A GALS Many-Core Heterogeneous DSP Platform with Source-Synchronous On-Chip Interconnection Network

Anh Tran, Dean Truong and Bevan Baas
University of California, Davis

NOCS '09
May 13, 2009

Outline

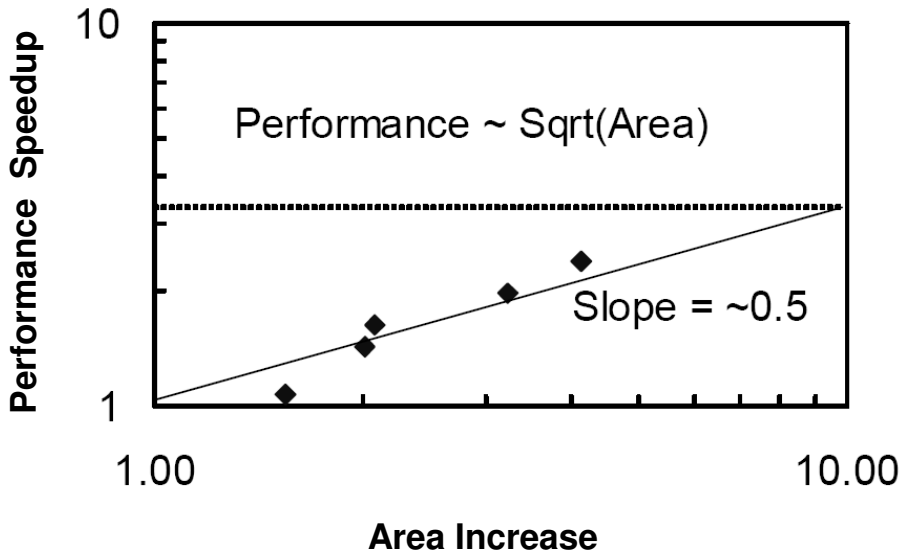
- Motivation
- Design of a GALS many-core DSP platform
- A GALS-compatible source-synchronous interconnect network
- Test chip implementation
- Mapping application case study: 802.11a/g baseband receiver
- Conclusion

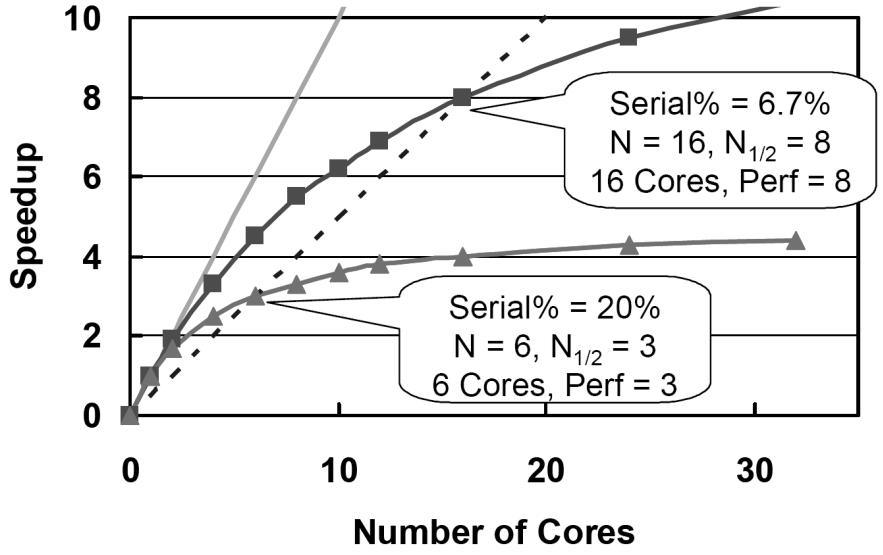
Outline

- Motivation
- Design of Our GALS many-core DSP platform
- The GALS compatible source-synchronous interconnect network
- Test chip implementation
- Mapping application case study: 802.11a/g baseband receiver
- Conclusion

Emergence of DSP multi-core platforms

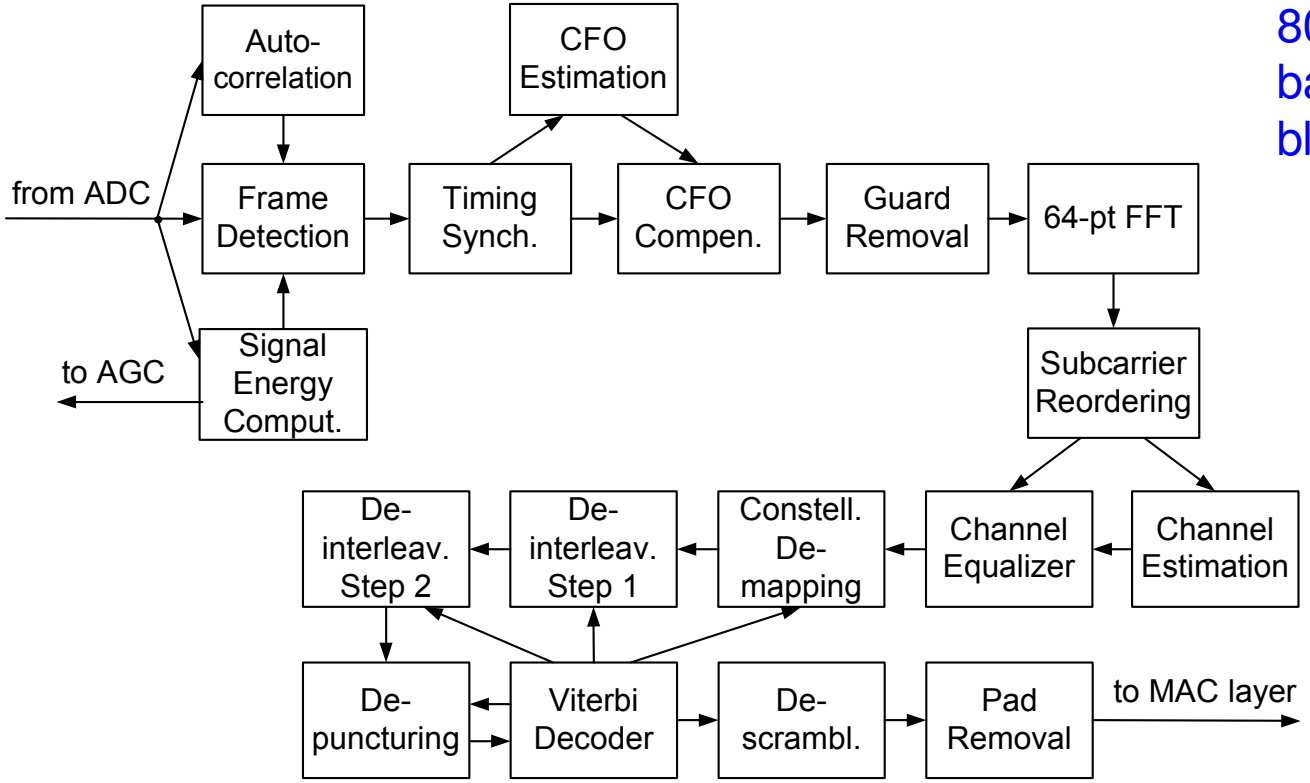
- Low design cost and short time-to-market favor programmable and reconfigurable DSP platforms
- Continually shrinking transistor sizes enable multi/many-core designs
- Pollack's Rule: many small cores outperform a few large cores for the same silicon area
- Amdahl's Law: performance speedup depends strongly on available parallelism





[S. Borkar, DAC, 2007]

High parallelism and deterministic connections in DSP Applications

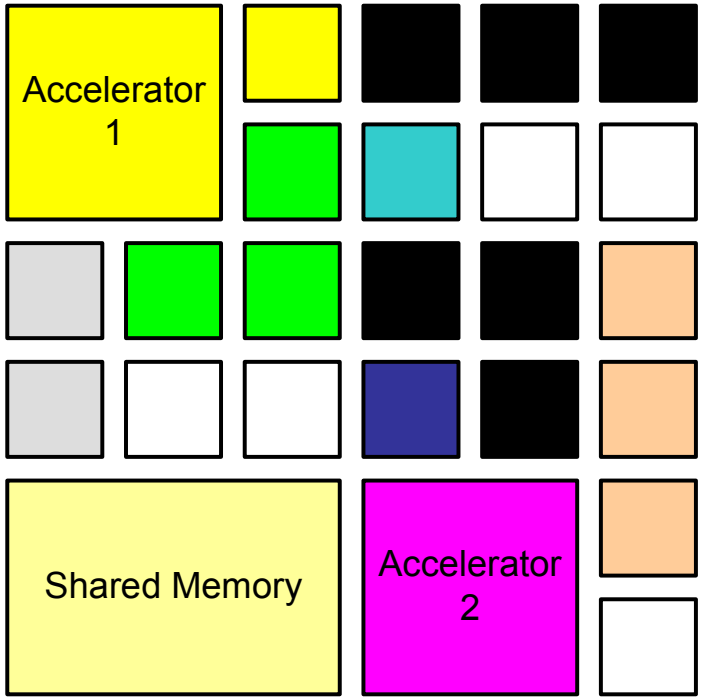


802.11a/g baseband receiver block diagram

- A high degree of task-level parallelism is available directly from task graphs for many DSP, multimedia, and embedded applications
- Often possible to map each task to one/few small processors
- A statically-configured interconnection network may be sufficient

Energy advantages of GALS, many-core and heterogeneous architectures

- Independent local clock oscillators
  - Eliminate difficult to design, power-hungry global clock trees
  - Allow use of different frequencies (and supply voltages) for processors depending on their workloads → reduce dynamic power
  - Allow complete turn off of unused processors → reduce idle power
- Support compute-intensive tasks by specific accelerators

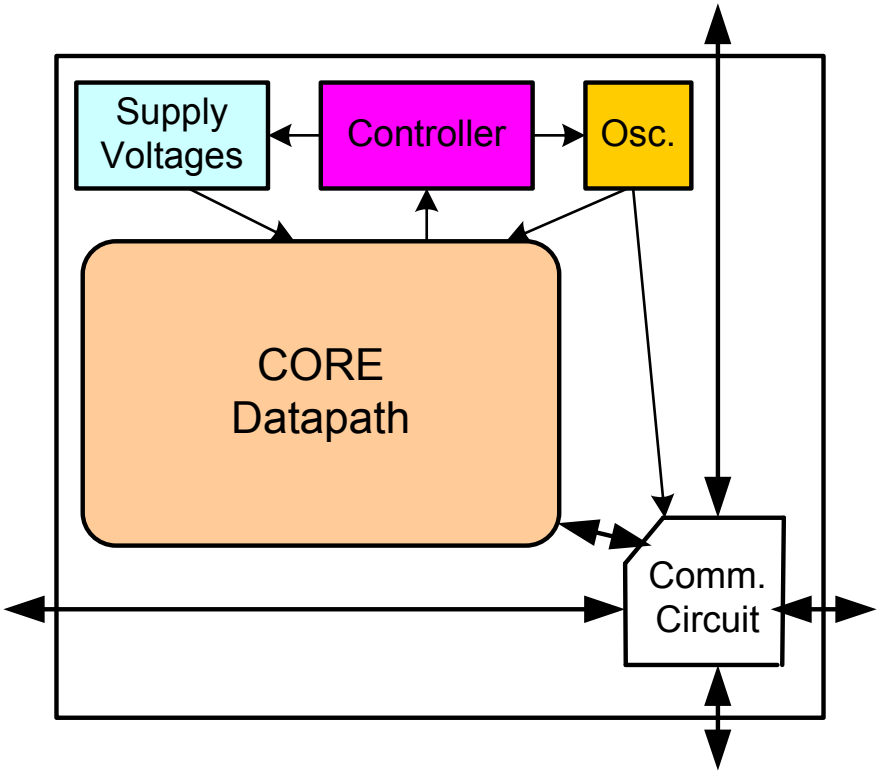


- Our approach for interconnection network of many-core heterogeneous GALS DSP platforms:
  - Static reconfigurable circuit-switched interconnects
  - Source-synchronous communication across multiple clock domains

