Optimal Logic Depth Per Pipeline Stage

Presentation & Discussion

What is FO4?

- Fan-out of 4 is a process-independent delay metric in CMOS tech
- C_{load}/C_{in} = FO4
- C_{load} = total MOS gate capacitance driven by logic gate under consideration
- C_{in} = the MOS gate capacitance of the logic gate under consideration
- FO4 is used as a fairer comparison metric because scaled technologies are inherently faster in absolute terms
- In terms of a delay metric:
  - 1 FO4 is the delay of an *inverter*, driven by an inverter 4x smaller than itself, while driving an inverter 4x larger than itself

Outline

- Abstract
- History
- Quantifying Latch Overhead
- Finding the Ideal Clock Frequency
- New Microarchitectural Structures
- Related Work
- Wrap-up

Discoveries

- Using SPEC 2000 benchmarks, for high-performance architecture in 100nm
  - 8FO4 for integer benchmarks
    - 6 for useful work
    - 2 for overhead
  - 6FO4 for floating point benchmarks
  - Insensitive to latch and skew overheads
- Further pipelining:
  - Performance boost upto 2x
- Difficult to design instruction issue windows to operate in one cycle, considering high clock frequencies
  - Segmented instruction window

Processor Performance History

- Performance boosted by IPC and clock frequency gains
  - How much further can reducing the amount of logic per pipeline stage improve performance?

Intel Processor Trend

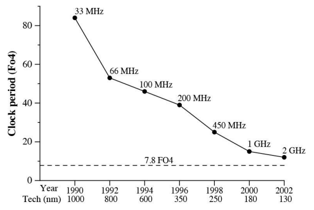


Figure 1: The year of introduction, clock frequency and fabrication technologies of the last seven generations of Intel processors. Logic levels are measured in fan-out-of-four delays (FO4). The broken line shows the optimal clock period for integer codes.

Caveats

- Reducing the amount of logic per pipeline stage
  - Seek balance between IPC and clock frequency

Overhead

$$\phi = \phi_{logic} + \phi_{latch} + \phi_{skew} + \phi_{jitter} \quad (1)$$

- Pipelined machines requires data and control signals at each stage to be saved at the end of every cycle
- Final estimate is 36 ps (1 FO4) at 100nm

Pulse Latch Circuit

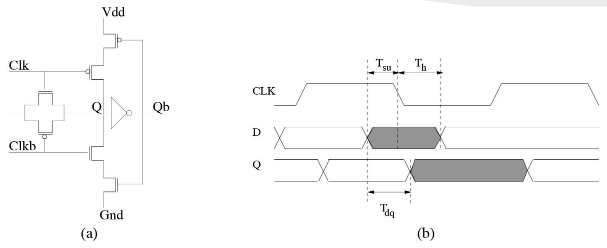


Figure 2: Circuit and timing diagrams of a basic pulse latch. The shaded area in Figure 2b indicates that the signal is valid.

Determining Overhead

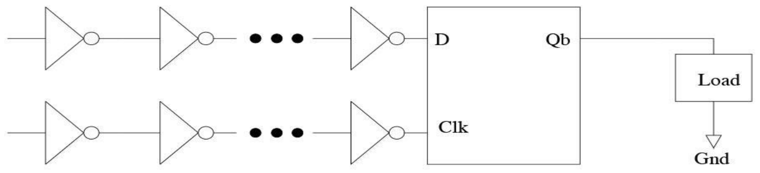
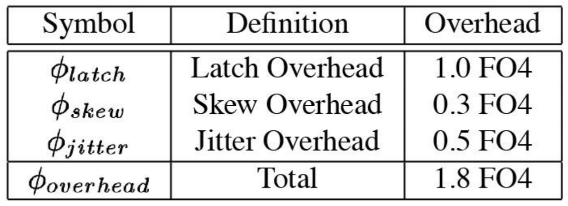


Figure 3: Simulation setup to find latch overhead. The clock and data signals are buffered by a series of six inverters and the output drives a similar latch with its transmission gate turned on.

