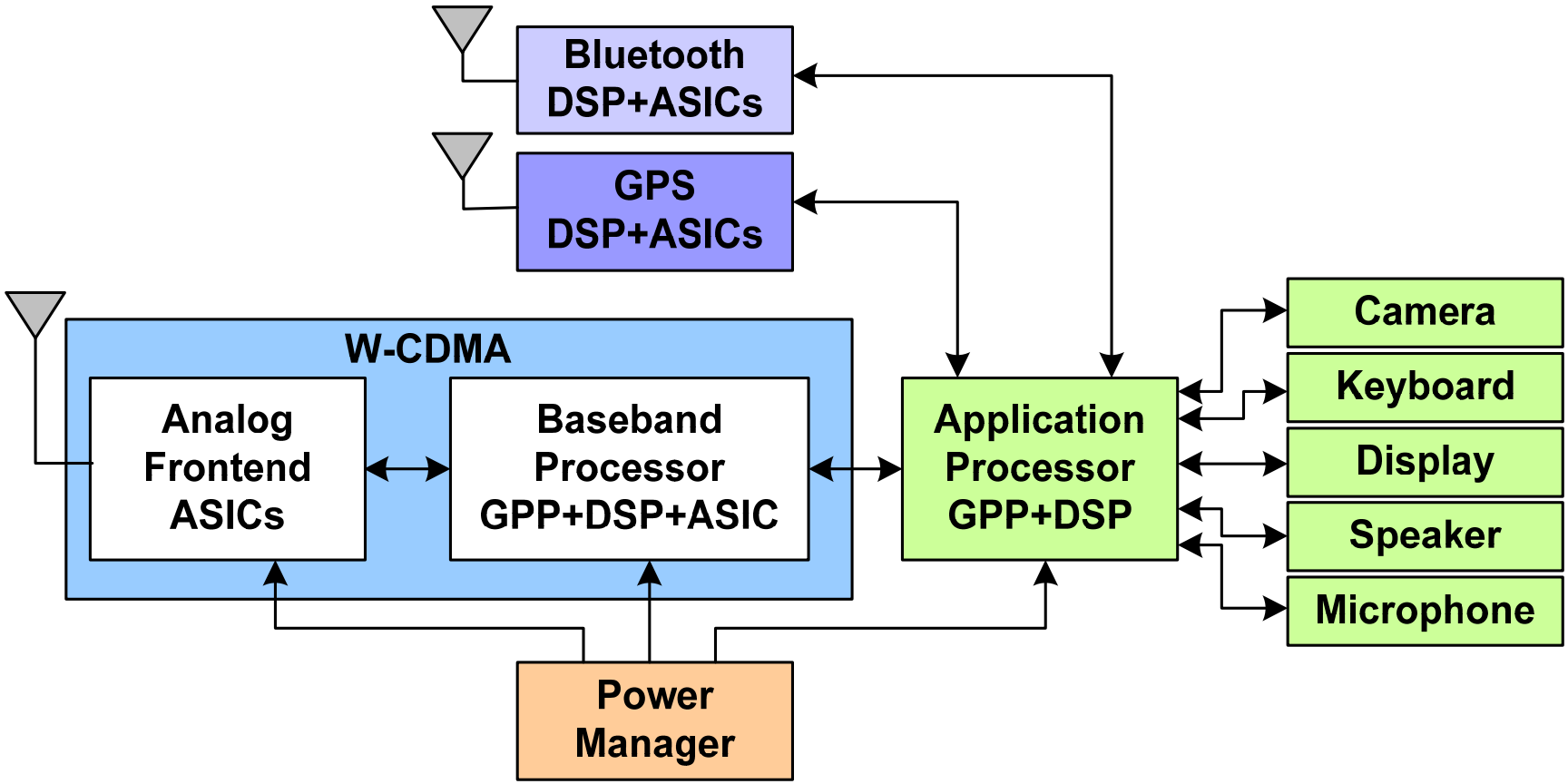
A Scalable Low-power Architecture For Software Radio

Scott Mahlke

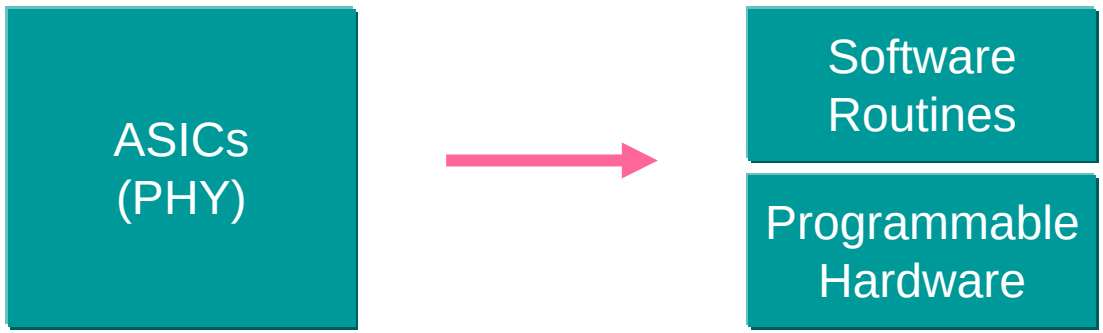
Collaboration between:
University of Michigan, Arizona State University, and ARM Ltd.