Outline

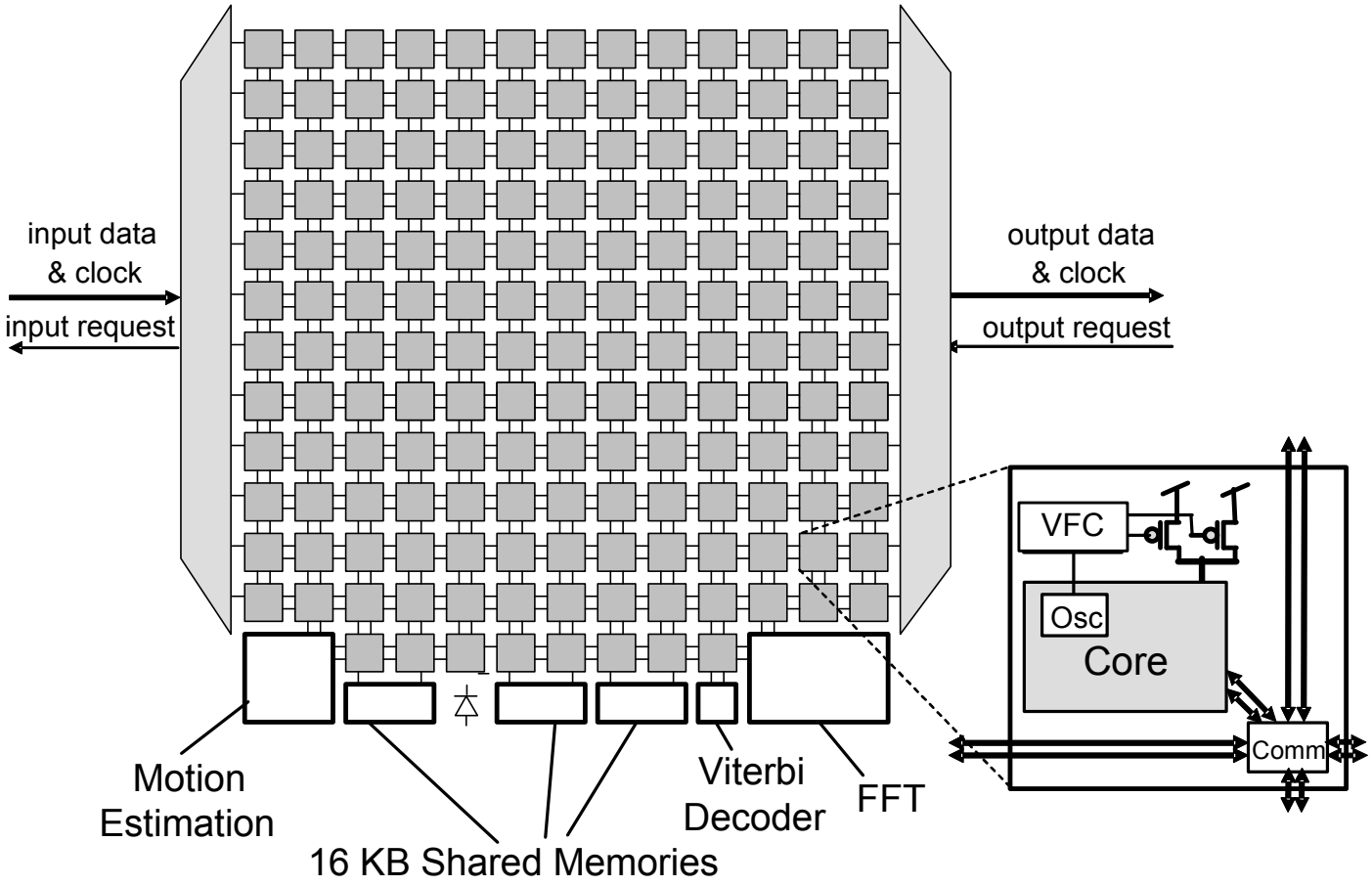
- Motivation
- Design of Our GALS many-core DSP platform
- The GALS compatible source-synchronous interconnect network
- Test chip implementation
- Mapping application case study: 802.11a/g baseband receiver
- Conclusion

Highly reusable design



- All programmable processors have identical design and physical layout
- The design of the oscillator and inter-processor communication circuitry are the same for all processing elements (PE)
  - They are designed as a generic wrapper that is reused for all PEs

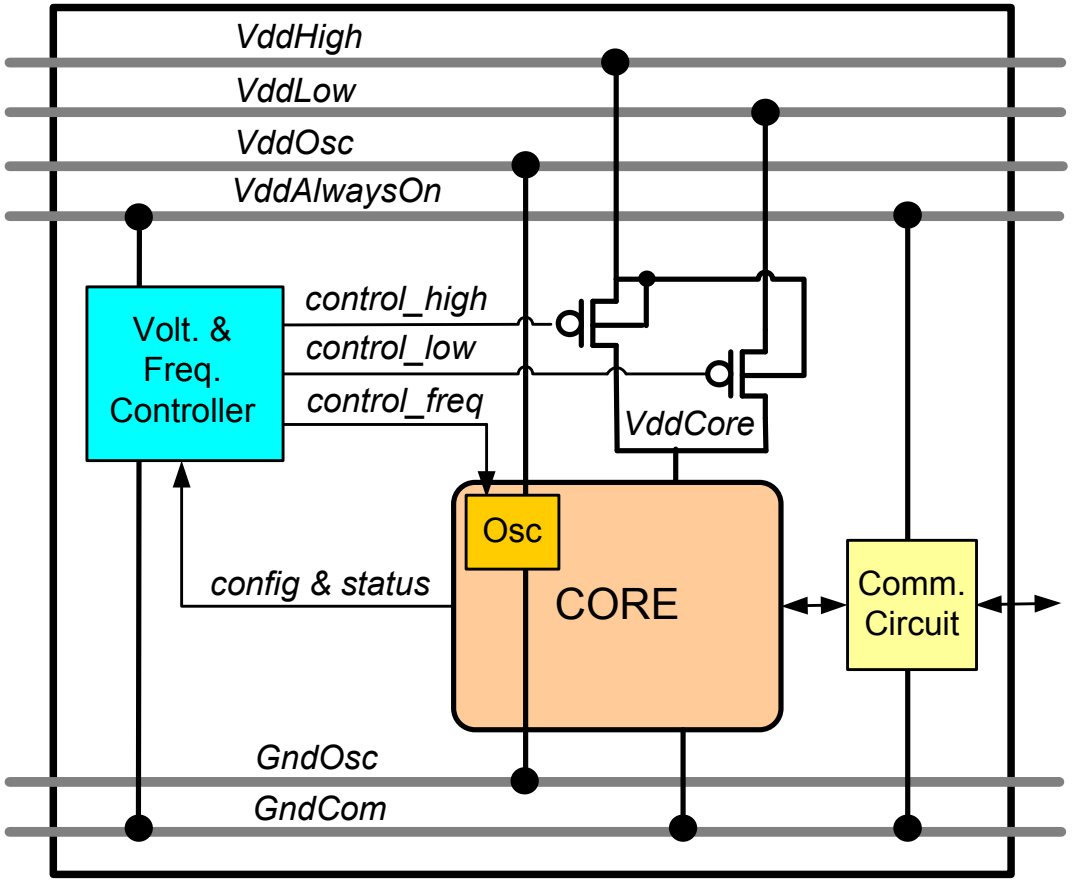
Our Platform Design



- 164 small fine-grained processors
- Three reconfigurable accelerators: FFT, Viterbi and Motion Estimation
- Three shared memory modules

Voltage and Frequency Controller

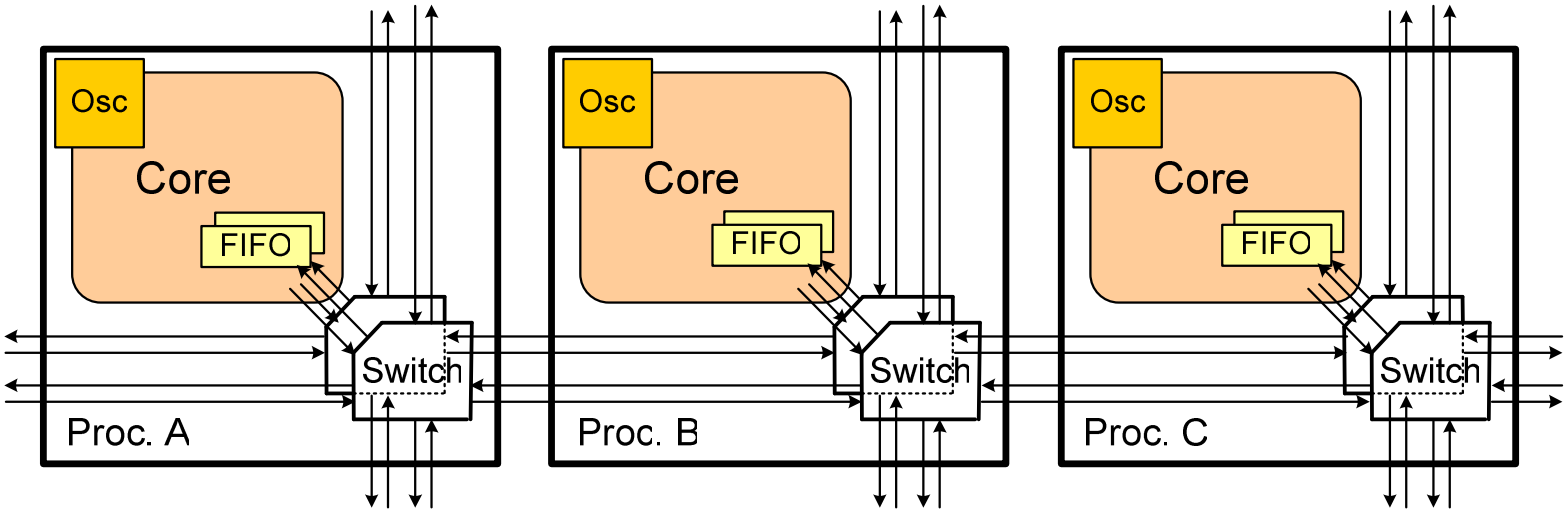
- Multiple power grids → low design cost, fast voltage switching
- Programmable ring oscillator runs on its own supply voltage for increased stability
- Supply voltage and clock frequency are set depending on the workload
  - Statically
  - Dynamically by software
  - Dynamically by hardware
- Inter-processor communication circuits run at a fixed voltage to avoid using many level shifters