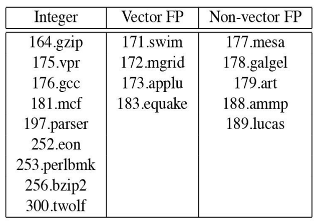
Skew & Jitter



Symbol	Definition	Overhead
ϕ_{latch}	Latch Overhead	1.0 FO4
ϕ_{skew}	Skew Overhead	0.3 FO4
ϕ_{jitter}	Jitter Overhead	0.5 FO4
$\phi_{overhead}$	Total	1.8 FO4

Table 1: Overheads due to latch, clock skew and jitter.

Simulation Framework

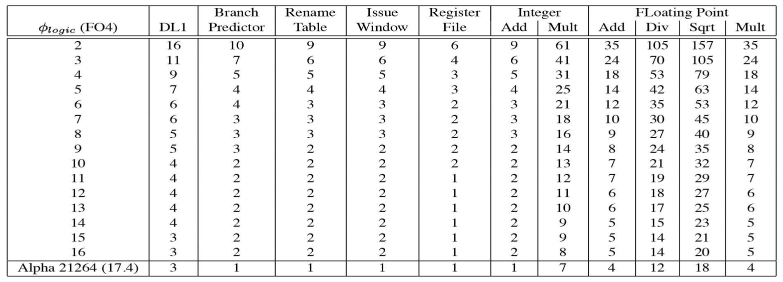


Integer	Vector FP	Non-vector FP
164.gzip	171.swim	177.mesa
175.vpr	172.mgrid	178.galgel
176.gcc	173.applu	179.art
181.mcf	183.equake	188.ammp
197.parser		189.lucas
252.eon		
253.perlbmk		
256.bzip2		
300.twolf		

Table 2: SPEC 2000 benchmarks used in all simulation experiments. The benchmarks are further classified into vector and non-vector benchmarks.

- Simulator that models both low-level structures of Alpha 21264 and execution core
  - Accuracy of 20% of a Compaq DS-10L
- Capacities of integer and floating point values were increased to 512 each
- Execution core permits addition of more stages to different parts of pipeline
  - Pipeline depth of different parts of the processor pipeline can be varied
- Experiments skip the first 500 million instructions of each benchmark and simulate the subsequent 500 million
- Cacti models on-chip microarchitectural structures and estimates their access times
  - Major MA structures were chosen to match the equivalent ones in the Alpha

Latencies & Scaling Pipelines



ϕ_{logic} (FO4)	DL1	Branch Predictor	Rename Table	Issue Window	Register File	Integer		FLoating Point			
						Add	Mult	Add	Div	Sqrt	Mult
2	16	10	9	9	6	9	61	35	105	157	35
3	11	7	6	6	4	6	41	24	70	105	24
4	9	5	5	5	3	5	31	18	53	79	18
5	7	4	4	4	3	4	25	14	42	63	14
6	6	4	3	3	2	3	21	12	35	53	12
7	6	3	3	3	2	3	18	10	30	45	10
8	5	3	3	3	2	3	16	9	27	40	9
9	5	3	2	2	2	2	14	8	24	35	8
10	4	2	2	2	2	2	13	7	21	32	7
11	4	2	2	2	1	2	12	7	19	29	7
12	4	2	2	2	1	2	11	6	18	27	6
13	4	2	2	2	1	2	10	6	17	25	6
14	4	2	2	2	1	2	9	5	15	23	5
15	3	2	2	2	1	2	9	5	14	21	5
16	3	2	2	2	1	2	8	5	14	20	5
Alpha 21264 (17.4)	3	1	1	1	1	1	7	4	12	18	4

Table 3: Access latencies (clock cycles) of microarchitectural structures and integer and floating-point operations at 100nm technology (drawn gate length). The functional units are fully pipelined and new instructions can be assigned to them every cycle. The last row shows the latency of on-chip structures on the Alpha 21264 processor (180nm).

