ESE5320: System-on-a-Chip Architecture

Day 5: September 19, 2022

Dataflow Process Model

Today

Dataflow Process Model

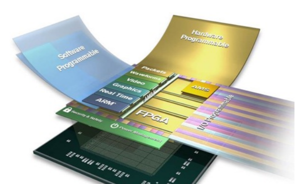
- Terms (part 1)
- Issues
- Abstraction
- Performance Prospects (part 2)
- Basic Approach
- As time permits (part 3)
  - Dataflow variants
  - Motivations/demands for variants

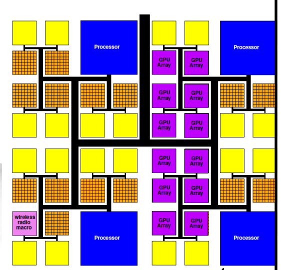
Message

- Parallelism can be natural
- Expression can be agnostic to substrate
  - Abstract out implementation details
  - Tolerate variable delays may arise in implementation
- Divide-and-conquer
  - Start with coarse-grain streaming dataflow
- Basis for performance optimization and parallelism exploitation

Programmable SoC

- Implementation Platform for innovation
  - This is what you target (avoid NRE)
  - Implementation vehicle





Reminder

- Goal: exploit parallelism on heterogeneous PSoC to achieve desired performance (energy)

Term: Process

- Abstraction of a processor
- Looks like each process is running on a separate processor
- Has own state, including
  - Program Counter (PC)
  - Memory
  - Input/output
- May not actually run on processor
  - Could be specialized hardware block
  - May share a processor

Thread

- Has a separate control location (PC)
- May share memory (contrast process)
  - Run in common address space with other threads
- May not actually run on processor
  - Could be specialized hardware block
  - May share a processor

7

Day 4

FIFO

- Hardware Block
- Outputs data in order received
  - First-In, First-Out
- Tell it when you are providing data
  - Write
  - May choose not to insert on a cycle
    - Need to signal
- Tell it when you are consuming data
  - Read
- Tells you when it's **empty** and has no data to provide
- Tells you when it's **full** and can hold nothing else

8

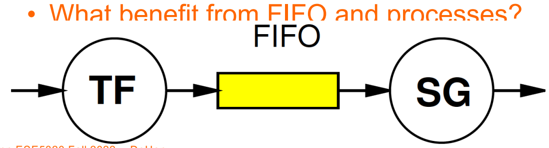
Process

- Processes (threads) allow *expression* of independent control
- Convenient for things that advance independently
- Process (thread) is the easiest way to express some behaviors
  - Easier than trying to describe as a single process
- Can be used for performance optimization to improve resource utilization

9

Preclass 2

- Average time for TF, SG independently?
  - 1 cycle 99% of time, 100 cycles 1% of time
- Throughput TF->SG with no FIFO?
  - Hint: what must wait on TF miss? SG miss?
- Throughput with FIFO?
  - How is FIFO changing?
- What benefit from FIFO and processes?



10

Preclass 2

- Independent probability of miss
  - P_f, P_g
- Concretely
  - 1 cycle in map
  - 100 run function and put in map
- If each runs independently (in isolation)
  - T~= 1*(1-P)+P*100
- If run together in lock step
  - Either can stall: $P=P_f+P_g-P_fP_g$
  - T~= 1*(1-P)+(P)*100

11

Multithread Web Page Load

- Typical browsers load images in separate threads
  - Allows parallelism in image loads
  - Doesn't block of text content (images that have already downloaded)
    - Get to see that even if image load slow
  - Separate thread keeps track of separate location in each image load

12

Model (from Day 4) Communicating Threads

- Computation is a collection of sequential/control-flow "threads"
- Threads may communicate
  - Through dataflow I/O
  - (Through shared variables)
- View as hybrid or generalization
- CSP – Communicating Sequential Processes → canonical model example

13

Issues

- **Communication** – how move data between processes?
  - What *latency* does this add?
  - *Throughput* achievable?
- **Synchronization** – how define how processes advance relative to each other?
- **Determinism** – for the same inputs, do we get the same outputs?