Outline

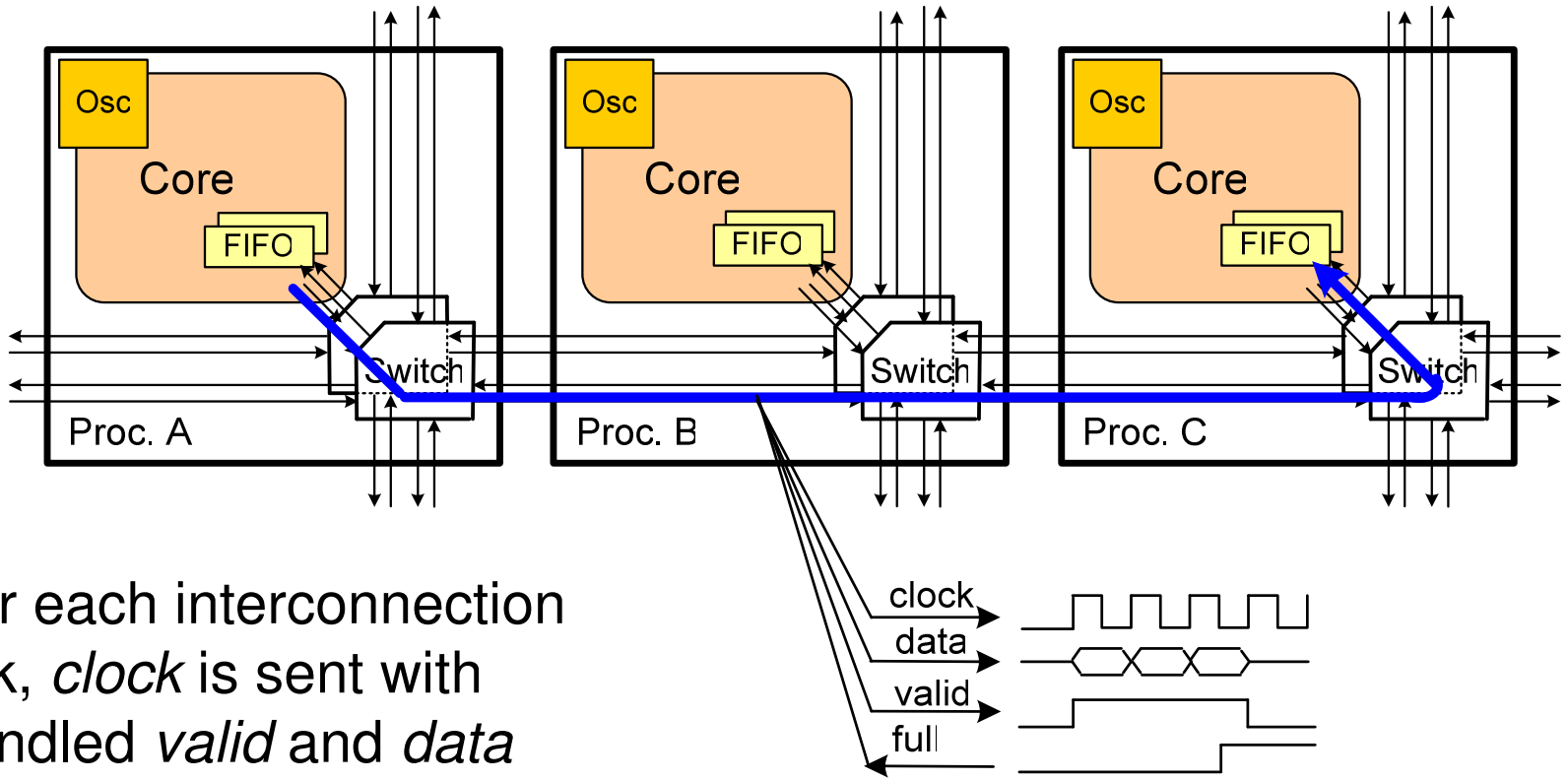
- Motivation
- Design of Our GALS many-core DSP platform
- The GALS compatible source-synchronous interconnect network
- Test chip implementation
- Mapping application case study: 802.11a/g baseband receiver
- Conclusion

2-D mesh static circuit-switched network



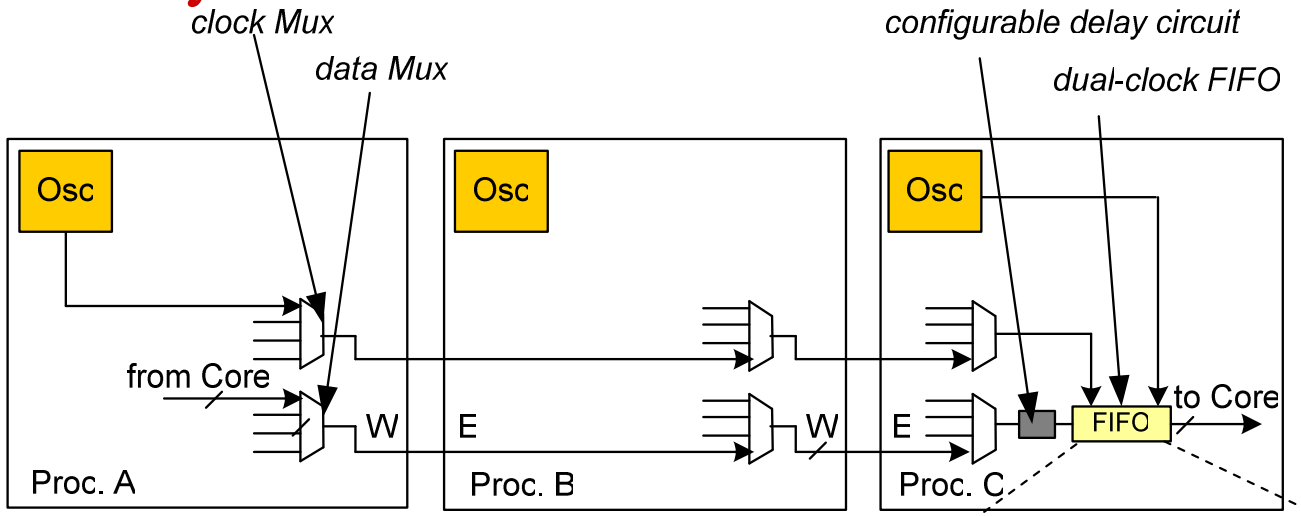
- Each switch has five ports and uses only 4-input MUXs
  - Switch contains no input/output queue buffer, routing control and arbitration circuitry → very small area and power
- Switches are configured before run-time to connect any two processors; thus links are fixed and not shared
  → high throughput, low latency
- Small switches allow to have multiple parallel networks for increasing interconnection capacity. This platform contains two in parallel.

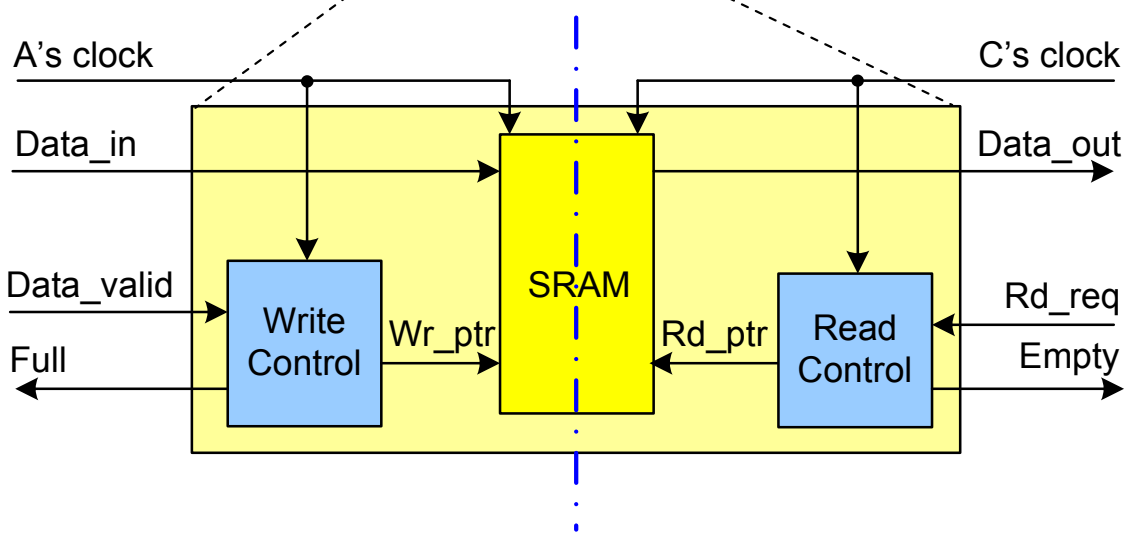
Source-synchronous communication (1)



- For each interconnection link, *clock* is sent with bundled *valid* and *data* signals from the source processor to the destination processor
- Links have a capacity of one data word per source-clock cycle
- No intermediate registering is needed, providing small area and low latency

Source-synchronous communication (2)

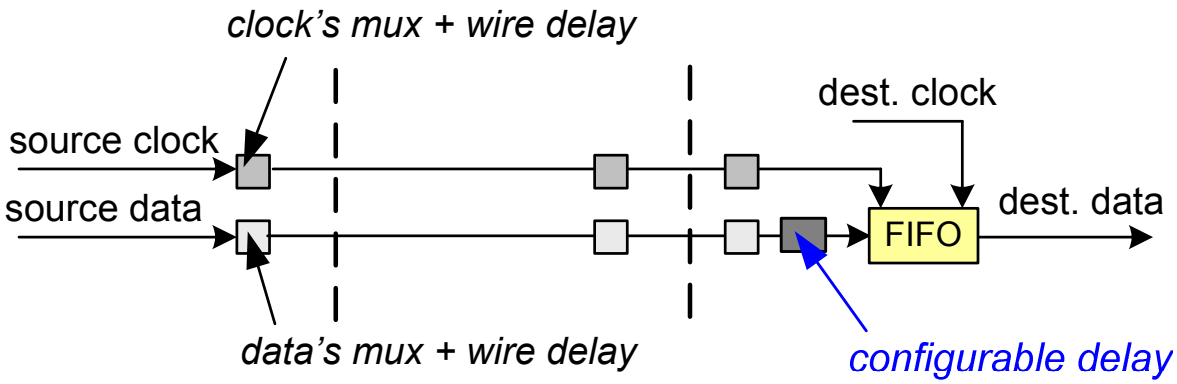




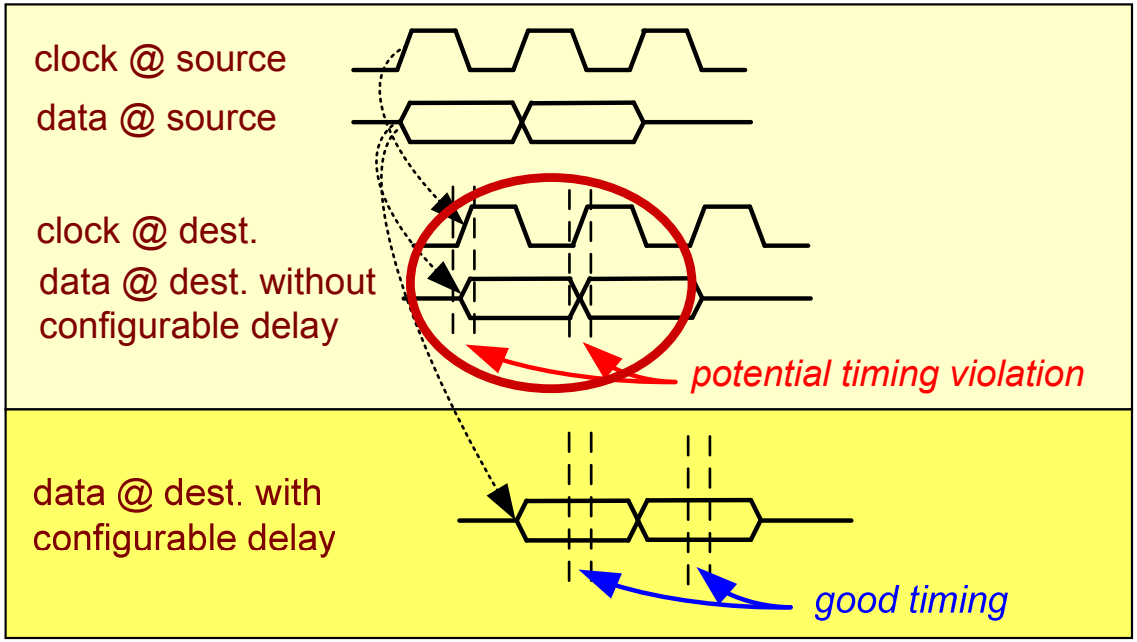
- Circular dual-clock FIFO uses SRAM array for dense data storage
- Write side controlled by source's clock;
  Read side controlled by destination's clock