Anatomy of a Cellular Phone



Software Defined Radio (SDR)

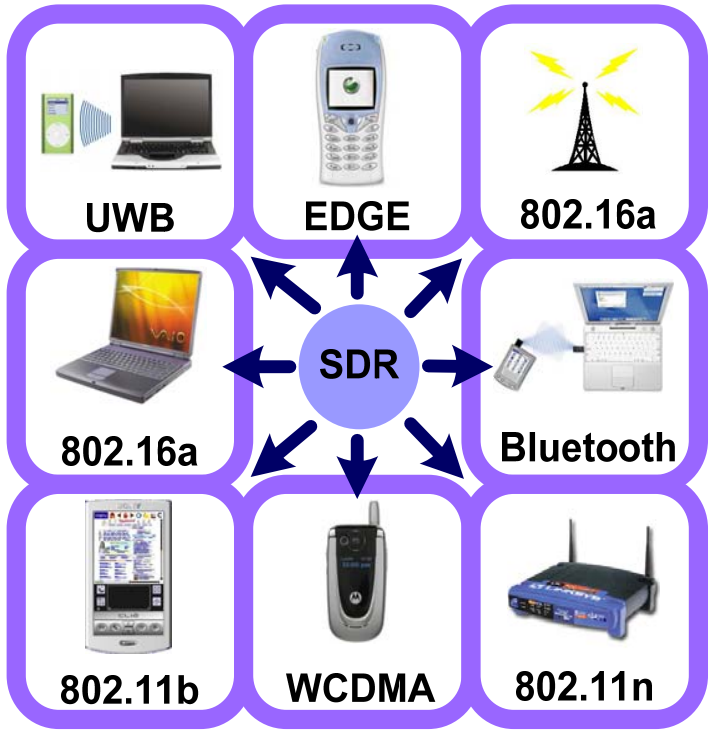
- Use software routines instead of ASICs for the physical layer operations of wireless communication system



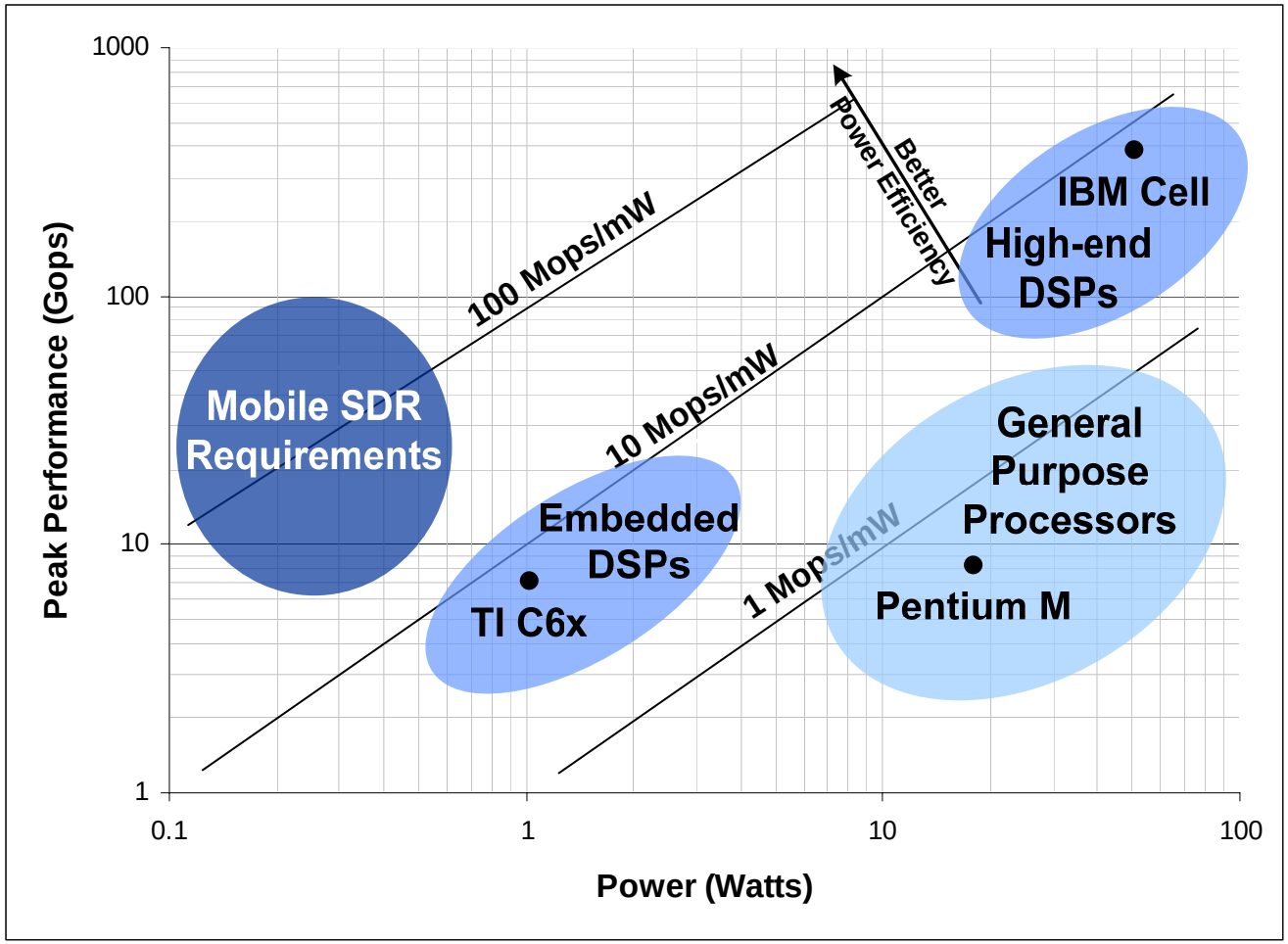
- Rest of the talk
  - Characteristics of SDR algorithms
  - SODA architecture for power-efficient SDR
  - Compilation challenges and approach