Pipelined Architectures

- Vary pipeline depth of an in-order issue processor to determine optimal clock frequency
  - 7 stages and the issue stage can handle 4 instructions in each cycle
  - Functional units are fully pipelined, new instructions assigned at every clock cycle

Benchmarks of IO Pipeline

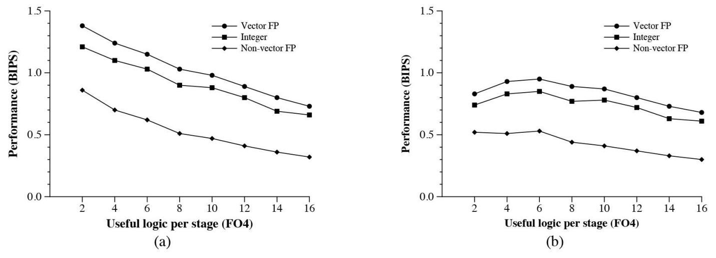


Figure 4: In-order pipeline performance with and without latch overhead. Figure 4a shows that when there is no latch overhead performance improves as pipeline depth is increased. When latch and clock overheads are considered, maximum performance is obtained with 6 FO4 useful logic per stage (ϕ_{logic}), as shown in Figure 4b.

CRAY Comparison

- The latency in one logic level for a CRAY-1S in CMOS would be 1.36 FO4
  - 8 gate levels for scalar
  - 4 gate levels for vector
- Optimal logic for vector benchmarks have stayed the same for the IO Pipeline and CRAY because of ample ILP
- Logic for integers has more than halved since the time of the CRAY
- Processor designed with modern techniques can be clocked at more than twice the frequency

Dynamically Scheduled Processor

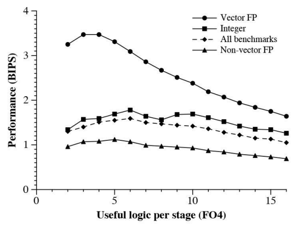


Figure 5: The harmonic mean of the performance of integer and floating point benchmarks, executing on an out-of-order pipeline, accounting for latch overhead, clock skew and jitter. For integer benchmarks best performance is obtained with 6 FO4 of useful logic per stage (ϕ_{logic}). For vector and non-vector floating-point benchmarks the optimal ϕ_{logic} is 4 FO4 and 5 FO4 respectively.

Relating Logic & Overhead

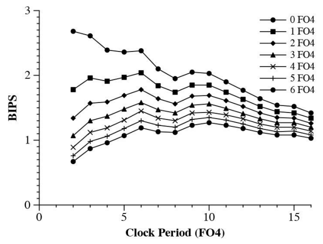


Figure 6: The harmonic mean of the performance of integer benchmarks, executing on an out-of-order pipeline for various values of $\phi_{overhead}$.

Logic & Structure Capacity

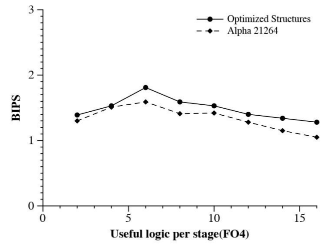


Figure 7: The harmonic mean of the performance of all SPEC 2000 benchmarks when optimal on-chip microarchitectural structure capacities are selected.

- Identify the best capacity and corresponding latency for on-chip structures:
  - Determine the sensitivity of IPC to the size and delay of each individual structure
  - Vary latency of each structure individually, with its capacity unchanged

IPC & Pipelining

- Increasing overall depth of a pipeline decreases IPC because of dependencies in critical loops
  - Issue, load, obtaining a value, and predicting a branch
  - Loops lengths increase
- Loops need to execute in fewest cycles

Critical Loops

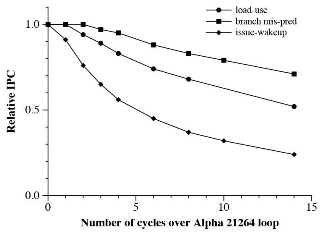


Figure 8: IPC sensitivity to critical loops in the data path. The x-axis of this graph shows the number of cycles the loop was extended over its length in the Alpha 21264 pipeline. The y-axis shows relative IPC.

