

# The Impact of Pipelining on Energy per Operation in Field-Programmable Gate Arrays

Steve Wilton  
Department of Electrical and Computer Engineering  
University of British Columbia  
Vancouver, Canada

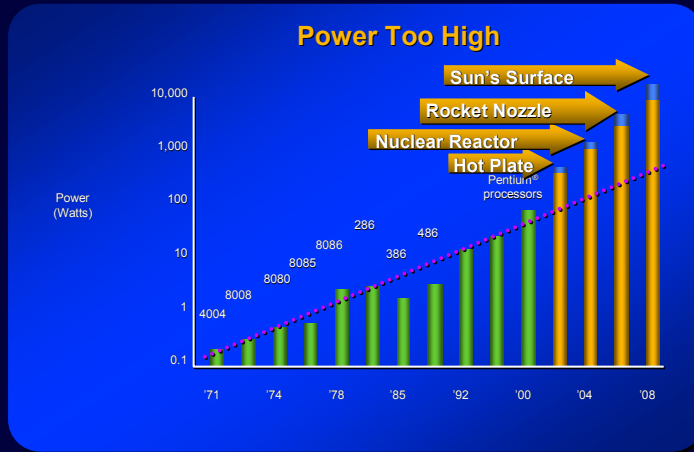
Su-Shin Ang, Wayne Luk  
Department of Computing  
Imperial College  
London, U.K.

## What this paper is about:

1. Quantify the power implications of pipelining for two real FPGA's:
  - A High-Density 0.13 $\mu$ m FPGA (Altera)
  - A Low-Cost 0.18 $\mu$ m FPGA (Xilinx)
2. What happens if we can get rid of all glitches?
3. What happens if we use power-aware physical design algorithms?

**Key Result: Pipelining can reduce power by 28-78%**

## Why Reduce Power?



Source: Intel

## Why Reduce Power? Hand-held applications



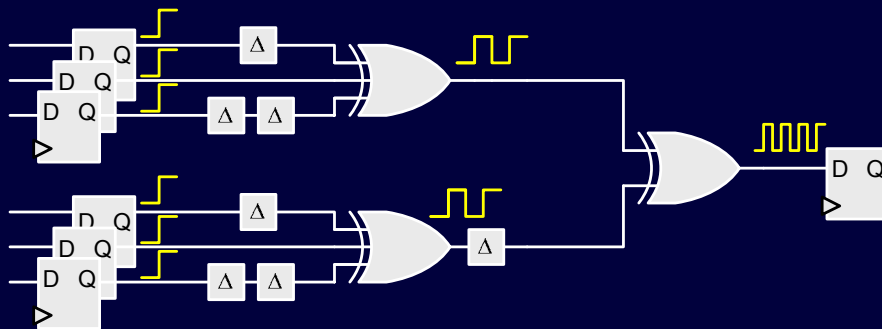
## What can we do about it?

1. Advanced Process Technologies
2. Power-Efficient Architectures
3. Power-Aware Physical Design Tools
4. Power-Aware System-level Design Tools

This paper: The interaction between **pipelining** and **energy consumption** in an FPGA

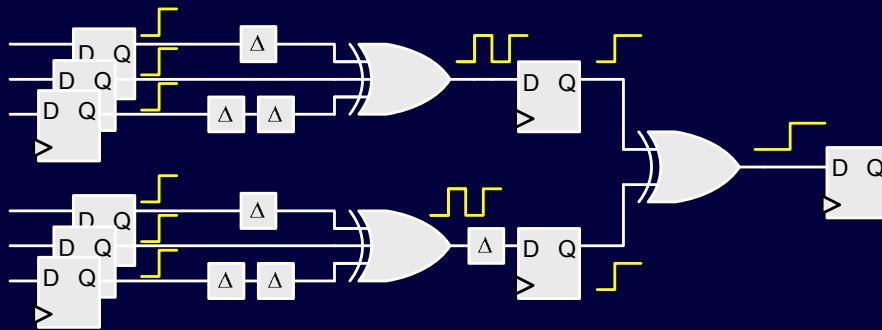
## Pipelining and Energy:

Intuitively, pipelining should reduce glitch power:



## Pipelining and Energy:

Intuitively, pipelining should reduce glitch power:



## Pipelining and Energy:

But, too much pipelining could hurt:

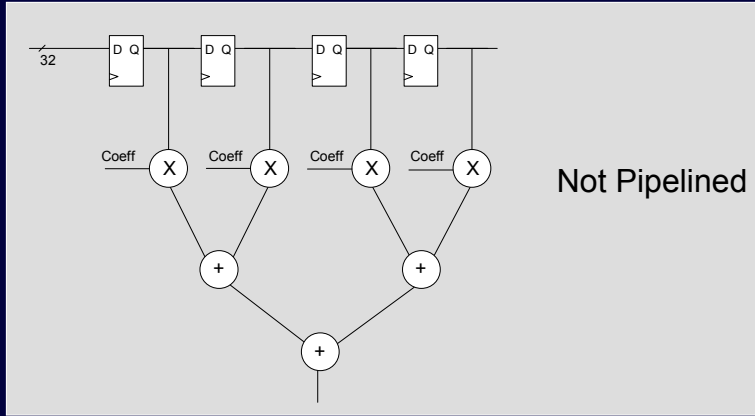
1. Extra flip-flops consume power as they switch
2. Extra burden on the clock tree

Why is this particularly interesting for an FPGA?

1. Wire delays can be long and switch slowly  
- leads to lots of glitches
2. Flip-flops are almost “free”, since they are there in the logic blocks anyway
3. Pipeline stages in the routing fabric

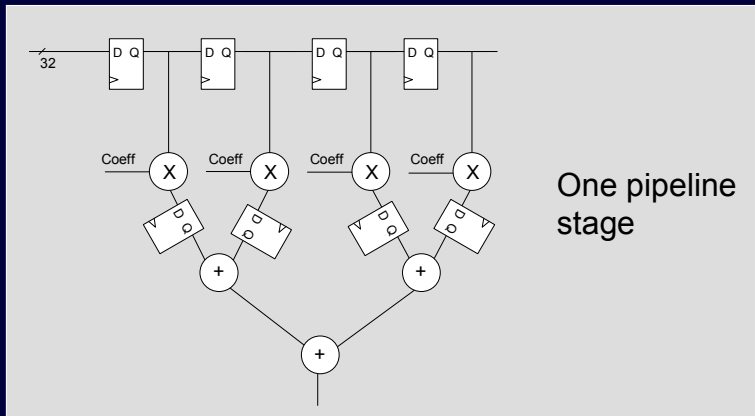
## Quantify the Power Implications of Pipelining:

Use four benchmark circuits. For each, vary the amount of pipelining. Example:



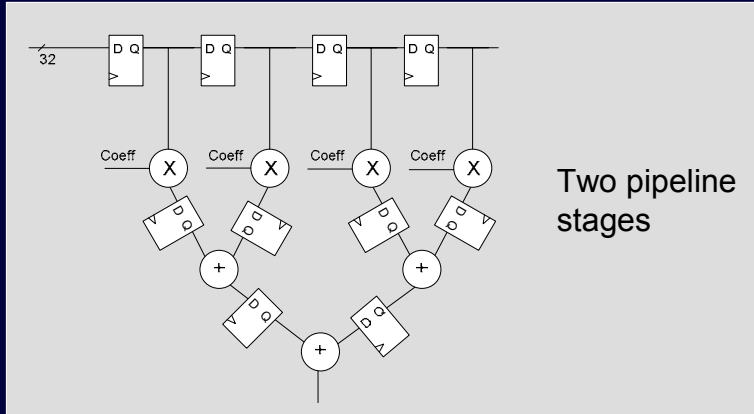
## Quantify the Power Implications of Pipelining:

Use four benchmark circuits. For each, vary the amount of pipelining. Example:



## Quantify the Power Implications of Pipelining:

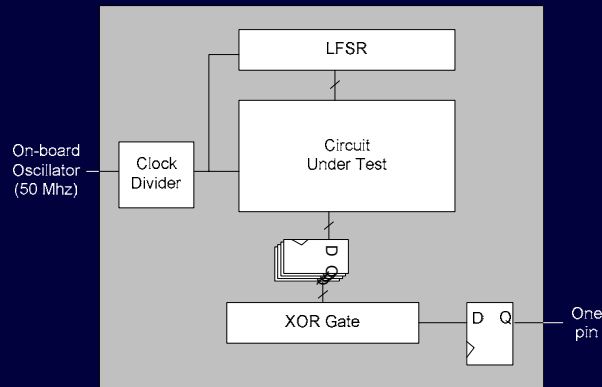
Use four benchmark circuits. For each, vary the amount of pipelining. Example:



	Number Pipeline Stages	Total LE's	Number of Registers	Max Stage Depth (LE's)
64-bit unsigned integer array multiplier	2	14 486	361	105
	4	15 289	555	73
	8	15 790	943	57
	16	16 223	1 719	49
	32	16 246	3 271	45
	64	15 526	6 374	43
Triple-DES encryption circuit	6	15 367	3 823	33
	12	17 826	4 542	17
	24	19 477	5 983	9
	48	20 579	9 023	5
8-Tap Floating Point FIR Filter	1	6 556	100	132
	2	6 408	353	100
	4	6 805	545	43
Cordic circuit to compute sine and cosine of angle	2	14 471	2 147	160
	4	16 135	3 811	80
	8	19 159	6 835	40
	16	25 495	13 171	20

## ***Test Set-up:***

---



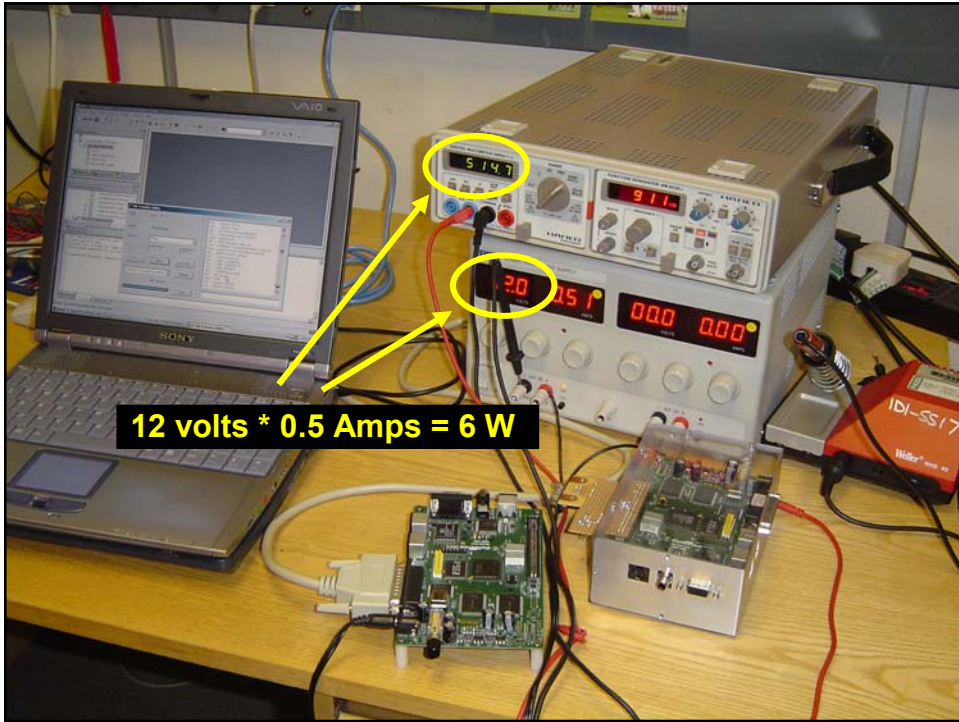
For the filter, the coefficients are kept constant.  
For the DES circuit, the key is kept constant.