Advantages of SDR

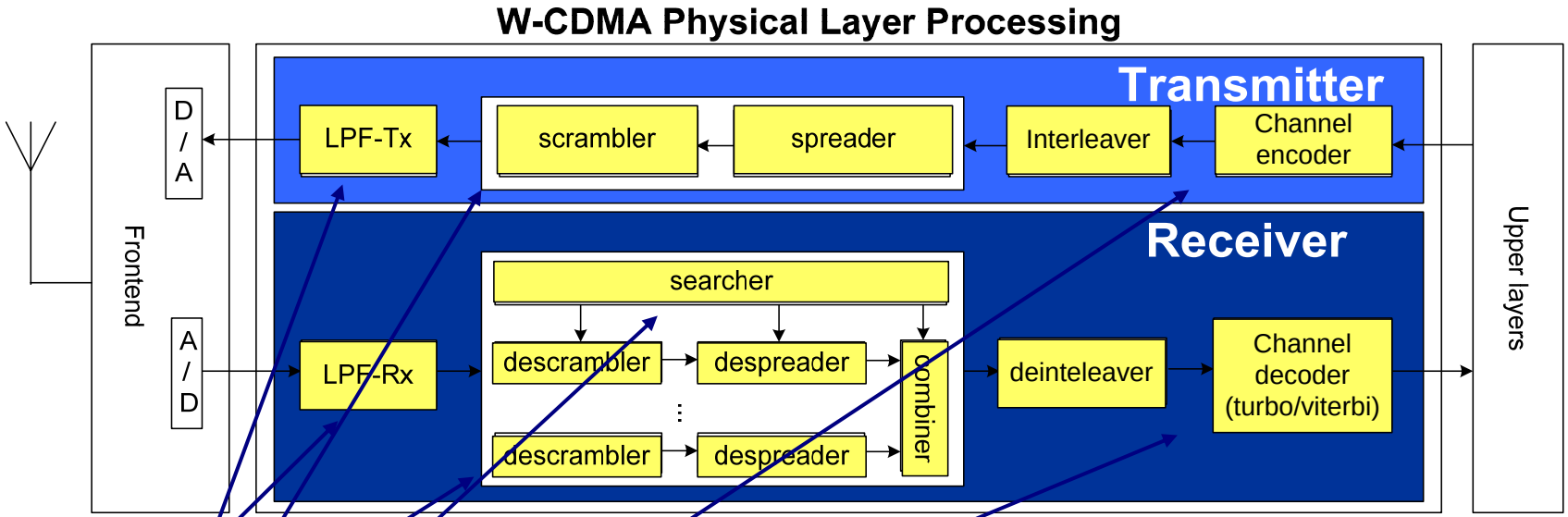
- Lower costs
  - Platform longevity, higher volume
  - SW has lower development costs
- Time to market
  - Future protocols will have complex implementations
  - Overlap testing/development cycles
- Adaptability
  - Standards change over time
  - Multi-mode operation
  - Sharing hardware resources



Why is SDR Challenging?



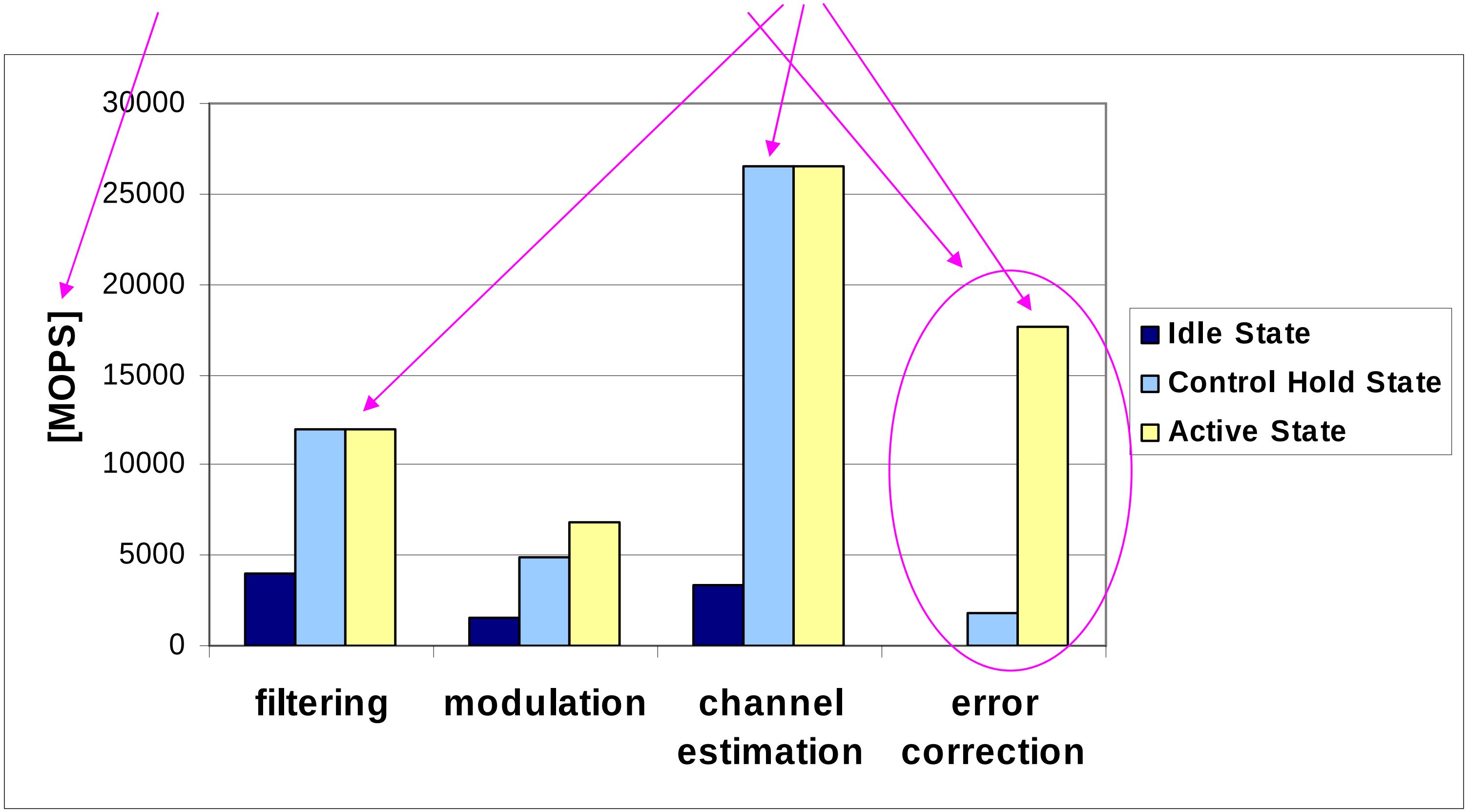
The Anatomy of Wireless Protocols



1. **Filtering:** suppress signals outside frequency band
2. **Modulation:** map source information onto signal waveforms
3. **Channel Estimation:** Estimate channel condition for transceivers
4. **Error Correction:** correct errors induced by noisy channel