[R. Apperson et al., TVLSI, 2007]

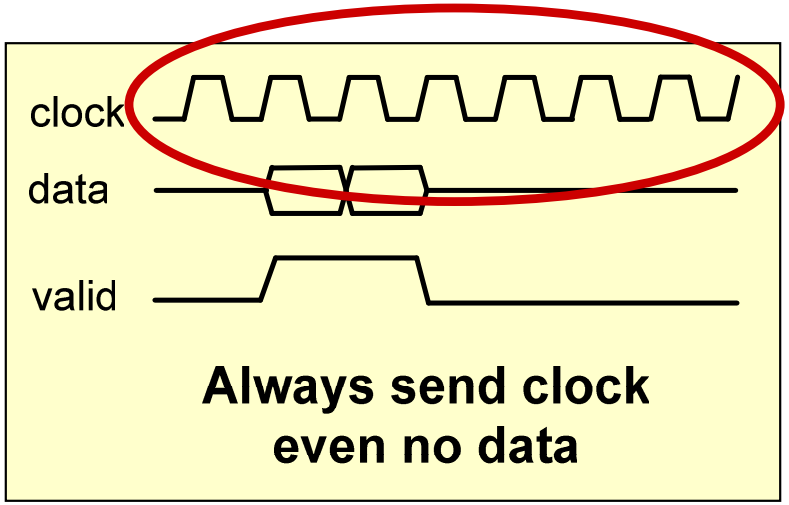
Communication Reliability

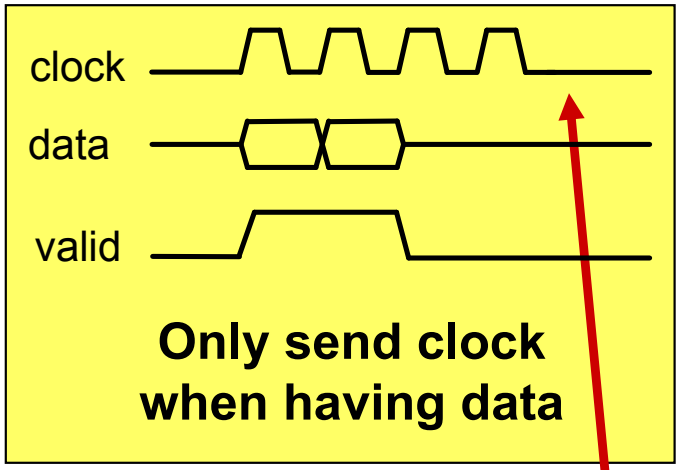


- Clock and data have equivalent delays → write clock can possibly trigger in the transition region of the data, causing a metastable failure
- A configurable delay is added to the data bus to keep the rising edge of the write clock in the stable data timing window

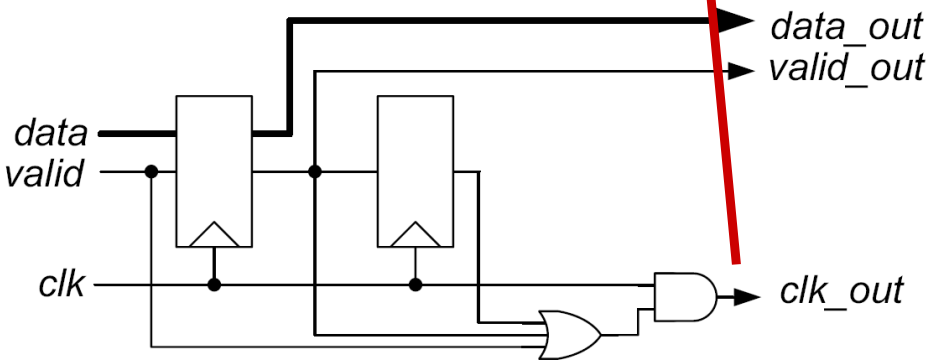


Low power communication strategy





- Always active clock dissipates unnecessary power
- Solution: send clock only when valid data is available
  - 45% power reduction
  - Requires at least one additional cycle due to the reconfigured delay

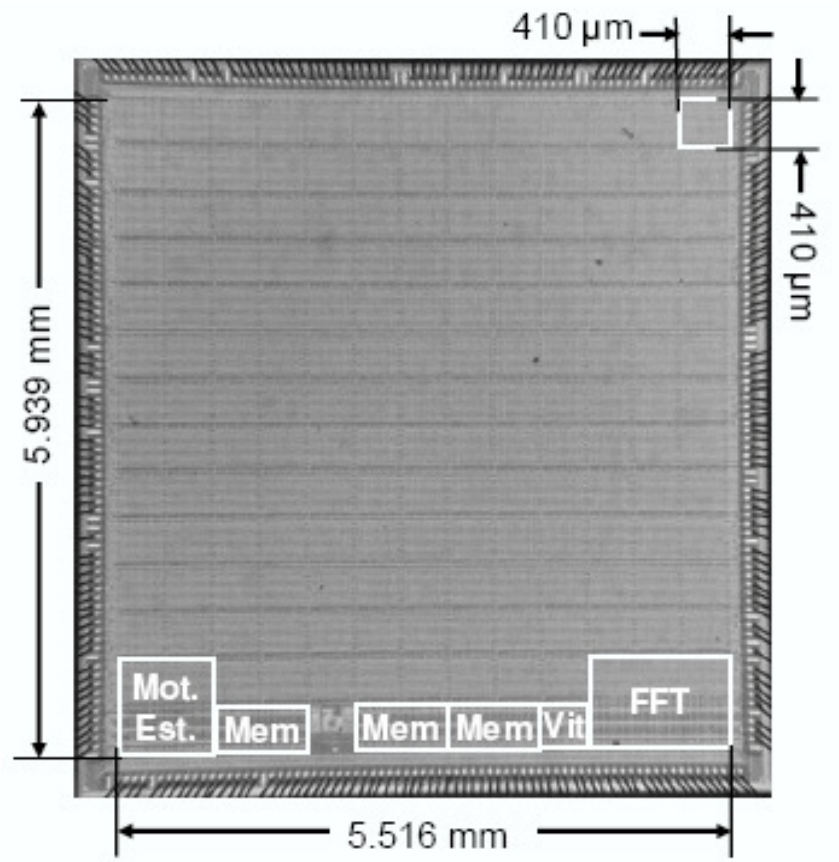


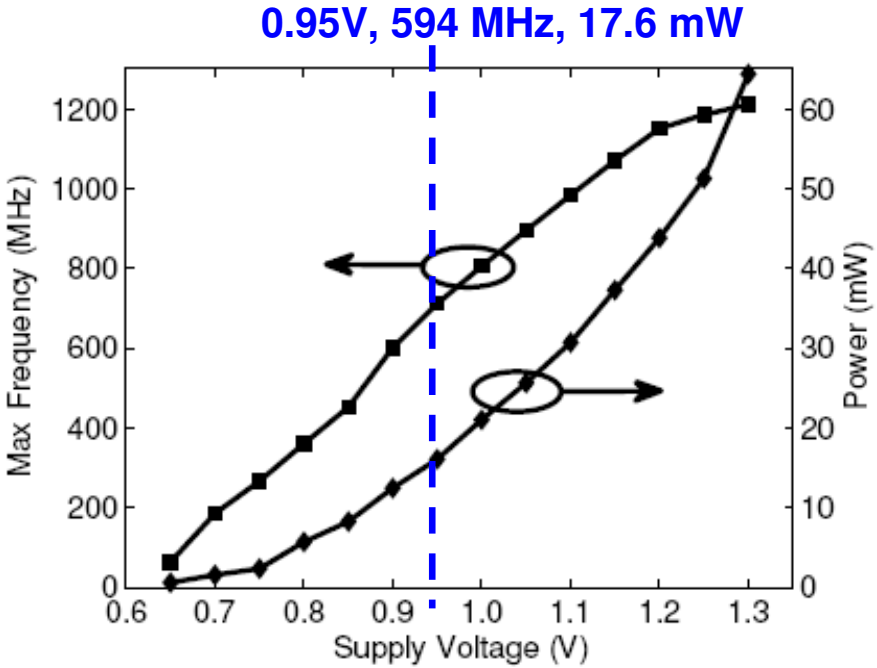
[Z. Yu and B. Baas, ICCD, 2006]

Outline

- Motivation
- Design of Our GALS many-core DSP platform
- The GALS compatible source-synchronous interconnect network
- Test chip implementation
- Mapping application case study: 802.11a/g baseband receiver
- Conclusion

Test chip implementation





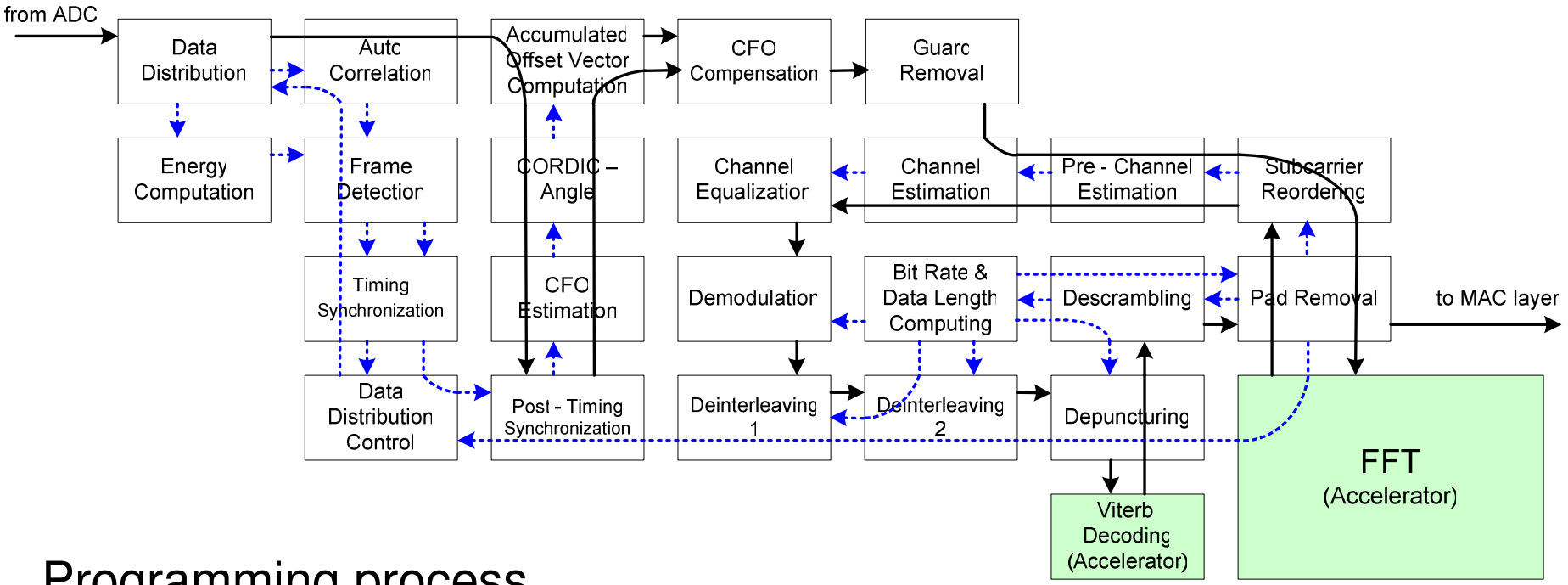
	100% Active	Stall (NOP)	Standby (Idle)
Prog. processor	17.6 mW	8.7 mW	0.03 mW
64-point FFT	12.7 mW	7.3 mW	0.33 mW
Viterbi	6.2 mW	4.1 mW	0.15 mW
FIFO write	1.9 mW	0.7 mW	~0 mW
Switch	1.1 mW	0.5 mW	~0 mW

