PAPER I - PHOENIX

1) SUMMARY: The paper talks about Phoenix, a field programmable on-chip hardware proposed by the authors in an attempt to make the whole chip patchable with minimum performance cost. The aim is to provide **in-the-field repair through detecting the design flows in the processors and trying to recover from them**.

A study of design defects in 10 different processors has been presented with categorization of defects in order to understand the types (concurrent and complex) and concentration (core's periphery) of the defects. The **key contribution is the proposal and evaluation of phoenix which can help to detect design bugs and thus, allow early-to-market strategy**. Phoenix is the hardware in processor chip that **taps all the control signals which may trigger bugs and flags when the signal combinations matches the conditions which are known to trigger defects**. On detection, it flushes the pipeline and tries to refill the pipeline (disabling one of the signals, if possible), or emulates few instructions in an attempt to change the event interleaving. This helps to avoid the defect in re-run as the triggering of the defects generally requires some particular interleaving between the events. If detection is late (at the time or after the occurrence of detect), it passes the control to a customized recovery handler. Rollback mechanisms can be implemented to recover in post detection scenarios. The paper also presents the usefulness and evaluation of phoenix as well as an algorithm to determine the size of phoenix for the purpose of designing new processors with phoenix hardware. Phoenix provides good detection and recovery coverage (especially for concurrent defects which are 69% of total critical defects) while introducing negligible overhead.

2) Best Point: It not only presents a novel technique to detect and recover from the design defects in the whole chip which can be re-programmed multiple times via defect signature to handle new defects found, but also discusses the practical aspects of the technique. Many papers just present the concept and it's POC without talking about its practical usage. The authors discuss the usefulness of this technique in the actual world, present an evaluation of the idea based on coverage, performance/hardware overhead and provide an algorithm that would be useful for processor designers to design new processors with phoenix hardware. The **focus on the practical adaptation of the concept is a very good attribute of this paper.**

3) Worst Point: The worst feature of this scheme is its dependence **on signature based detection and recovery.** The processors in the field need to be updated with the patches and should have latest signature in the signature buffer. It will have the same problem as we have with software patches i.e., majority of the **users don't apply them** which leads to successful exploitation. As the users don't apply the patches, the software vulnerabilities exist in the software even though the fix is out there. Same problem will be faced with this technique of detection and recovery. Even if the processor manufacturer releases a new patch, users may not update their machines and thus, **the defects won't get patched.** This would inhibit the adoption of this technique. Because of the signature based detection, **it won't be useful to detect unknown bugs.** So, if we release the hardware early, it may have many defects and may get skipped by the verification engineers. As the signature won't be covering such defects, it would lead to hardware failure and other problems. Instead, redundancy based detectors such as DIVA would be

able to detect unknown defects. Tapping only those signals which may lead to trigger of the defect may worsen this scenario. What if we are the signals trapped at design time are not sufficient? If a defect involves a signal not trapped at design time, this technique won't be able to provide a patch for this.
This limits the dynamic nature of the detection and recovery mechanism proposed in the paper.

4) COMPARISON WITH PAPERS STUDIED IN THE CLASS: This paper is related to the DIVA paper discussed in the class as both propose techniques to detect and recover from the defects in the in-field processors. The authors mention DIVA in the related work section. The paper was published in 2006, around 7 years after the publication of the DIVA paper. Both the techniques are to detect and recover from defects once the processor is in the field and thus, allows faster time-to-market while minimizing the financial loss because of the new defects found. But, the ways to achieve the goal are different. DIVA adds a checker at the end of the pipeline while Phoenix traps the signals to perform signature based defect detection.

DIVA focuses on inner-core (fetch and Exec units) defect detection while the defects in the core's periphery (specially, TLB plus virtual memory and multiprocessor structures) dominate. So, DIVA will not work well if we want to provide detection for maximum possible design bugs. Therefore, DIVA may not be able to detect many of the defects that phoenix can detect. Phoenix seems to provide better coverage (for the type of errors considered in the paper) as compared to DIVA. In addition, DIVA won't be able to detect defects such as cache corruption.

DIVA may be useful in detecting unknown bugs but has power overhead as it is always active and re-executes each instruction. Phoenix needs re-execution only when it flags based on given signature. So, though the two papers are related, the approaches are much different and applicable to different types of bugs with different ways of detection and recovery. The basic goal of the paper might be inspired by DIVA but techniques used are different (some of which were proposed in other papers).
Techniques such as SRT, CRT studied in the class also provide detection and recovery methods but those are for transient defects and are not suitable for design bugs.

5) IMPROVEMENT: Given the advancement in the processor design technology in the last 8 years, I would like to explore the problem of detecting complex defects and how we can detect as many defects as possible without compromising the performance or design overhead. We can try to study the common types of complex defects reported in different processors (as done in the paper for concurrent defects). Even if we can't detect all the defects, we can design hardware in an attempt to detect most common defects balancing the performance and cost as well. This is important because though the number of complex defects is lesser than concurrent, they may be very critical and may cause a great loss. Having so much advancement and high performance processors, we can try to maximize overall defect detection. In 2014, we can try to explore detection possibility of complex defects by capturing the sequences and identifying trigger conditions using advanced ML and pattern recognition techniques. But, feasibility of such algorithms at hardware level would have to be evaluated. Otherwise, some other pattern matching and state transition techniques can be used.