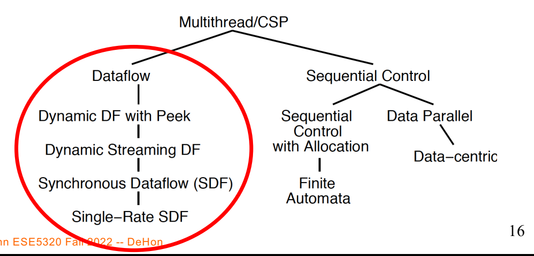
14

Today's Stand

- Communication – FIFO-like channels
- Synchronization – dataflow with FIFOs
- Determinism – how to achieve
  - …until you must give it up.
    - Only hint at giving up at end of lecture, time permitting

15

Dataflow Process Model



16

Operation/Operator

- **Operation** – logical computation to be performed
  - A *process* that communicates through dataflow inputs and outputs
- **Operator** – physical block that performs an Operation
  - E.g. processor, hardware block

17

Day 4

Dataflow / Control Flow

Dataflow
- Program is a graph of operations
- Operation consumes **tokens** and produces tokens
- All operations run concurrently
  - All processes

Control flow (e.g. C)
- Program is a sequence of operations
- Operation reads inputs and writes outputs into common store
- One operation runs at a time
  - defines successor

18

Day 4

Token

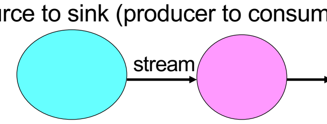
- Data value with presence indication
  - May be conceptual
    - Only exist in high-level model
    - Not kept around at runtime
  - Or may be physically represented
    - One bit represents presence/absence of data

Penn ESE5320 Fall 2022 -- DeHon 19

19

Stream

- Logical abstraction of a persistent point-to-point communication link between operations (processes)
  - Has a (single) source and sink
  - Carries data presence / flow control
  - Provides in-order (FIFO) delivery of data from source to sink (producer to consumer)



Penn ESE5320 Fall 2022 -- DeHon 20

20

Streams

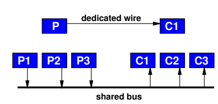
- Captures communications structure
  - Explicit producer→consumer link up
- Abstract communications
  - Physical resources or implementation
  - Delay from source to sink
  - Delay of Operators
- Contrast
  - C: producer->consumer implicit through memory
  - Verilog/VHDL: cycles visible in implementation
  - (can add **on top of** either C or Verilog)

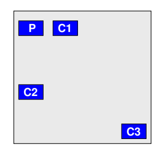
Penn ESE5320 Fall 2022 -- DeHon 21

21

Variable Delay Source to Sink

- How would placement of source and sink operator impact delay?





- How could sharing of interconnect between source and sink impact delay?

Penn ESE5320 Fall 2022 -- DeHon 22

22

Communication Latency

- Once map to multiple processors
- Need to move data between processors
- That costs time

Penn ESE5320 Fall 2022 -- DeHon 23

23

Day 3

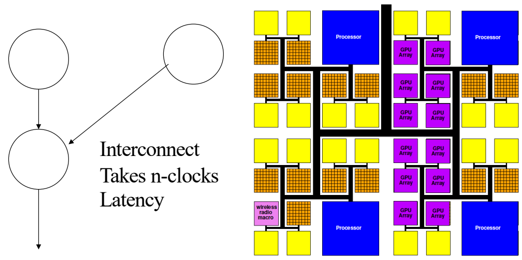
On-Chip Delay

- Delay is proportional to distance travelled
- Make a wire twice the length
  - Takes twice the latency to traverse
  - (can pipeline)
- Modern chips
  - Run at 100s of MHz to GHz
  - Take 10s of ns to cross the chip