- Fabricated in ST 65nm low-leakage CMOS
- Each processor occupies 0.17 mm^2 with only 7% area for comm. circuits
- Fully functional from 1.2 GHz at 1.3V down to 5 MHz at 0.6 V

Outline

- Motivation
- Design of Our GALS many-core DSP platform
- The GALS compatible source-synchronous interconnect network
- Test chip implementation
- Mapping application case study: 802.11a/g baseband receiver
- Conclusion

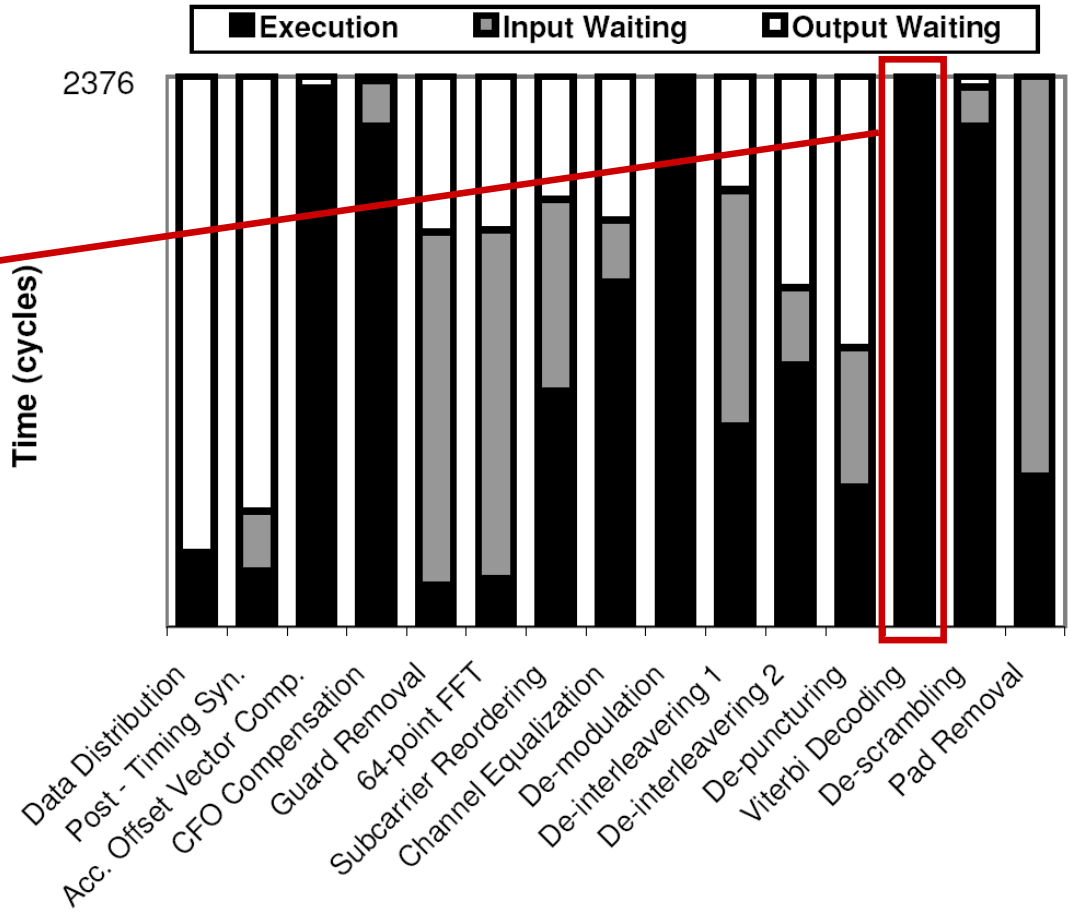
Mapping of a 802.11a/g baseband receiver



- Programming process
  - Manually partition tasks onto one/many processors
  - Program processors using a simple version of C language, combined with assembly language for interconnection configuration and code optimization
  - Simulate whole system at the cycle-accurate RTL level using NC Verilog
  - Compare results with a Matlab model to verify functionality
- Use activity percentages reported by the simulator for power estimation

Throughput evaluation

- OFDM data symbols are processed by an inter-connected sequence of processors
- The Viterbi processor is the slowest one and thus determines throughput of the receiver
- Faster processors stall on either input or output while waiting to receive or send data
- Each processor processes one 4 µs OFDM data symbol in 2376 cycles
  → 54 Mbps throughput at 594 MHz and 0.95 V



Power estimation at 594 MHz and 0.95V

- Power is estimated based on the number of cycles that each processor spends for execution, stalling with active clock, standby with halted clock, and the number of data items sent on each link and the distance of each link

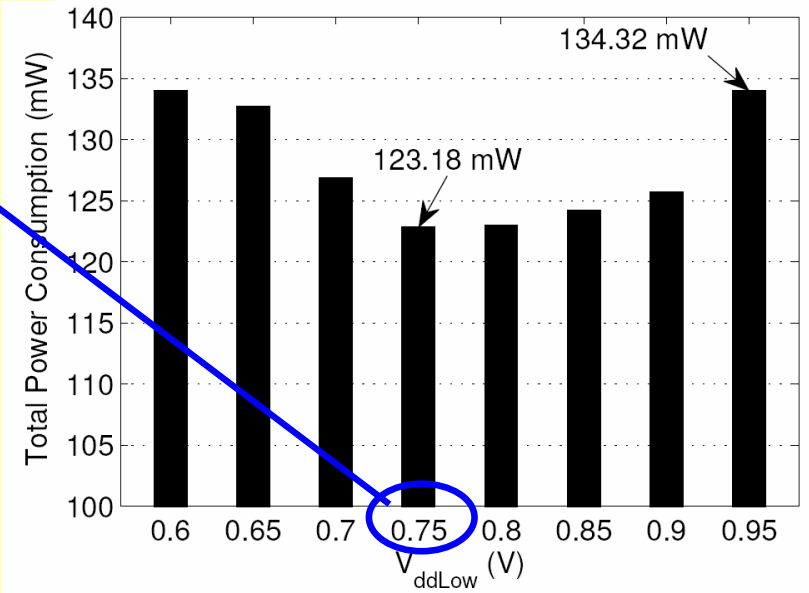
Processor	Execution Time (cycles)	Stall with Active Clock (cycles)	Standby with Halted Clock (cycles)	Output Time (cycles)	Comm. Distance (# switches)
Data Distribution	320	960	1096	80 x 2	6
Post-Timing Sync.	240	960	1176	80 x 2	5
Acc. Offset Vector Comp.	2320	56	0	80 x 2	2
CFO Compensation	2160	216	0	80 x 2	2
Guard Removal	176	768	1432	64 x 2	6
64-point FFT	205	768	1403	64 x2	3
Subcarrier Reorder	1018	576	782	48 x 2	4
Channel Equalization	1488	576	312	48 x 2	2
De-modulation	2352	24	0	288	2
De-interleaving 1	864	1512	0	288	2
De-interleaving 2	1130	1246	0	288	2
De-pucturing	576	1800	0	432	2
Viterbi Decoding	2376	0	0	216	3
De-scrambling	2160	216	0	216	2
Pad Removal.	648	1296	432	216	2

$$P_{Total} = \sum P_{Exe.i} + \sum P_{Stall.i} + \sum P_{Standby.i} + \sum P_{Comm.i}$$ = 174.76 mW

12.18 mW (or 7%)

Power reduction by freq. and volt. scaling

	Frequency scaling only		Frequency & Voltage scaling	
Processor	Optimal Frequency (MHz)	Power Consumed (mW)	Optimal Voltage (V)	Power Consumed (mW)
Data Distribution	80	3.52	0.75	0.63
Post-Timing Sync.	60	2.78	0.75	2.11
Acc. Off. Vector Comp.	580	17.72	0.95	17.72
CFO Compensation	540	16.53	0.95	16.53
Guard Removal	44	2.23	0.75	1.73
64-point FFT	51	1.64	0.75	1.23
Subcarrier Reorder	257	8.12	0.75	5.22
Channel Equalization	372	11.34	0.95	11.34
De-modulation	588	18.38	0.95	18.38
De-interleaving 1	216	7.36	0.75	4.95
De-interleaving 2	283	9.34	0.95	9.34
De-pucturing	144	5.70	0.75	4.10
Viterbi Decoding	594	7.13	0.95	7.13
De-scrambling	540	16.72	0.95	16.72
Pad Removal	162	5.52	0.75	3.71
Ten non-critical Procs.			0.95	*0.31*
Total (mW)		**134.32**		**123.18**



$$f_{Opt.i} = \frac{N_{Exe.i} \quad (cycles)}{4 \quad (\mu s)} \text{ (MHz)}$$

$$\sum P_{Comm.i} = 12.18 \text{ mW (or 10\%)}$$

Estimation and measurement

Configuration Mode	Estimated Power (mW)	Measured Power (mW)	Difference
At 594 MHz and 0.95 V	174.8	177.9	1.8%
At optimal frequencies only	134.3	139.6	3.9%
At both optimal freq. & volt.	123.2	129.8	5.1%

- The receiver operates correctly on the test chip
- Total time for designing, simulating, and testing this receiver is about 3 months
- The difference between estimated and measured power is within 2-5%

Outline

- Motivation
- Design of Our GALS many-core DSP platform
- The GALS compatible source-synchronous interconnect network
- Test chip implementation
- Mapping application case study: 802.11a/g baseband receiver
- Conclusion

Conclusion

- Many-core designs are a promising solution for programmable DSP platforms
- When coupled with GALS and heterogeneous architectures, it allows to achieve high performance at high energy efficiencies
- A test chip was fabricated in 65 nm CMOS and is fully functional
  - Uses static circuit-switched interconnection networks with simple switches that are highly suitable for many DSP applications
  - The networks utilize a simple yet effective source-synchronous communication technique across multiple clock domains
- An 802.11a/g Wi-Fi baseband receiver mapped onto this platform obtains 54 Mbps throughput while consuming only 130 mW, with 10% dissipated in its interconnection links