W-CDMA Workload Profile

- [illegible] workloads



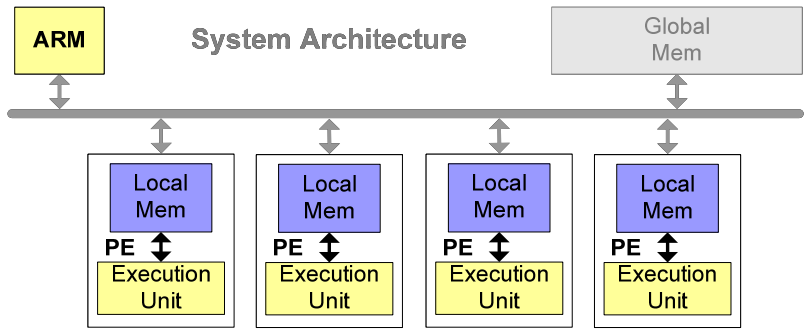
SDR Kernel Characteristics

- 8 to 16-bit precision
- Vector operations
  - long vectors
  - constant vector size
- Static data movement patterns
- Scalar operations

Kernels	Type of Computation	Vector Width
W-CDMA		
Filter	Vector	64
Modulation	Vector	2560
Channel Est.	Vector	320
Error Correction	Mixed	8 or 256
802.11a		
Filter	Vector	33
Modulation (FFT)	Vector	64
Channel Est.	Mixed	16
Error Correction	Mixed	64

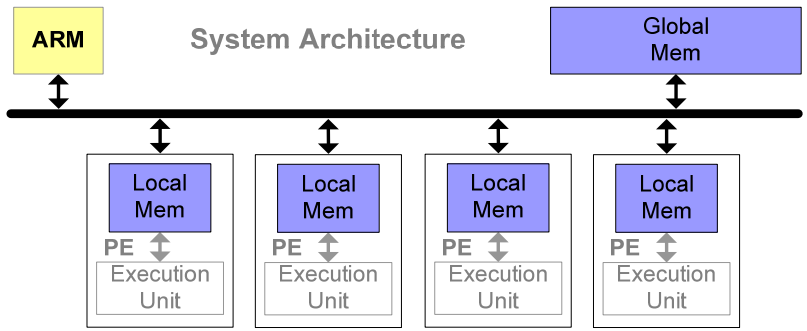
SODA System Architecture for 3G

- 4 PEs
  - static kernel mapping and scheduling
  - SIMD+Scalar units
- 1 ARM GPP controller
  - scalar algorithms and protocol controls



SODA System Architecture for 3G

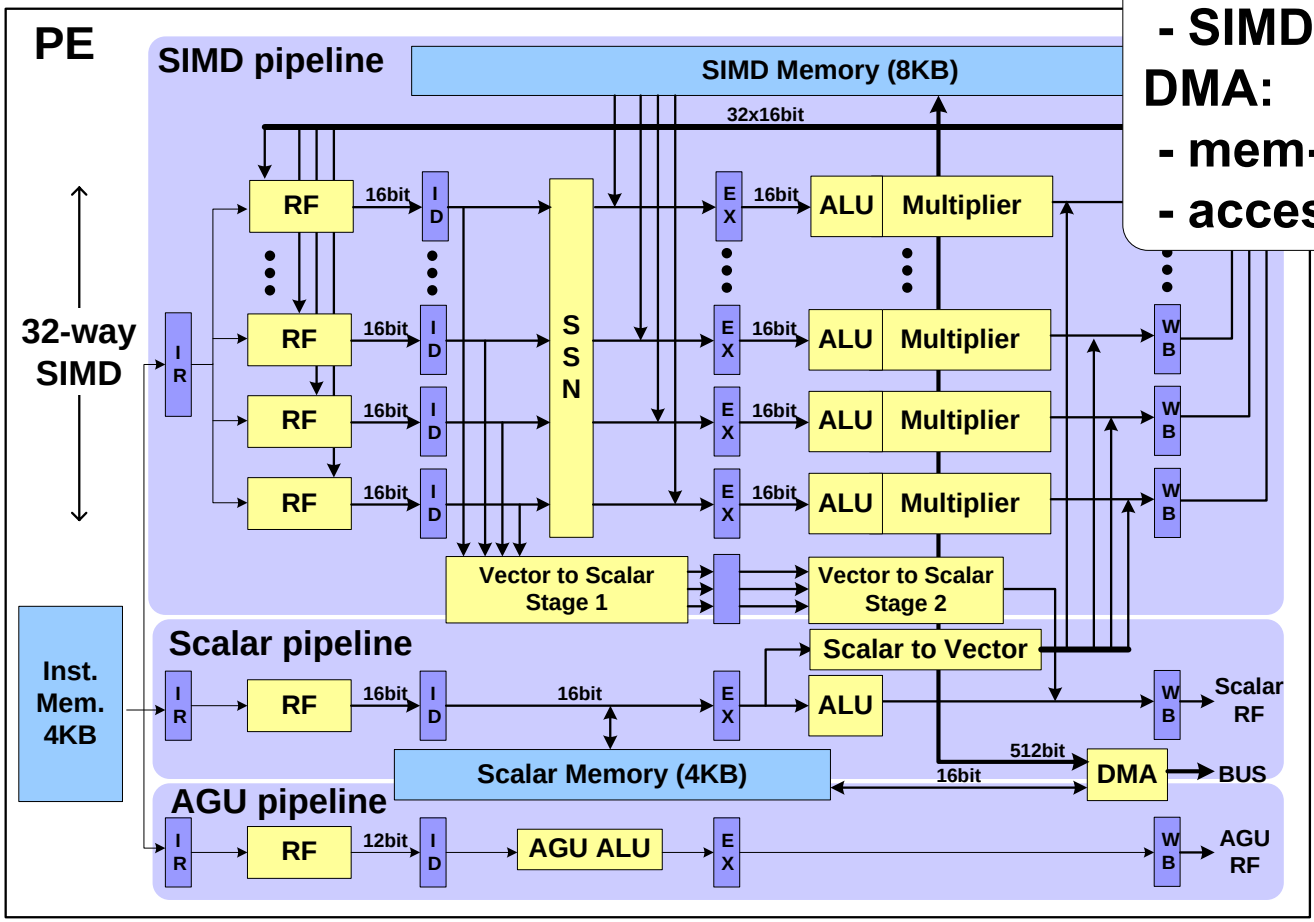
- 2-Level scratchpad memories
  - 12KB Local scratchpad memory for stream queues
  - 64KB global scratchpad memory for large buffers
- Low-throughput shared bus
  - 200MHz 32-bit bus
  - inter-PE communication using DMA