## ***What exactly do we measure?***

---

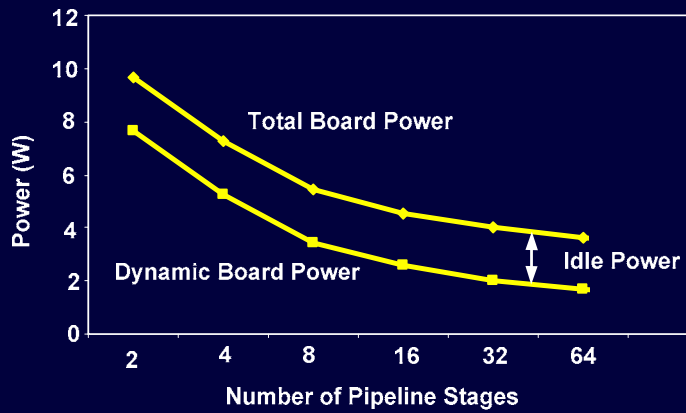
We really care about energy/operation

We can measure energy per operation by keeping the clock rate constant, and measuring power



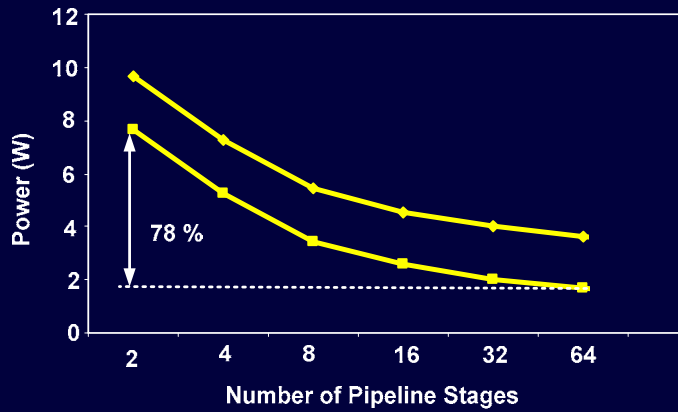
## Results for a 0.13 $\mu$ m device (Stratix)

64-Bit Multiplier:



## Results for a 0.13 $\mu$ m device (Stratix)

64-Bit Multiplier:



## Results for a 0.13 $\mu$ m device (Stratix)

Difference between most and least pipelined variants:

64-bit unsigned integer array multiplier	78%
Triple-DES encryption circuit	67%
8-Tap Floating Point FIR Filter	66%
Cordic circuit to compute sine and cosine of angle	40%

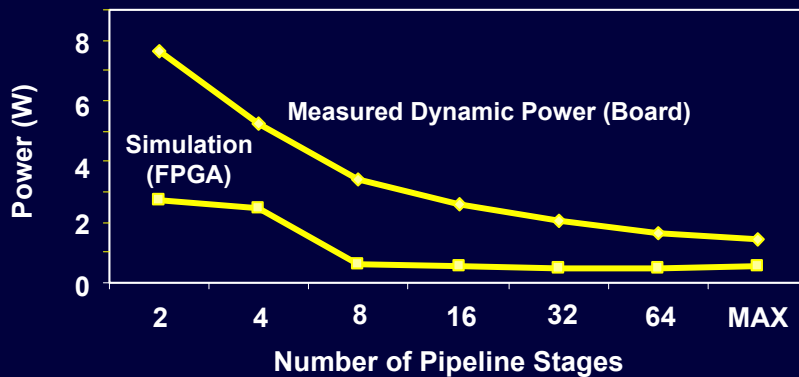
## Results for a 0.18 $\mu$ m device (Spartan)

Difference between most and least pipelined variants:

16-bit unsigned integer array multiplier	48%
4-Tap Floating Point FIR Filter	28%
Cordic circuit to compute sine and cosine of angle	66%

## Simulation vs. Measured Power (Stratix)

64-Bit Multiplier:

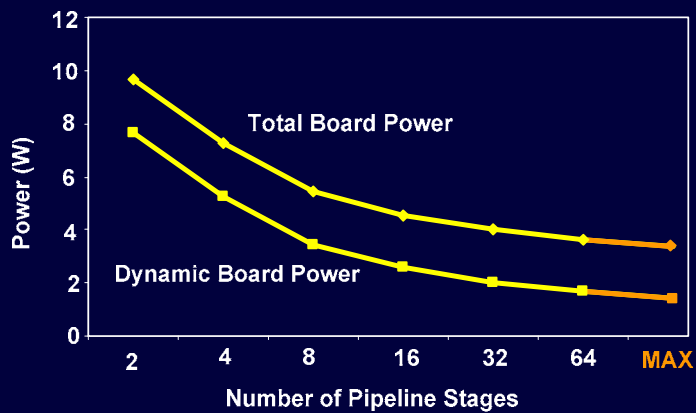


How much better could we possibly do?