PAPER II - RAZOR

1) SUMMARY: "Dynamic voltage scaling" refers to dynamically controlling the supply voltage in an attempt to minimize the energy consumption by lowering the voltage during low CPU utilization periods. **Razor, a better-than-worst-case design, is a DVS approach that tries to regulate the voltage supply by monitoring the error rate of circuit timing errors.** Razor permits the microprocessor to tolerate circuit timing errors so that it can operate with a lower voltage supply. It detects and corrects such errors dynamically. **The key contribution is the proposal of a technique that eliminates the necessity of voltage margins and thus, it allows performing circuit operations even at sub-critical voltage levels while maintaining controlled low error rate resulting in lesser energy consumption**.
The authors have introduced a razor flip flop, i.e., each delay critical flip flop is extended with a shadow latch. The main flip flop is controlled using a fast clock as usual and the shadow latch using a delayed clock. So, the pipeline stage value is latched twice and delay error in main flip flop (if any) can be detected by comparing the two samples. In case of an error detection, the value in the shadow latch (which is expected to be correct) is stored into the main flip flop making it available to the next stage in the pipeline. The pipeline recovery can be done either using clock gating (stalling the pipeline) or counterflow pipelining. Metastability in main flip flop is handled by introducing a metastability detector and if detected, it is corrected using the same razor error detection component. In case of a meta stable error signal, a panic signal is generated resulting in flushing of the pipeline and restart with raising the level of supply voltage to avoid instruction failure. They also discuss about supply voltage control system that adjusts the supply voltage based on the rate of the errors monitored to maintain a constant small error rate. By implementing it on a multiplier and simulating for an adder, they show that this approach results in significant energy reduction with small overhead.

2) BEST POINT: The error detection technique lets us know error rate in the circuit which helps us in eliminating all forms of voltage margins and thus, allowing more reduction in the energy consumption as compared to the other approaches. The **concept of eliminating the voltage margin altogether and performing operations at even sub-critical voltage level (or at the failure point) is a novel and the best point as it showed a new direction to the researchers**. The proposals before this paper used to consider some kind of voltage margins and thus, couldn't achieve that much reduction in energy consumption. This changed the ideology from worst case microprocessor design to better-than-worst-case design. So, it allows the processor to operate at much lower voltage supply and facilitates the recovery in case of failure.

3) WORST POINT: This **approach doesn't sound much practical as it requires significant new hardware and a different microprocessor design methodology.** Given the current state of the microprocessors, it expects to discard the existing methodologies and design the microprocessors in a different way. This is because to achieve good reduction in the energy consumption this technology has **to be applied to all the components of the microprocessor design and thus, requires a lot of hardware, circuitry and new design methodology.** The authors have demonstrated this technique on a multiplier and a simulated adder. But, the objective to achieve this for the whole microprocessor in design and production would have to face a lot of **complexities and challenges** (for example, complex flip flops and additional circuitry for the whole microprocessor) and would result in area overhead also. Processor designing

requires years to design and verify a new processor. Thus, it requires a lot of efforts and investment to design razored microprocessor which would result in complex implementation. Even after that, we are not sure whether we would be able to achieve satisfactory energy savings as the authors didn't explore it at such large extent and thus, are not much aware of the practical challenges at global level. Challenges in dealing with metastability and timing constraints also make it less practical.

4) COMPARISON WITH PAPERS STUDIES IN THE CLASS: This paper uses the concept of redundancy studied in the class by using a shadow latch to double-sample the values at pipeline stages. The concept of redundancy has been used before in the DIVA paper which was published around 4 years before this paper. Both the papers try to detect and recover from the errors. So, we can say that the basic idea of redundancy and error detection/recovery is borrowed from the DIVA paper (and some other papers of the time). But, the objective and the use of redundancy are different. The main objective of razor technology is to minimize the energy consumption by regulating the voltage based on the delay error rates. Detection and recovery is an additional benefit of this. On the other hand, DIVA was proposed with a key objective of detecting and recovering from the errors.
DIVA authors suggested that it can be used to minimize the voltage margins by monitoring the error rates but they didn't explore this. It is possible that the Razor borrowed this general idea from DIVA approach and proposed a specific technique. DIVA attempts to fix the errors in the pipeline by re-executing all the instructions while Razor doesn't do the same. Razor just takes the pipeline stage value twice and compares them. DIVA doesn't modify the main pipeline as it just augments the checker at the end of the pipeline. Razor need the design modification for the main pipeline as shadow latch is introduced at each pipeline stage. Another point, Razor just tries to detect and fix the delay errors while DIVA covers a broader category of errors. Razor mainly exploits temporal redundancy while DIVA emphasizes on the use of spatial redundancy by introducing a more reliable checker. SRT/CRT technique proposed in 2002 also uses redundancy but the goals are different here. Moreover, CRT uses the existing multi-threading techniques to detect errors while razor requires much change in processor design.

5) IMPROVEMENT: To improve upon this work, I would like to explore this area by implementing this technology on broader level. In 2014, we have better resources at cheaper price and we are capable of handling more complexities. I would like to simulate its application on the whole microprocessor (or at least a large number of components) and see how much power saving we can actually achieve. This would give a better picture of practical adaptation of this technique. Another interesting aspect can be analyzing different components/parts of microprocessor with this technique implemented and do a comparative study to finally categorize and short list the components for which we can achieve the best results. It may give a way to achieve more reduction in energy consumption while balancing the hardware cost and complexity at the same time. It would finally help to understand whether it is reasonable to apply this for the whole microprocessor design and, how different components and their interdependencies limit the usefulness of this approach. We can also try to explore some other implementations of voltage control systems and techniques to deal with metastability as well as long and short path constraints to make it more practical.