Acknowledgments

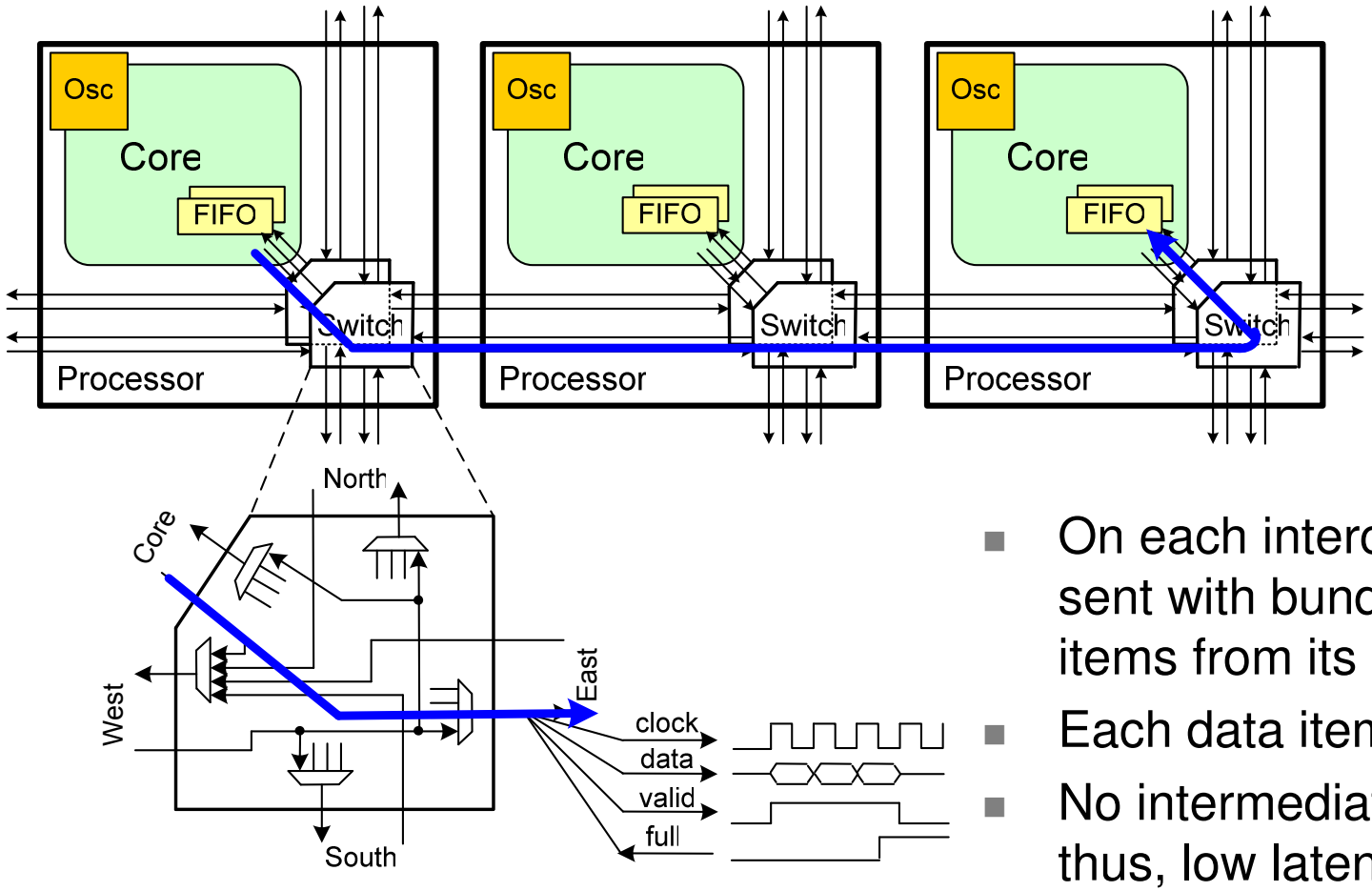
- NSF Grant 430090 and CAREER award 546907
- SRC GRC Grant 1598 and CSR Grant 1659
- Intellasys
- UC Micro
- Intel
- ST Microelectronics
- A VEF Fellowship
- SEM
- J.-P. Schoellkopf, P. Cogez, Y.-P. Cheng, A. Gatherer, R. Krishnamurthy, K. Bowman, and M. Anders

THANK YOU!

Backup/Extra Slides

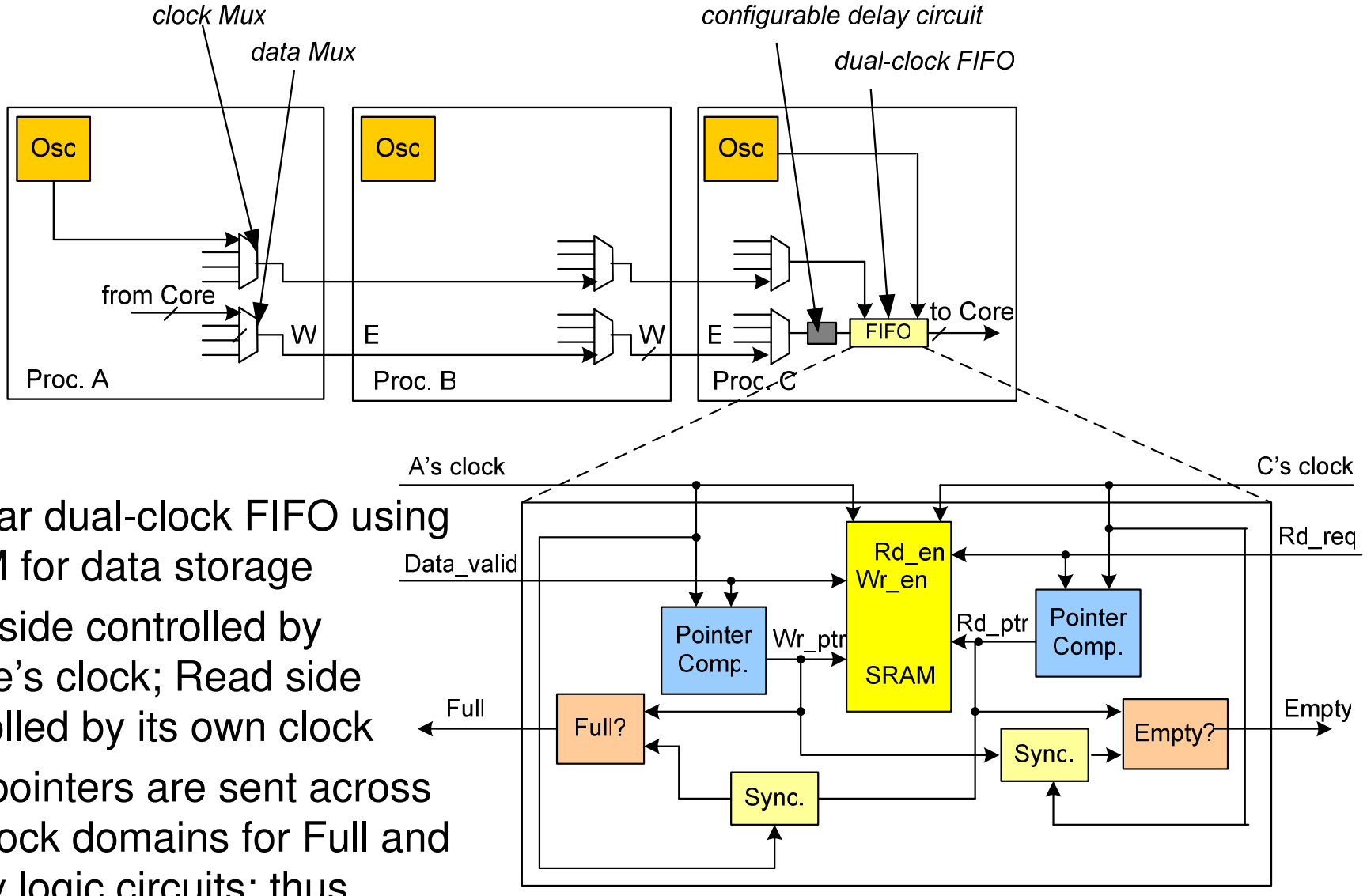
- Source-synchronous interconnects:
  - Switch structure
  - Dual-clock FIFO
- Programming so that the receiver operates obeying a FSM model:
  - Save power
  - Obtain high throughput
- Power estimation equations:
  - Based on activity percentages of execution, stall, standby, output times of each processor and its interconnection distance

Source-synchronous communication (1)



- On each interconnect link, clock is sent with bundled valid + data items from its source to destination
- Each data item is sent per cycle
- No intermediate register is needed; thus, low latency

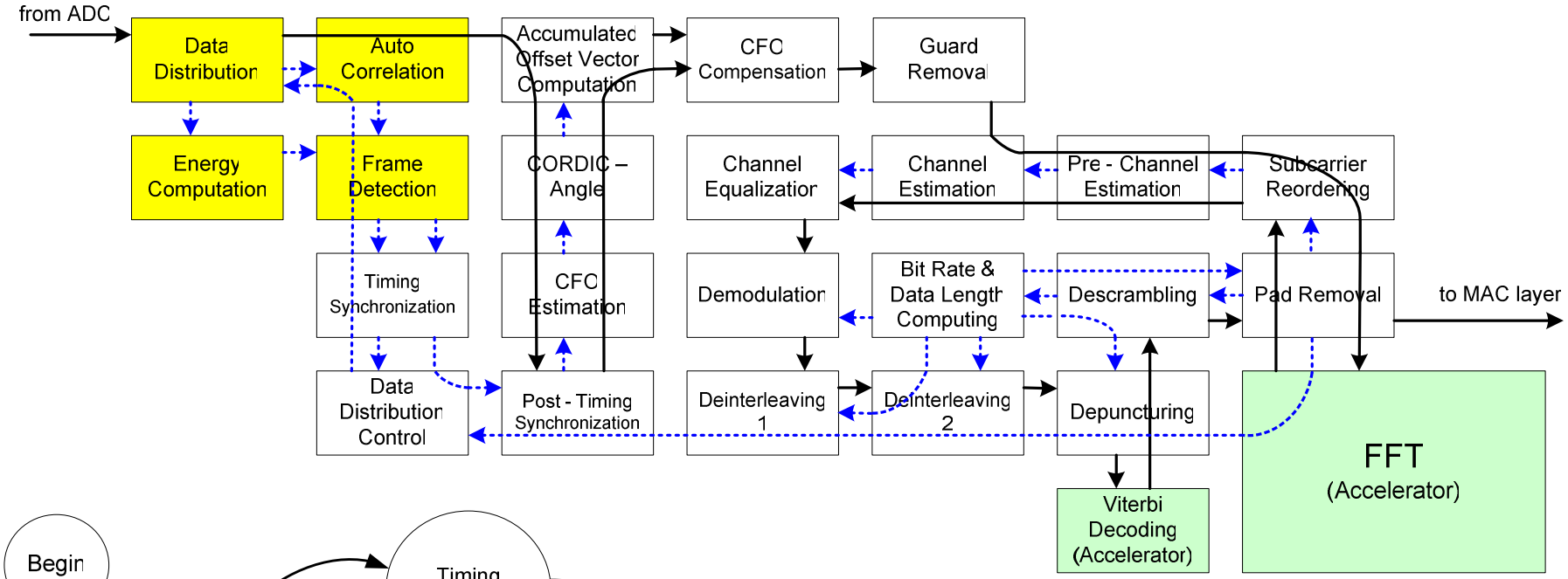
Source-synchronous communication (2)