Penn ESE5320 Fall 2022 -- DeHon 24

24

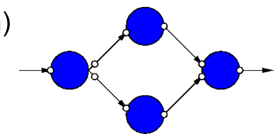
Dataflow gives Clock Independent Semantics



25

Dataflow Process Network

- Collection of Operations
- Connected by Streams
- Communicating with Data Tokens
- (CSP restricted to stream communication)



26

Dataflow Abstracts Timing

- Doesn't say
  - on which cycle calculation occurs
- Does say
  - What order operations occur in
  - How data interacts
    - i.e. which inputs get mixed together
- Permits
  - Scheduling on different # and types of resources
  - Operators with variable delay
  - Variable delay in interconnect

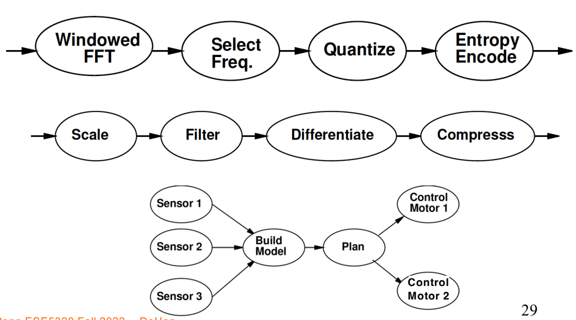
27

Dataflow Graphs Parallel Performance Prospect

Part 2

28

Some Task Graphs



29

Synchronous Dataflow (SDF) with fixed operators

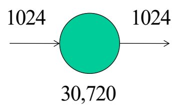
- Particular, restricted form of dataflow
- Each operation
  - Consumes a fixed number of input tokens
  - Produces a fixed number of output tokens
  - Operator performs fixed number of operations (in fixed time) – data independent
  - When full set of inputs are available
    - Can produce output
  - Can fire any (all) operations with inputs available at any point in time

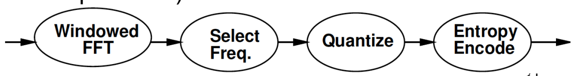
30

SDF Operator

FFT

- 1024 inputs
- 1024 outputs
- 10,240 multiplies
- 20,480 adds
- (or 30,720 primitive operations)





Processor Model

- Simple (for today's lecture)
  – Assume one primitive operation per cycle
- Could embelish
  – Different time per operation type
    - E.g. adds: 1 cycle, multiply: 3 cycles
  – Multiple memories with different timings

Time for Graph Iteration on Processors

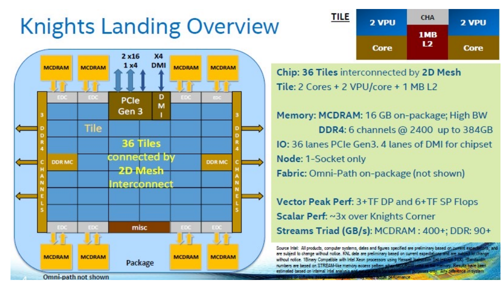
- Single processor $T_{one} = \sum_i Nops_i$
- One processor per Operation (process)
  - ❑ $T_{each} = \max(Nop_1, Nop_2, Nop_3, \ldots)$
- General

$$T_{map} = max\left(\sum_i c(1,i)\times Nops_i, \sum_i c(2,i)\times Nops_i, \sum_i c(3,i)\times Nops_i, \ldots\right)$$

c(x,y) – 1 if Processor x runs task y

(simplified resource bound model)

Intel Knights Landing



https://www.nextplatform.com/2016/06/20/intel-knights-landing-yields-big-bang-buck-jump/

[Intel, Micro 2016]