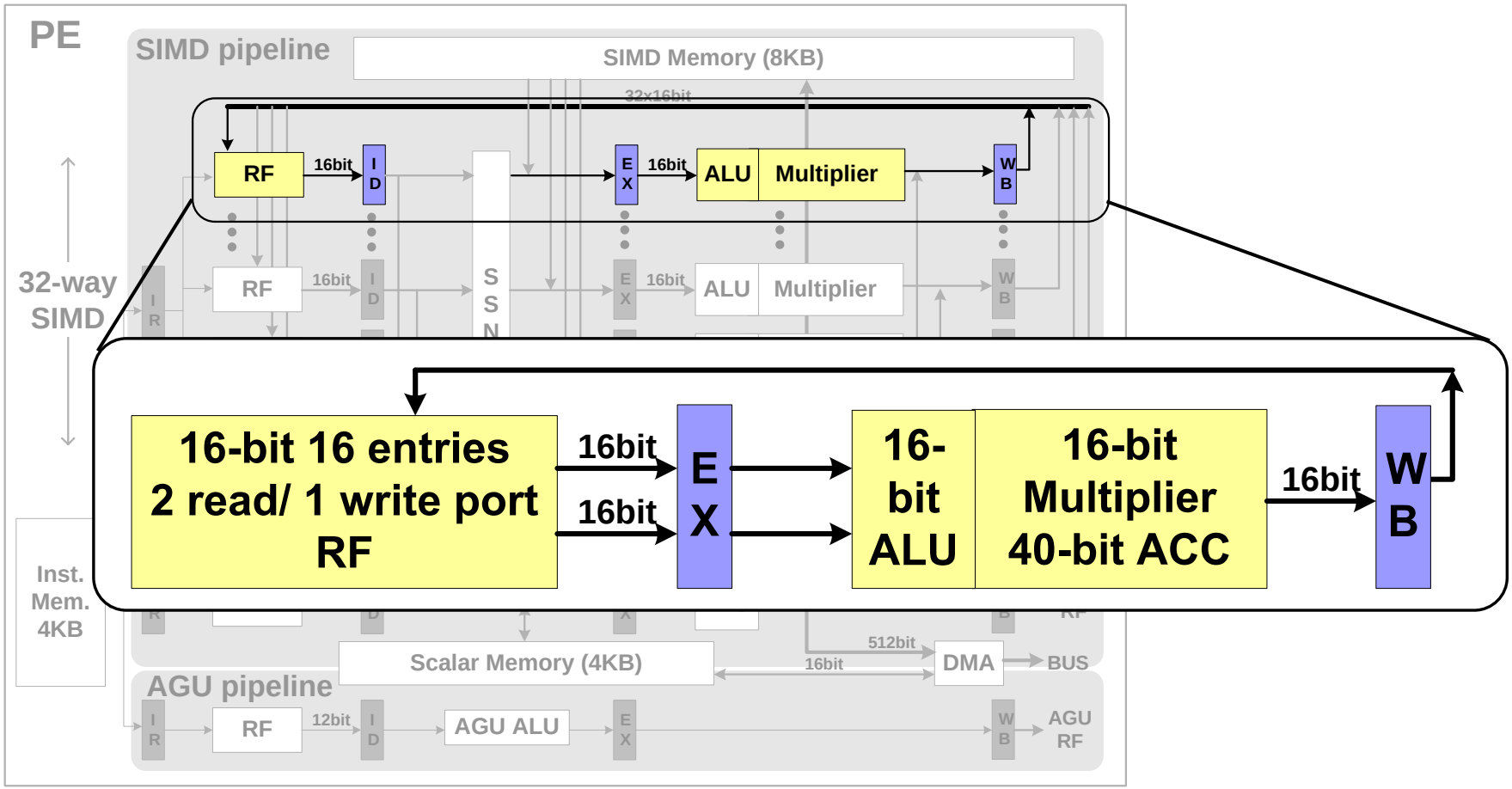
SODA PE Architecture

2 issue LIW (400MHz)
- **SIMD + (Scalar or AGU)**

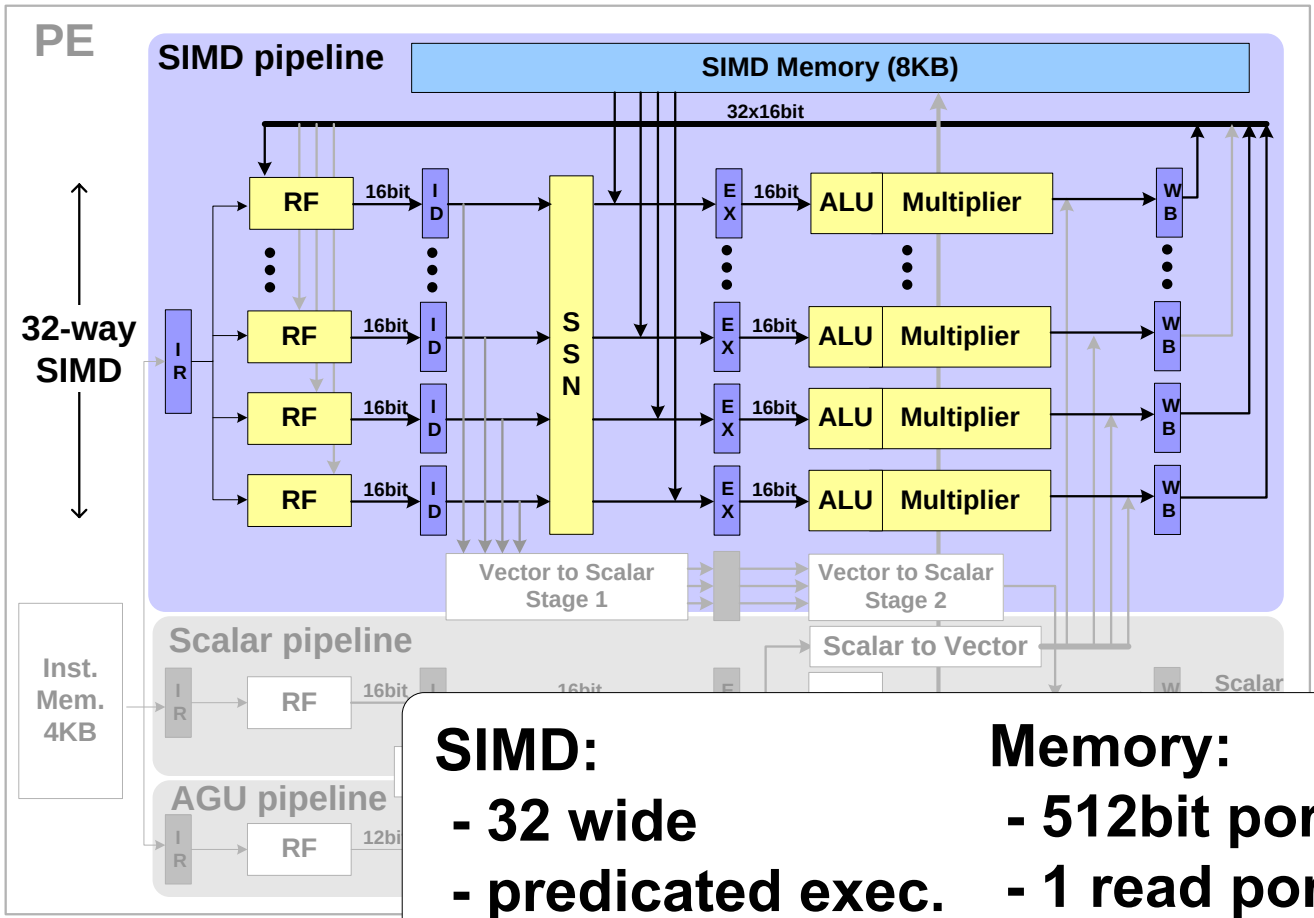
DMA:
- **mem-to-mem transfer**
- **access global memory**