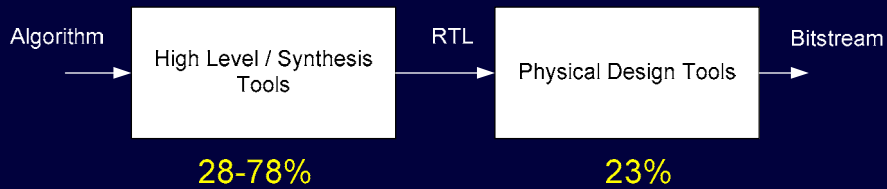
- Put a flip-flop after every logic element
- Functionality may be different, but it will give an estimate of the best we could do via pipelining

## How much better can we do? (Stratix)

64-Bit Multiplier:

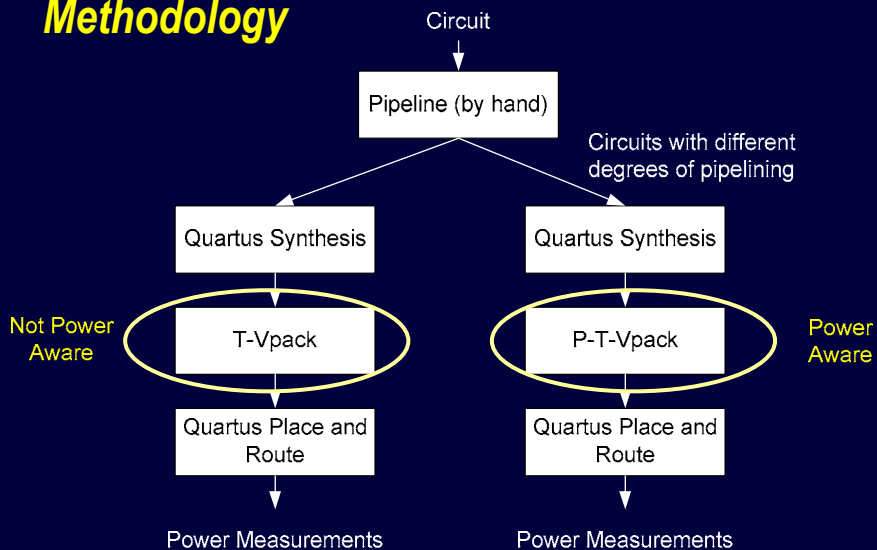


## High Level vs. Low Level Optimizations:



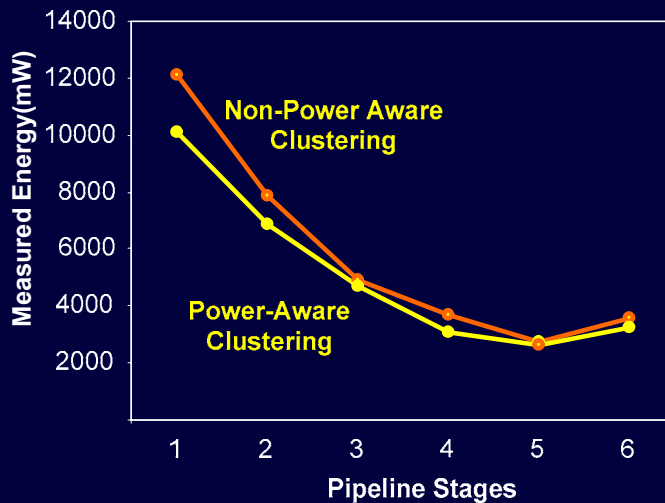
**Important Point:** by reducing number of high activity nets in the high level, perhaps lower level optimizations won't work as well

## **Methodology**



## 64-Bit Multiplier

---



## Summary:

---

We have quantified the power implications of pipelining for:

- A High-Density 0.13 $\mu$ m FPGA (Altera)
- A Low-Cost 0.18 $\mu$ m FPGA (Xilinx)

**Key Result: Pipelining can reduce power by 28-78%**

What happens if we can get rid of all glitches?

**Key Result: We are getting close, few gains**

What happens if we use power-aware physical design algorithms?

**Key Result: It still helps, but not as much**