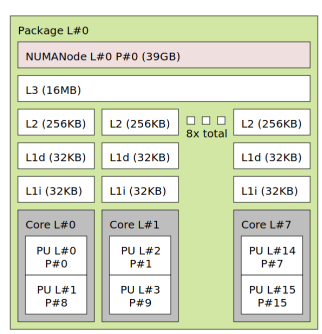
GRVI/Phallanx

- Puts 1680 RISC-V32b Integer cores
- On XCVU9P FPGA
- http://fpga.org/2017/01/12/grvi-phalanx-joins-the-kilocore-club/

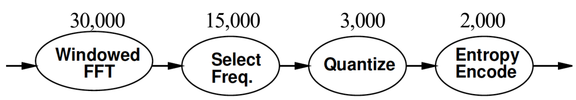
Fig 6: A 400 GRVI Phalanx. 10x5 clusters of 8 PEs (KU040)

[Gray, FCCM 2016]

Biglab



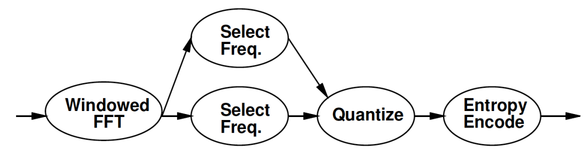
Map to different processors



- Map to (preclass 1)
  - One processor performance?
  - One process per processor performance?
  - Two processors
    - How?
    - Performance?
  - Bottleneck?

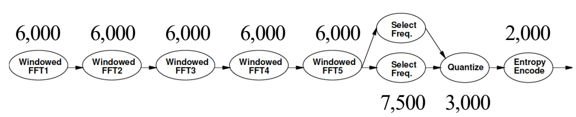
Refine Data Parallel

- If component is data parallel, can split out parallel tasks



Refine Pipeline

- If operation internally pipelineable, break out pipeline into separate tasks



Performance with one processor per operation?
Achieve same performance with how many processors?

Apple A15 Bionic

- 108mm^2, 5nm
- 15 Billion Tr.
- iPhone 13
- 6 ARM cores
  - 2 fast (3.2GHz)
  - 4 low energy (2GHz)
- 5 custom GPUs
- 16 Neural Engines
  - 11 Trillion ops/s?

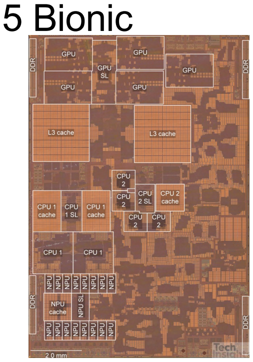
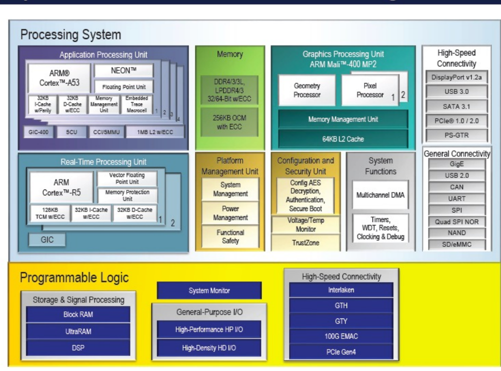
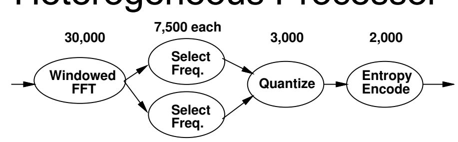


Image from https://semianalysis.com/apple-a15-die-shot-and-annotation-ip-block-area-analysis/
https://www.anandtech.com/show/16983/the-apple-a15-soc-performance-review-faster-more-efficient

Zynq® UltraScale+™ MPSoCs: EG Block Diagram

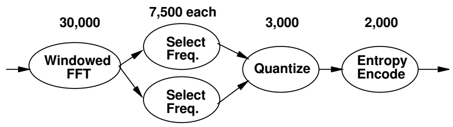


Heterogeneous Processor



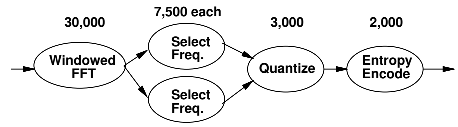
- GPU perform 10 primitive FFT Ops per cycle
- Fast CPU can perform 2 ops/cycle
- Slow CPU 1 op/cycle
- Map: FFT to GPU, Select to 2 Fast CPUs, quantize and Entropy each to own Slow CPU
- Cycles/graph iteration?