SODA PE SIMD Pipeline



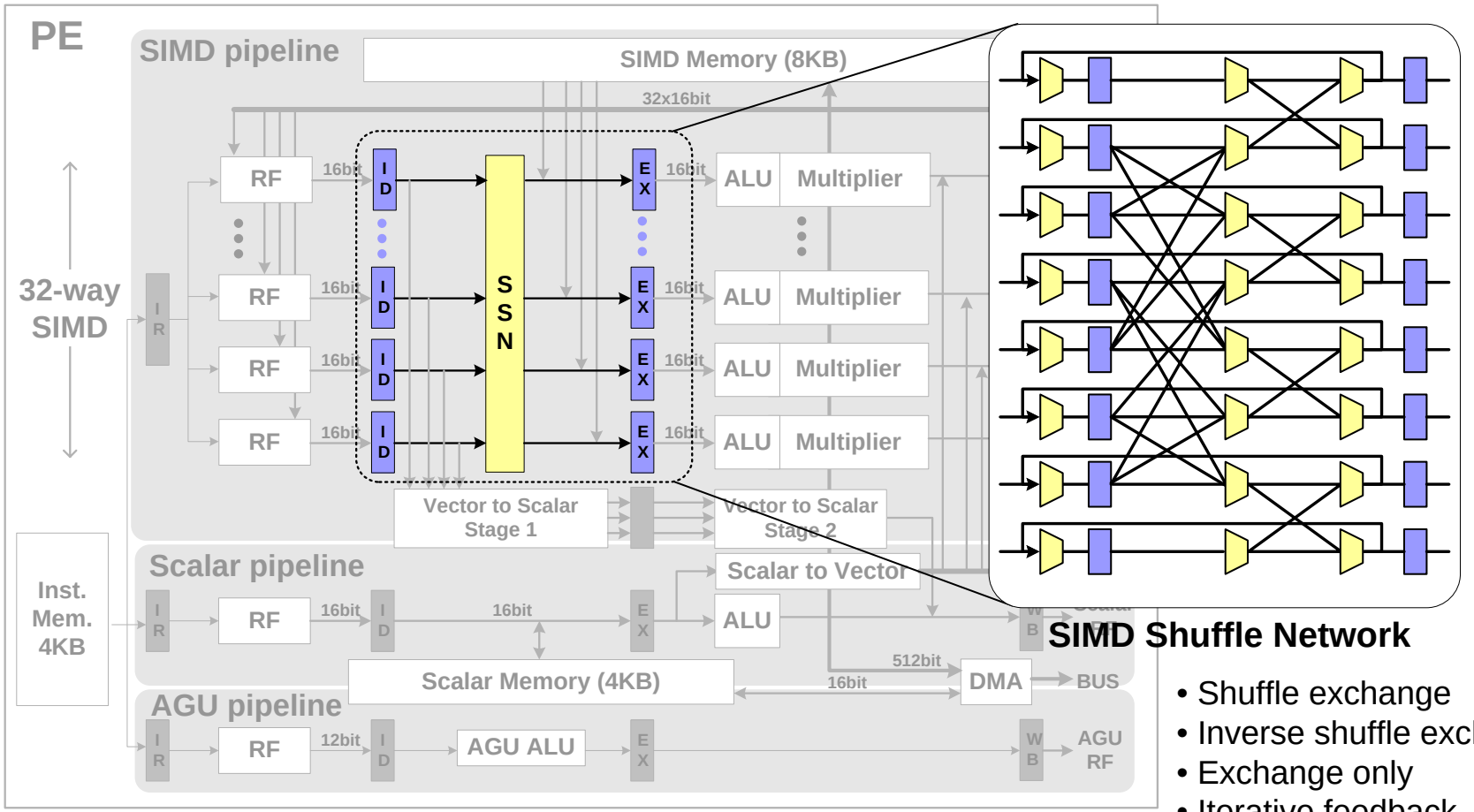
SODA PE SIMD Pipeline



SIMD:
- 32 wide
- predicated exec.
- predicated neg.

Memory:
- 512bit port
- 1 read port
- 1 write port