Segmented Instruction Window

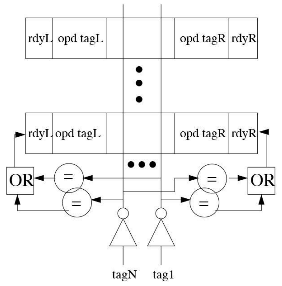


Figure 9: A high-level representation of the instruction window.

- For new instructions every cycle, the instruction issue window examines which instructions can be issued (woken up)
- Every cycle that a result is produced, a destination tag is broadcast to all entries in the instruction window

Instruction Window Stages

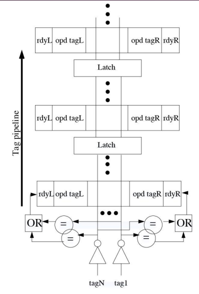


Figure 10: A segmented instruction window wherein the tags are broadcast to one stage of the instruction window at a time. We also assume that instructions can be selected from the entire window.

- 3 components constitute delay to wake up instructions
  - Delay to broadcast tags
  - Delay to perform tag comparisons
  - Delay to OR individual match lines
- To reduce tag broadcast latency, organize instruction window into stages

Effects of Stages

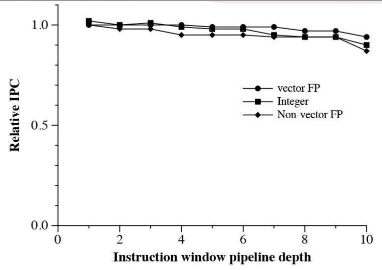


Figure 11: IPC sensitivity to instruction window pipeline depth, assuming all entries in the window can be considered for selection.

Selection Logic

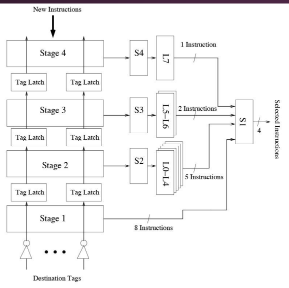


Figure 12: A 32-entry instruction window partitioned into four stages with a selection logic fan-in of 16 instructions

- Conventional processor, select logic examines the entire instruction window to select instructions for issue
- Decrease latency by reducing fan-in
- At every clock cycle, preselection logic blocks S2-S4 pick ready instructions in their stage
    - Instructions are stored in latches L1-L7 at the end of the cycle
    - In the second cycle, the select logic of S1 selects 4 instructions from Stage 1 and those in L1-L7 to be issued

Related Work

- Instructions can be woken up speculatively when grandparents are issued
    - If grandparents' tags are broadcast during the current cycle, parents' tags will probably be issued the same cycle
    - Cannot be issued until parents are issued
    - Reduces IPC, but enables functionality at higher frequencies
- Moving selection logic off the critical path
    - Two separate stages for wake-up and select
    - In the wake-up stage instructions are woken up by producer tags
    - All woken instructions speculate that they will be selected for issue in the following cycle - i.e. available
    - Result tags are broadcast as if all of them have been issued
    - Still, selection logic selects only a limited number from 'available', others are detected and rescheduled

Wrap-Up

- 6FO4 is the amount of useful logic per stage that will provide the best performance
  - Below this value, the improvement in clock frequency cannot compensate for a decrease in IPC and vice-versa
- Optimal clock frequencies are dependent on on-chip microarchitectural structures, these structures need to be pipelined to operate at high frequencies
- Segmented instruction window for pipelining with 4 stages without a marked decrease in IPC
- Pipelining can at most improve the clock rate by a factor of two, additional improvements must come from
  - Concurrency, ILP or TLP, or a combination
  - To follow historic performance trends, concurrency would have to increase at 33% per year, with 15 IPC before another 15 years has passed

END