Heterogeneous Processor



	GPU	Fast CPU	Slow CPU
Windowed FFT	3,000	15,000	30,000
Select Freq. 1		3,750	7,500
Select Freq. 2		3,750	7,500
Quantize		1,500	3,000
Entropy Encode		1,000	2,000

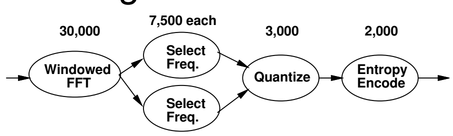
Heterogeneous Processor



	GPU	Fast CPU	Slow CPU
Windowed FFT	3,000	15,000	30,000
Select Freq. 1		3,750	7,500
Select Freq. 2		3,750	7,500
Quantize		1,500	3,000
Entropy Encode		1,000	2,000

Max(3000,3750,3000,2000) = 3750

Heterogeneous Processor

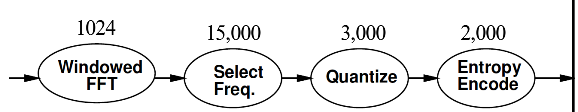


	GPU	Fast CPU	Slow CPU
Windowed FFT	3,000	15,000	30,000
Select Freq. 1		3,750	7,500
Select Freq. 2		3,750	7,500
Quantize		1,500	3,000
Entropy Encode		1,000	2,000

General case – find assignment with optimal timing

Custom Accelerator

- Dataflow Process doesn't need to be mapped to a processor
- Map FFT to custom datapath on FPGA logic
  - Read and produce one element per cycle
  - 1024 cycles to process 1024-point FFT

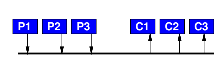


Operations

- Can be implemented on different operators with different characteristics
  - Small or large processor
  - Hardware unit
  - Different levels of internal
    - Data-level parallelism
    - Instruction-level parallelism
    - Pipeline parallelism
- May itself be described as
  - Dataflow process network, sequential, hardware register transfer language

Streams

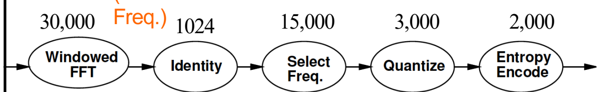
- Stream: logical communication link
- Some implementation options:
  - TCP/IP link over Internet
  - On-Chip bus
  - Buffer in memory



- Appropriate for
  - 2 processes on separate processors on same chip
  - 2 threads on same processor
  - One process at Penn, one at Amazon

Add Delay

- What does it do to computation if add an operation that copies inputs to outputs with some latency?
  - Impact on function?
  - What is throughput impact when Identity operation has
    - Latency 10, throughput 1 value per cycle?
    - (reminder 1024 values between FFT and Select Freq.)



49

Semantics (meaning)

- Need to implement semantics
  - *i.e.* get same result as if computed as indicated
- But can implement any way we want
  - That preserves the semantics
  - Exploit freedom of implementation

50

Basic Approach

51

Approach (1)

- Identify natural parallelism
- Convert to streaming flow
  - Initially leave operations in software
  - Focus on correctness
- Identify flow rates, computation per operator, parallelism needed
- Refine operations
  - Decompose further parallelism?
  - E.g. data parallel split, ILP implementations
  - model potential hardware

52

Approach (2)

- Refine coordination as necessary for implementation
- Map operations and streams to resources
  - Provision hardware
  - Scheduling: Map operations to operators
  - Memories, interconnect
- Profile and tune
- Refine