- 8 KBytes

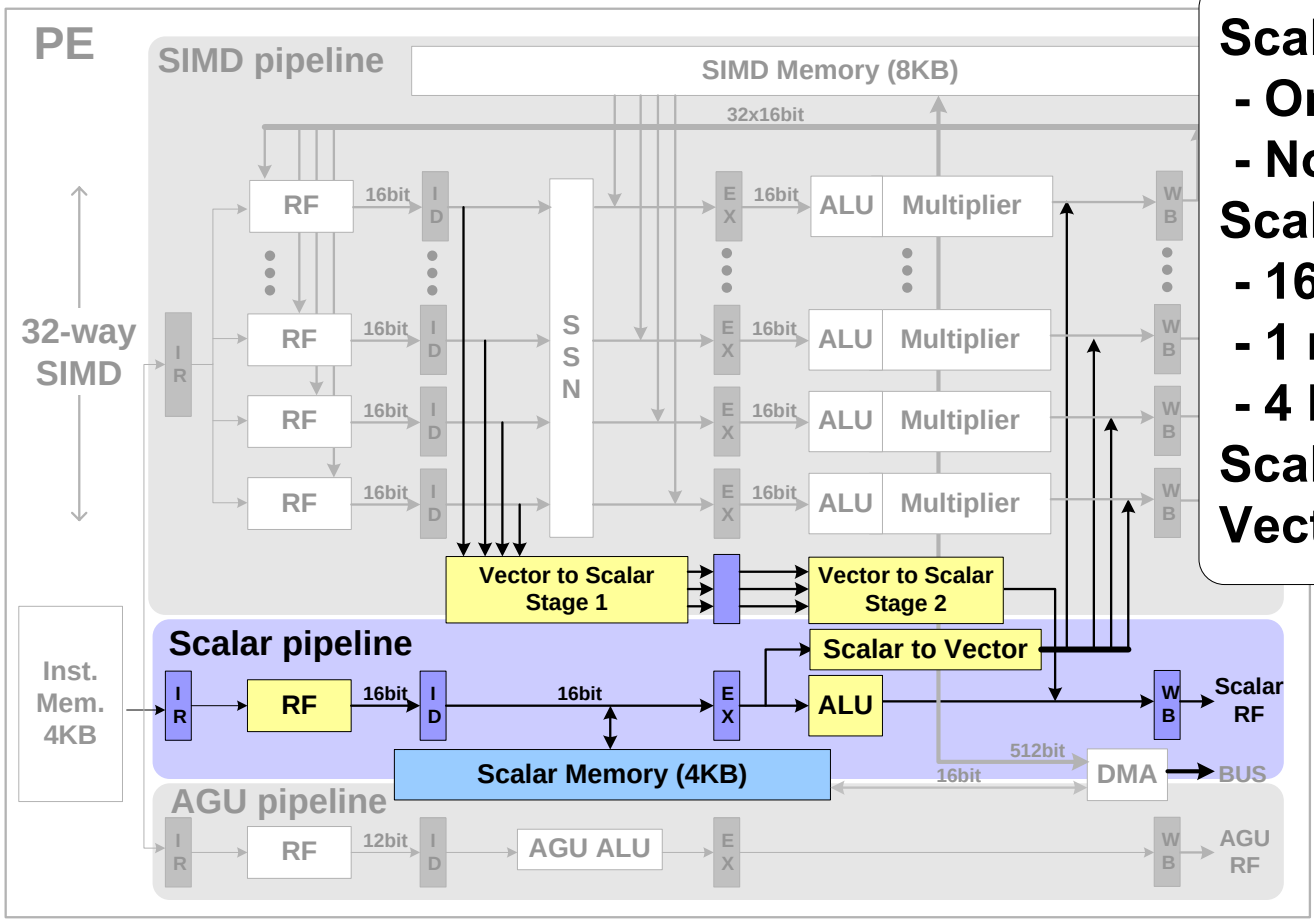
SODA PE SIMD Shuffle Network



SIMD Shuffle Network

- Shuffle exchange
- Inverse shuffle exchange
- Exchange only
- Iterative feedback