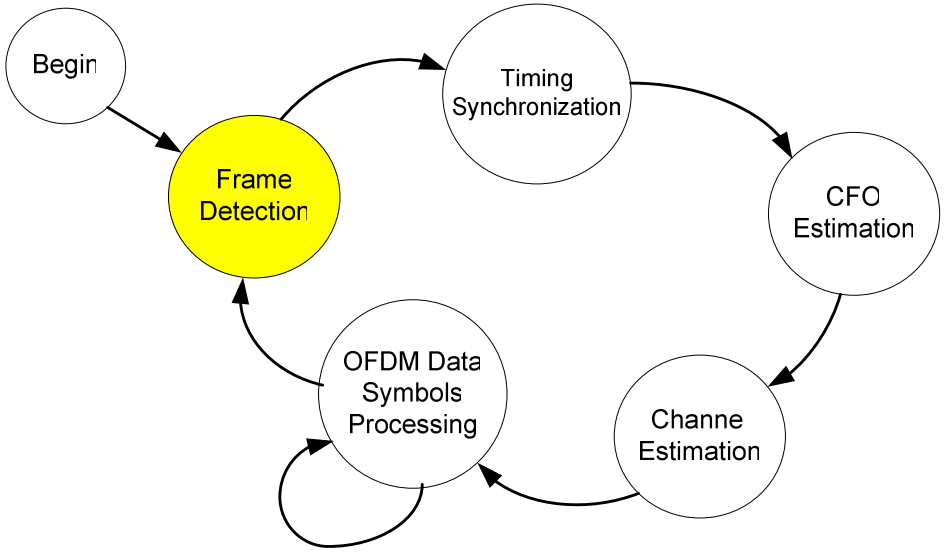


- Circular dual-clock FIFO using SRAM for data storage
- Write side controlled by source's clock; Read side controlled by its own clock
- Only pointers are sent across two clock domains for Full and Empty logic circuits; thus synchronizers are needed

[R. Apperson et al., TVLSI, 2007]

The Receiver Operates Obeying a FSM



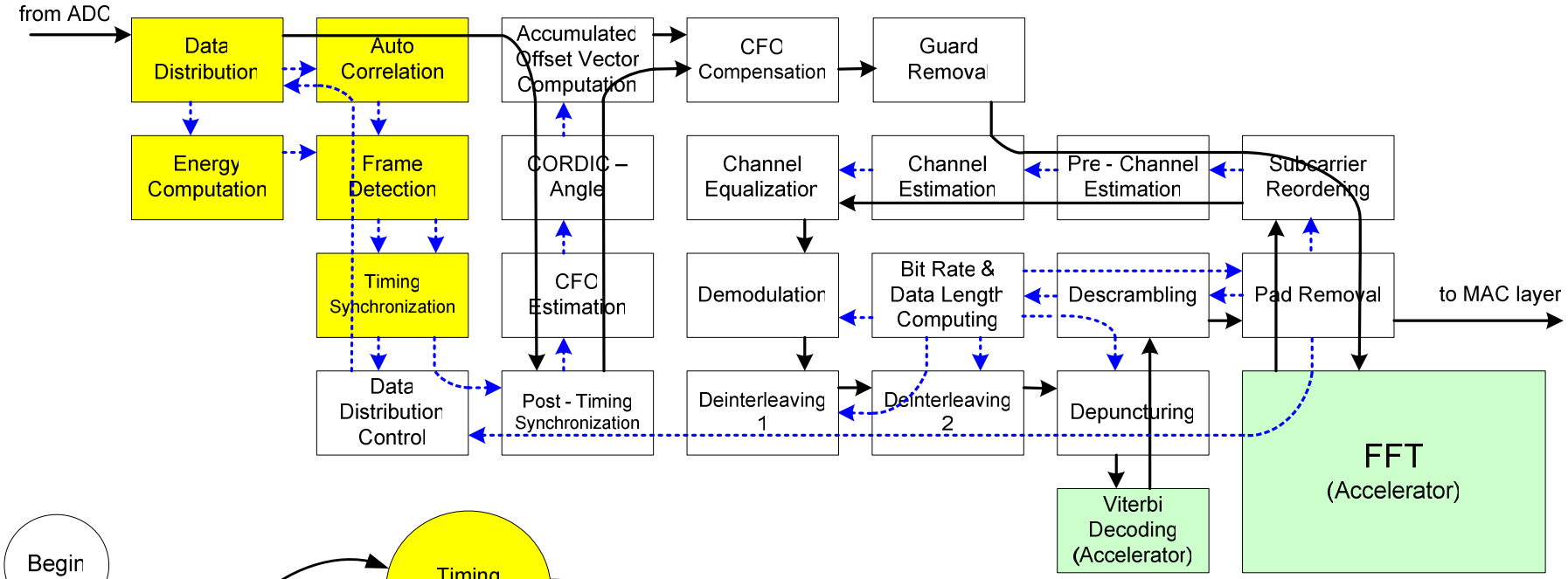


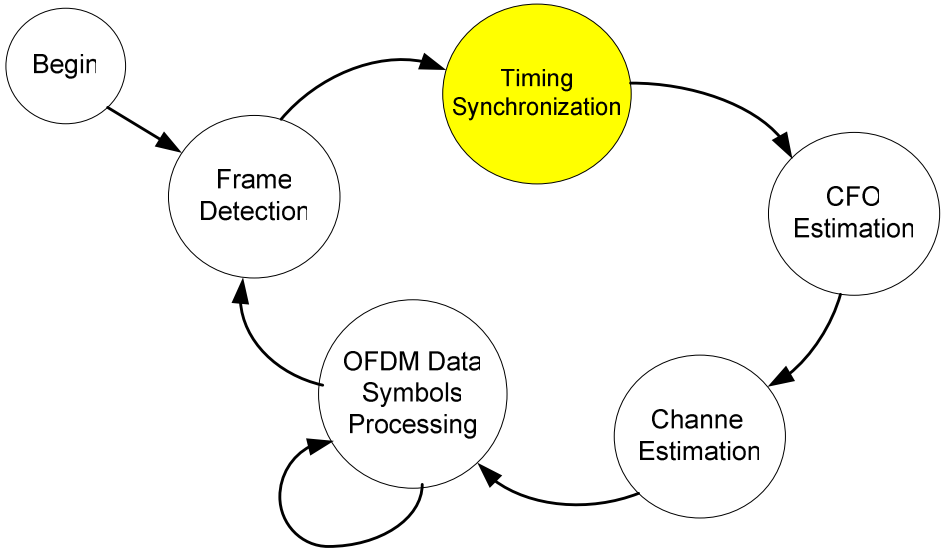
- Compute P(n) and Q(n)
- Frame is detected if

$$|P(n)|^2 > Th_{\text{det}} \cdot Q(n)^2$$

for 48 consecutive samples

The Receiver Operates Obeying a FSM

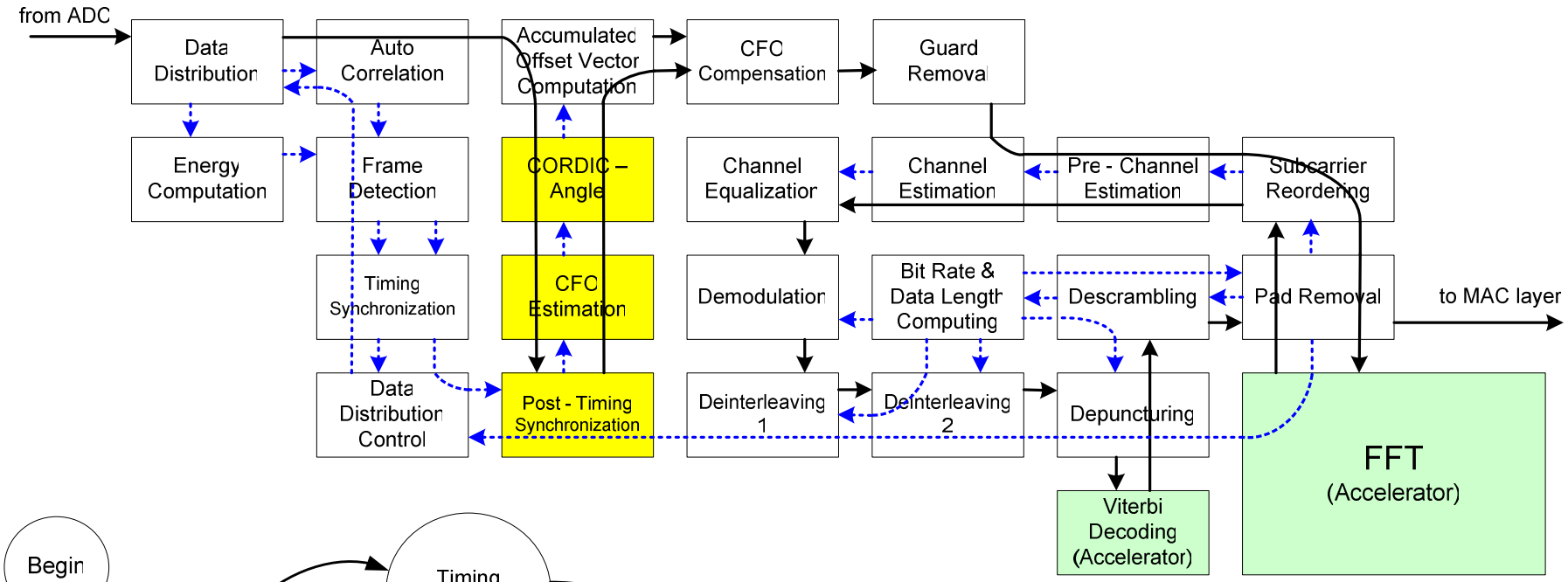


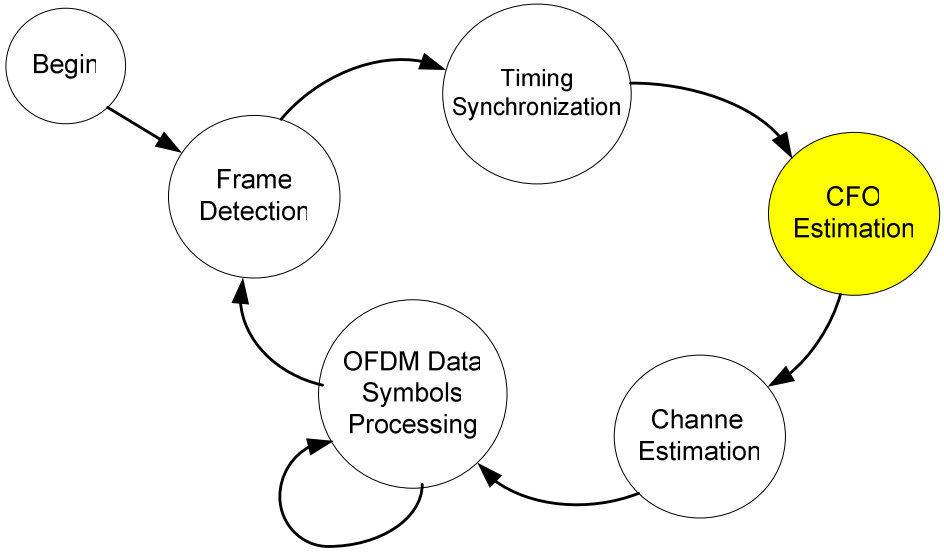


- Compute P(n) and Q(n)
- After frame is detected
- Timing is synchronized at first sample that satisfies:

$$|P(n)|^2 < Th_{syn} \cdot Q(n)^2$$

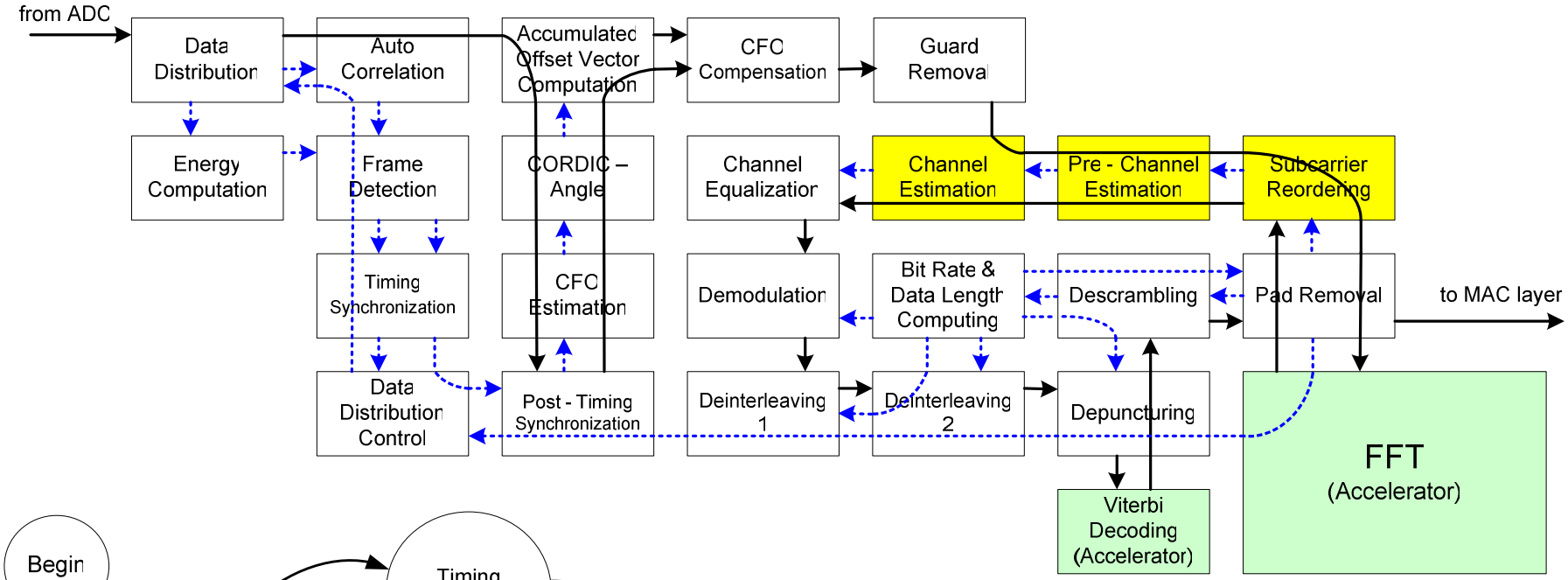
The Receiver Operates Obeying a FSM

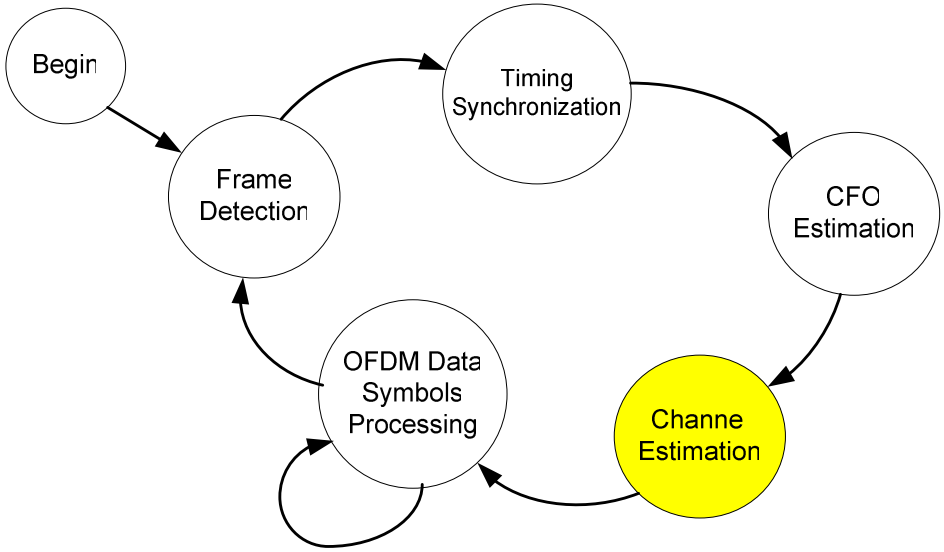




- Compute offset vector using two long-training symbols
- Compute offset angle α using CORDIC Angle algorithm

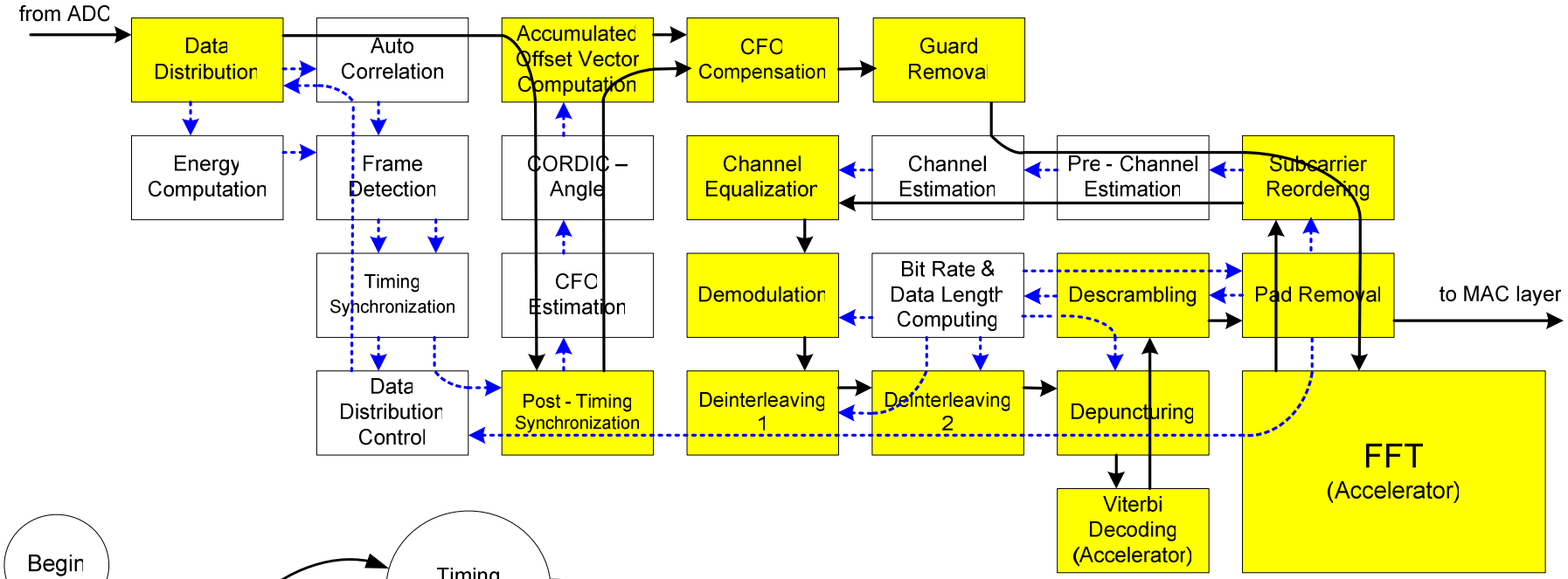
The Receiver Operates Obeying a FSM

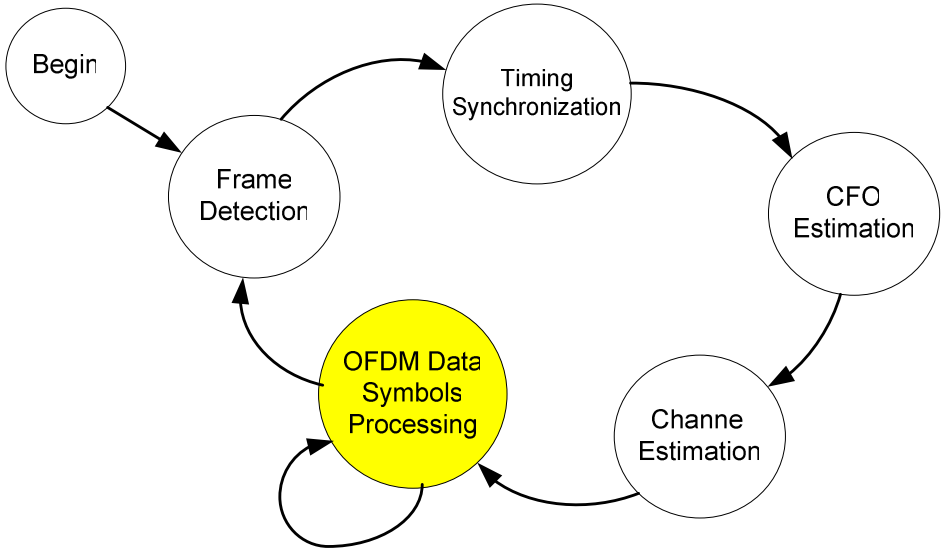




- Compute C(n) from two long-training symbols in the frequency domain (after FFT)

The Receiver Operates Obeying a FSM





- Includes all processors on the critical data path
- The OFDM SIGNAL symbol is used to decide the modulation scheme and code rate for all DATA symbols

Power estimation

$$P_{Exe.i} = \alpha_i \cdot P_{ExeAvg}$$

$$P_{Comm.i} = \gamma_i \cdot [(P_{SwitchActive} + 2P_{SwitchStall}) \cdot L_i + (P_{FIFOWriteActive} + 2P_{FIFOWriteStall})]$$

Processor	Execution Time (cycles)	Stall with Active Clock (cycles)	Standby with Halted Clock (cycles)	Output Time (cycles)	Comm. Distance (# switches)
Data Distribution	320	960	1096	80 x 2	6
Post-Timing Sync.	240	960	1176	80 x 2	5
Acc. Offset Vector Comp.	2320	56	0	80 x 2	2
CFO Compensation	2160	216	0	80 x 2	2
Guard Removal	176	768	1432	64 x 2	6
64-point FFT	205	768	1403	64 x2	3
Subcarrier Reorder	1018	576	782	48 x 2	4
Channel Equalization	1488	576	312	48 x 2	2
De-modulation	2352	24	0	288	2
De-interleaving 1	864	1512	0	288	2
De-interleaving 2	1130	1246	0	288	2
De-pucturing	576	1800	0	432	2
Viterbi Decoding	2376	0	0	216	3
De-scrambling	2160	216	0	216	2
Pad Removal.	648	1296	432	216	2

$$P_{Stall.i} = \beta_i \cdot P_{StallAvg} \qquad P_{Standby.i} = (1 - \alpha_i - \beta_i) \cdot P_{StandbyAvg}$$

$$P_{Total} = \sum P_{Exe.i} + \sum P_{Stall.i} + \sum P_{Standby.i} + \sum P_{Comm.i}$$