Técnica de diagnóstico de SEU utilizando diccionarios de fallos incompletos

1. Resumen.
2. Introducción.
    a. Introducir la problemática de los circuitos en espacio, decir que hay radiación ionizante, la cual produce hard errors y soft errors, y explicar lo que es un SEU. Importante decir que con la miniaturización esto también aplica en aviónica y a nivel del mar (referencia “Cosmic radiation comes to ASIC and SOC design” de Santarini y/u otras que encuentres).
    b. Dar razones de por qué puede ser interesante tener cierta capacidad de diagnóstico de SEU.
3. Estado del arte.
    a. Hacer búsquedas en IEEE Xplore por “fault location” -> vas a ver diagnóstico de fallos de fabricación, que suelen ser tipo stuck-at-0, stuck-at-1, bridge (corto entre dos señales)
    b. Revisar las referencias de los dos artículos del grupo que citamos en el paper de RADECS (de Mogollón)
    c. Buscar también por si hubiera algo de fault location / diagnosis de SEU
    d. En esta sección contaría que lo que existe principalmente es para los fallos de fabricación, que para SEU se ha hecho poquito y basado en el cálculo de códigos hash (papers de Mogollón). Explicar que el problema de la técnica basada en códigos hash es que necesita diccionarios de fallos completos/exhaustivos, y generar los diccionarios exhaustivos es inviable para cada . Dejar claro que el diccionario resultante de una campaña exhaustiva es un diccionario exhaustivo o completo, y uno al que le falten “runs”.
4. Inyección de fallos
    a. Explicar qué es la inyección de fallos
    b. Definir toda la jerga: carrera (del inglés run), inyección, localización, ciclo, daños, etc…
    c. FT-Unshades2
        i. Repaso rápido y le pones referencias
        ii. Aquí es interesante comentar que las técnicas desarrolladas en el presente trabajo se pueden utilizar con cualquier herramienta de inyección de fallos, los únicos requisitos son 1) que permita guardar los patrones de salidas erróneas completos y 2) que sea preciso con respecto a un test de radiación (que las salidas sean las mismas). El desarrollo de una herramienta que cumpla 1) y su validación frente a radiación para asegurar 2) escapa al alcance del presente trabajo.
5. Primera aproximación a una métrica apropiada, la distancia de levenshtein. (diagnóstico utilizando la distancia de Levenshtein)
    a. Explicar dist. Levenshtein
    b. Explicar cómo se construye la base de datos de distancias
    c. Explicar cómo se diagnostica
        i. Explicar qué pasa (cómo se diagnostica) si tu patrón de salidas erróneas no está en la base de datos

    d. Resultados experimentales. (puedes y debes comentar lo que sale)
        i. Diccionarios exhaustivos
        ii. Diccionarios no exhaustivos.
            1. Ver cuándo se pierde la capacidad de diagnóstico y cómo se va perdiendo (no acertamos el bit pero estamos en el mismo vector, el ciclo exacto no está entre los candidatos pero está acotado entre los candidatos, no acertamos ni el bit ni el vector, el ciclo exacto está fuera de los candidatos pero por poco (en proporción al Nº de ciclos totales))
6. Inclusión de la distancia temporal en el algoritmo de selección de candidatos.
    a. Explicar el concepto por encima usando un par de runs como ejemplo
    b. Explicar cómo se modifica el algoritmo de diagnóstico al incluir la distancia temporal
    c. Resultados experimentales (mismos sub-apartados que en el cap anterior)
7. Cálculo de la distancia en FF. Mejora de la distancia temporal.
    a. Explicar el concepto (todavía no tenemos claro cómo calcularlo), seguramente usando un par de runs como ejemplo
    b. Explicar cómo se modifica el algoritmo al incluir la distancia en flip-flops
    c. Resultados experimentales (mismos sub-apartados que en el cap anterior)
8. Campañas iterativas a partir de los candidatos seleccionados del diccionario parcial.
    a. Explicar en qué consiste: comentar que cuando el espacio de inyección es condenadamente grande, una campaña “grande” (de 20000 runs) puede ser menos de un 0.01 % del diccionario exhaustivo. Entonces la idea es hacer un random sampling y a partir de ahí acotar
    b. Resultados experimentales (no tienen por qué tener los mismos apartados que los de los capítulos anteriores)
9. Conclusiones y trabajos futuros.
    a. Conclusiones:
        i. Se ha hecho X, Y, y funciona muy bien
        ii. Se ha visto que ocurre A, B, C (la distancia de levenshtein es una métrica válida para el diagnóstico de SEU en circuitos digitales (complejos), que la capacidad de diagnóstico está relacionada con / se pierde cuando ocurre tal…)
    b. Trabajos futuros
        i. Cosas que se podrían mejorar (velocidad de los cálculos, uso de matriz dispersa para reducir el consumo de memoria), o si no hay nada, al menos contrastar esto en radiación.
10. Referencias
    a. Referenciar todos los conceptos que salgan en la introducción y estado del arte (preguntar a María la versión exacta de tnt que usó para las campañas y citar al manual de TNT (y posiblemente al de UFF también) para esa versión. Los manuales están en : http://walle.us.es/doc/)

Acelerar la convergencia de las campañas iterativas (No se si esto podría ser materia para otro punto de alguna forma).

Ante la información de salida de candidatos para runs de circuitos aun comprensibles humanamente a simple vista es fácil (a veces más que otras) determinar el ciclo correcto en caso de que todos apunten al mismo bit o incluso estimar medianamente el registro y bit en caso de que la exhaustividad del diccionario sea baja y se tengan unos candidatos no unánimes ante registro y/o bit.
Para circuitos de estos y varios registros candidatos al menos es medianamente apostable cual de los registros seguro que no es. Por eso me pregunto si se podría hacer algún algoritmo que hiciera este descarte de cara a acelerar la convergencia de las campañas iterativas.
Habría que pararse a pensar en qué cosas se fija una persona para tomar estas decisiones (o entrenar una red neuronal, en python hay librerías para crear redes neuronales).
Se me ocurre que esto, al ser como “predictivo”, se podría configurar con un nivel de “agresividad”.

Observación diccionarios muy incompletos (<1%) (porque me acabo de acordar, si tenemos tiempo te lo comento y si no aquí queda para septiembre)
El 0.55% para el adder no es comparable a los 0.87% con los que se hicieron los cálculos para la uart según lo veo yo, ya que el 0.55% del adder es un diccionario con 11 líneas de las 2000 totales si fuera exhaustivo y el 0.87% de la uart eran muchísimas más líneas, no tengo fresco el dato, pero aunque el porcentaje sea parecido hay más dónde comprar.

Matemáticamente los candidatos deberían de ser de la misma “calidad”, pero según los resultados que he podido observar hasta ahora, los candidatos de la uart iban por el buen camino, creo recordar que con una iteración convergió, y sin embargo los dos candidatos de los 11 que seleccionó el algoritmo para el adder no tenían mucho que ver con el target.

También habría que destacar que el adder solo tiene un registro/bus, por lo que en cuanto a acertar el bus, FF y ciclo, para el mismo orden de exhaustividad si que se quedan ambos en el mismo nivel de precisión, el bus. (Pero eso, en el adder parece peor la aproximación porque solo tiene uno).