SODA PE Scalar Pipeline



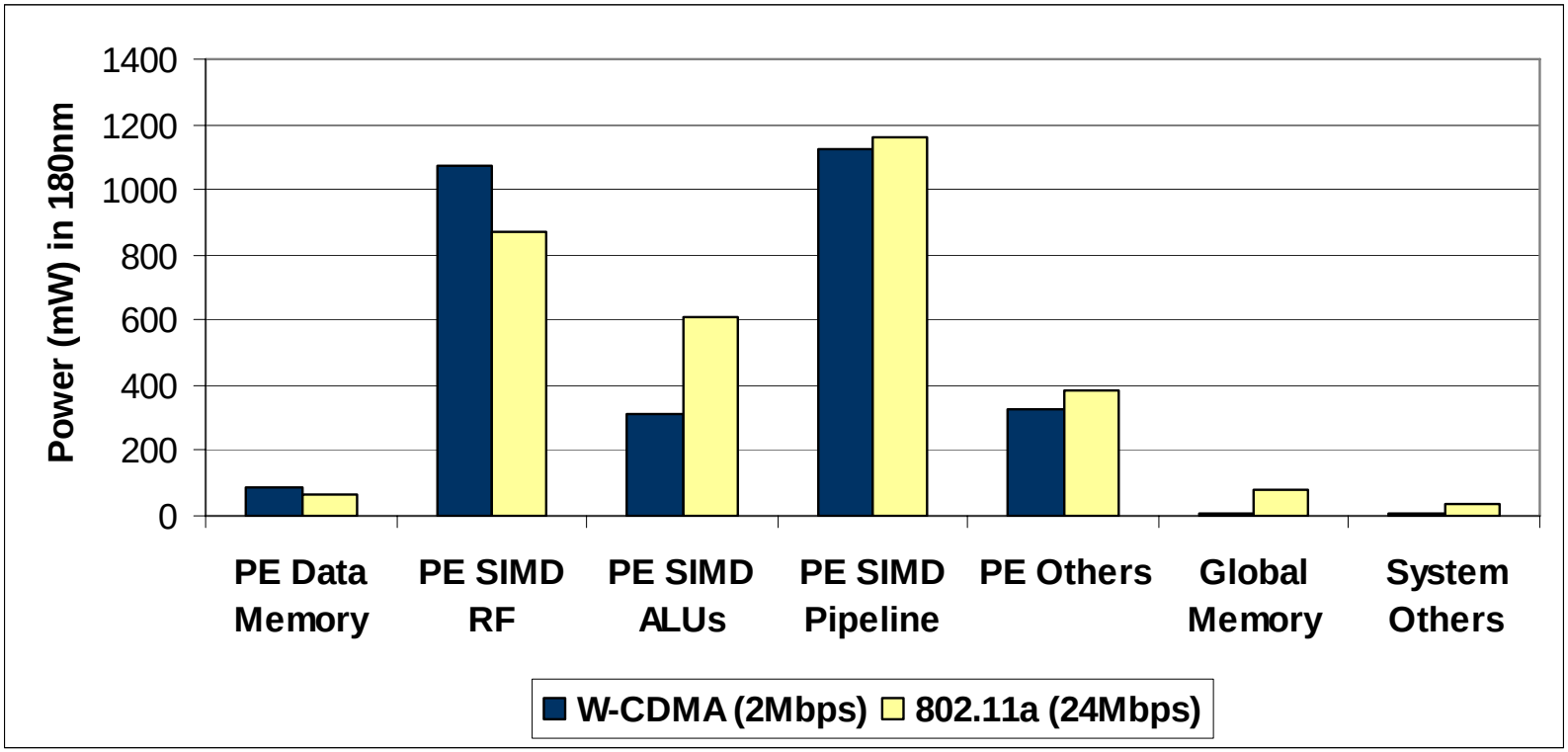
Scalar:
- One 16-bit datapath
- No mult unit

Scalar memory:
- 16bit port
- 1 read/write port
- 4 KBytes

Scalar-to-Vector

Vector-to-Scalar

Power Consumption at 180nm



- 180nm → ~ 3 W, 26.6 mm^2
- 90nm (est) → ~ 0.5 W, 6.7 mm^2

SDR Compilation Strategy

- Two level application description
  - System-level: Concurrent tasks extracted from “C + channels + attributes”
  - Kernel-level: Data parallelism extracted from “C + vectors + Matlab operators”
- System compilation – Task level parallelism
  - Generates tasks, schedules, communication stubs, DMA requests, timing assertions, synchronization, debug support
- Kernel compilation – Data level parallelism
  - Lower virtual DLP to physical implementation

Stylized Automatic Parallelization

```
void main() {
  for(int i=0; i<N; ++i) {
    int x = spread(w[i]);
    int y = scramb(x);
    fir(y);
  }
}
```

```
void t1() {
  for(int i=0; i<N; ++i) {
    int x = spread(w[i])@P0;
    putchannel(X,x); }}
void t2() {
  for(int i=0; i<N; ++i) {
    int y = scramb(getchan(X))@P
    putchannel(Y,y); }}
void t3() {
    for(int i=0; i<N; ++i) {
    fir(getchannel(Y)) @ P2; }}
main() {
  fork (t1);
  fork (t2);
  fork (t3);
}
```

1. Task assignment

2. Decompose tasks

Kernel Level Compilation

```
template<class T, TAPS, BSIZE>
kernel FIR {
  vector<T, TAPS> z;
  vector<T, TAPS> coeff;

  void set_coeff(vector<T, TAPS> c)
  { coeff = c; }

  void run(channel<T, BSIZE> inbuf,
           channel<T, BSIZE> outbuf)
  {
    T in, out;

    for (i = 0; i < BSIZE; i++) {
      in = inbuf.pop();
      z += coeff * in;
      out = z[0];
      outbuf.push(out);
      z = (z(1:TAPS-1),0);
    }
  }
};
```

1. object declaration

```
int main() {
  FIR<int8, 64, BUF_SIZE>   fir65r;
  ...
}
```

2. operation translation

```
// in = inbuf.pop();
sld(sin, inbuf);
agu_incr(inbuf, sizeof(int8));

// z += coeff * in;
vdup(vin, sin);
vmul(vt1, _fir65r_coeff_v0, vin);
vmul(vt2, _fir65r_coeff_v1, vin);
vadd(_fir65r_z_v0, _fir65_z_v0, vt1);
vadd(_fir65r_z_v1, _fir65_z_v1, vt2);
```

Final Thoughts

- 2G and 3G SDR solutions are achievable in 90nm
  - 3.9G → 4-10x more performance with mW power consumption
- Core technologies for future networks
  - OFDM → 64 – 2048 point FFT
  - MIMO – use of multiple antennas for transmission/reception
  - Low density parity check codes
- Key insight: SDR requires innovation across algorithm, software and hardware
- SDR platforms offer low-cost, longevity, and adaptability