53

Dataflow Variants

Part 3:

(coverage here depends on time available)

54

Variable Delay

- Two different causes of “variable” delay
  1. Operator-dependent
  2. Data-dependent
- Operator-dependent
  – Depends on operator select
    - Fast processor, slow processor, GPU
    - Fixed time once select
- Data-Dependent
  – Depends on data being processed
    - Examples to come

Motivations and Demands for Dataflow Options

Time Permitting

Data-Dependent Variable Delay Operators

- Why might a multiplier have **data-dependent** variable delay?
  – Hint: consider shift-and-add multiply
    - Multiply by 3 vs. multiply by 16,777,215

GCD (Preclass 3)

- What is delay of GCD computation?

- while(a!=b)
  – t=max(a,b)-min(a,b)
  – a=min(a,b)
  – b=t
- return(a);

Data-Dependent Variable Delay Operators

- Operators with Data-Dependent Variable Delay
  – Cached memory or computation
  – Shift-and-add multiply
  – Iterative divide or square-root

Dynamic Rates?

- Dynamic rates – use of inputs or production of outputs is **data-dependent**
  – if (good_input(x)) out.write(x)
  – If (destination_high(x) high.write(x)
    else low.write(x)
- What is implication of static rates
  – on compression?
  – Filtering?
    - (e.g. discard all spam packets)

Data-Dependent Rates?

- Static Rates limiting
  - Compress/decompress
    - Lossless
    - Even Run-Length-Encoding
  - Filtering
    - Discard all packets from spamRus
  - Anything data dependent

Non-Blocking Stream Operations

- Blocking
  - only operations are read, write
  - If data not present, block for data to be available
- Non-blocking
  - Add operations to ask if data is available (if stream ready for write)

  if (not(empty(in1)) next_pkt=in1.read()

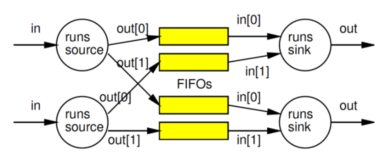
  else if (not(empty(in2)) next_pkt=in2.read()

When non-blocking necessary?

- What are cases where we need the ability to ask if a data item is present?
- Consider a server with multiple clients
  - Clients requests are independent, random
    - No guarantee make same number or rate of requests
  - What happens if must wait for a request from each of clients?
  - What would prefer to do?

When non-blocking necessary?

- Consider an IP packet router:



- Why need non-blocking here?

Non-Blocking

- Removed model restriction
  - Can ask if token present
- Gained expressive power
  - Can grab data as shows up
- Weaken our guarantees
  - Possible to get non-deterministic behavior
    - Depends on timing
      - Which we've said may vary with mapping
- Use when necessary, avoid if possible

Turing Complete

- Can implement any computation describable with a Turing Machine
  - (theoretical model of computing by Alan Turing)
- Turing Machine – captures our notion of what is computable
  - If it cannot be computed by a Turing Machine, we don't know how to compute it

Process Network Roundup

Model	Deterministic Result	Deterministic Timing	Turing Complete
SDF+fixed-delay operators	Y	Y	N
SDF+variable (data-dependent) delay operators	Y	N	N
Dynamic Rate DF blocking	Y	N	Y
Dynamic Rate DF non-blocking	N	N	Y
	Good For correctness	Good For Real-Time	Completeness (Compute anything)

Big Ideas

- Capture gross parallel structure with Process Network
- Use dataflow synchronization for determinism
  - Abstract out timing of implementations
  - Give freedom of implementation
- Exploit freedom to refine mapping to optimize performance
- Minimally use non-determinism as necessary

Admin

- Remember feedback
  - Today's lecture and HW2
- Reading for Day 6 on web
- HW3 due Friday
  - Implementing multiprocessor solutions on homogeneous (ARM) processor cores