A. Statements

First, we restate the Constraints on the symbolic starting state and *Axioms* for a bug-free processor for completeness of this document.

Constraint C-1: There exists a time T_C, at which all SIF instructions commit (i.e., no SIF instruction can write to the architectural state after T_C), while all Symbolic QED instructions commit after T_C.

Constraint C-2: At T_C, the architectural state is QED consistent. Further, features besides instructions (such as test modes) cannot write to architectural state after T_C (e.g., scan must be turned off for registers).

Constraint C-3: All Symbolic QED operand data of every Symbolic QED instruction I_1, must have one of the following properties:

i) if the operand data is available (i.e., I_1 has already read data for this operand) then this data must match the corresponding register/memory location (i.e., source operand location) data at T_C.

ii)[4] if the operand data is not available at T_C, then I_1 is waiting for the result of an earlier Symbolic QED instruction for this operand data.

It will be proven that the above Constraints form a sufficient condition to guarantee no false positives, given that any bug-free design satisfies the following two *Axioms* after T_C.

Axiom-1: Let I_1 and I_2 be two Symbolic QED instructions given abstractly as:

$$I_1 : d \;\; op \;\; s_1,\, s_2, \ldots,\, s_m$$
$$I_2 : d' \;\; op \;\; s'_1,\, s'_2, \ldots,\, s'_m\,.$$

Here op represents an operation performed on data stored in the operand locations (s_1, s'_1 etc.), and the instructions write the computed result to a destination location in the architectural state (e.g., $d,\ d'$). Let $data(s_1,\ I_1)$ be a notation to represent the operand s_1 data used by instruction I_1 for computing the result, (Note the actual data for the source operand may be either obtained from an architectural state, e.g., architectural register, or a micro-architectural state (e.g., by data forwarding). $data(s,\ I)$ abstracts away these extra details and only represents the data value used by instruction I for operand s.) and $val\left(T_{I_1},\ d\right)$ represents the value in the architectural location d immediately after instruction I_1 commits at time T_{I_1} (i.e., the value written by instruction I_1 to location d).

Now, *Axiom-1* states: If two Symbolic QED instructions I_1, I_2 have the same op and $\forall i \in \{1,\ldots,m\}$, $data(s_i,\ I_1) = data(s'_i,\ I_2)$, then $val\left(T_{I_1},\ d\right) = val\left(T_{I_2},\ d'\right)$.

This *Axiom* simply states that when a Symbolic QED instruction executes twice on the same data, then it should result in the same outputs.

Axiom-2: Let I_1 and I_2 be two Symbolic QED instructions given as below:

$$I_2 : s_i \;\; op_2\, s'_1,\, s'_2, \ldots,\, s'_m$$
$$I_1 : d \;\; op_1\, s_1,\, s_2, \ldots,\, s_m,$$

where $i \in \{1,\ldots,m\}$ and I_2 is the last Symbolic QED instruction with which I_1 has Read-after-Write (RAW) dependency for operand s_i. Then:

$$\text{if } I_1, I_2 \text{ are as above, } data(s_i,\ I_1) = val\left(T_{I_2},\ s_i\right).$$

Further, if there are no earlier Symbolic QED instructions writing to an operand s_i of I_1, where $i \in \{1,\ldots,m\}$, then $data(s_i,\ I_1) = val(T_C, s_i)$.

In words, this *Axiom* states that when any operand of a Symbolic QED instruction has RAW dependency with any earlier Symbolic QED instruction(s), it should obtain the correct source value (which is the result of the last instruction it is dependent on). Also, when an operand of a Symbolic QED instruction does not have dependencies with earlier Symbolic QED instructions, it obtains the correct source value from the architectural state at time T_C, to be used in its computation.

We can now precisely state the main Theorem of the paper.

Theorem-1: Let Constraints C-1, C-2, and C-3 be satisfied by a symbolic starting state of a processor core. Let *Axioms 1* and *2* hold after T_C for any bug-free design of the core (even if initialized at an unreachable starting state). If any QED property fails, then the failure must be caused by a bug in the design.

Proof Outline: See below for the full proof. We first define notation for a sequence of Symbolic QED instructions in a QED-compatible bug trace. Next, we isolate the first pair of Symbolic QED instructions which cause a QED property failure. We decompose the execution of these two instructions into a union of six mutually disjoint cases. For each case, we give a proof by contradiction (of one or more *Axioms*) that there must be a bug in the design, thus concluding the proof of the Theorem. ■

Next, a Corollary for partially-correct designs follows.

Corollary-1 (partially-correct designs): Let Constraints C-1, C-2, and C-3 be satisfied by a symbolic initial state of a processor core. If *Axiom-1* holds after T_C for any bug-free design of the core and a QED property fails, then the design cannot satisfy *Axiom-2*. Likewise, if *Axiom-2* holds after T_C for any bug-free design of the core and a QED property fails, then the design cannot satisfy *Axiom-1*.

Proof Outline: This result follows directly from the proof technique of Theorem-1. We again isolate the first pair of Symbolic QED instructions that causes a QED property failure and decompose the execution of these two instructions into two classes. In the first class, *Axiom-1* must be violated and in the second class, *Axiom-2* must be violated. The result follows.
■

B. Full Proofs

Theorem-1: Let Constraints C-1, C-2, and C-3 be satisfied by a Symbolic initial state of a processor core. Let *Axioms 1* and *2* hold for any bug-free design of the core (even at an unreachable starting state). If any QED property fails, then the failure must be caused by a bug in the design.

Proof: Let ▯ $= \{O_1,\ D_1,\ O_2, D_2,\ \dots,\ O_n, D_n\}$ be a sequence of Symbolic QED instructions chosen by the BMC tool that fails at least one QED property, i.e., these instructions are given in a QED-compatible bug trace. Here $\{O_i\}_{i=1}^{n}$ represents the subsequence of original instructions and $\{D_i\}_{i=1}^{n}$ represents the corresponding subsequence of duplicate instructions (i.e., using duplicate registers and memory space). Indexing of instructions in either of the two subsequences corresponds to the order in which they commit to architectural state. We can assume without loss of generality that original and duplicate instructions are pairwise-interleaved as written in ▯ above[8], and the first failed QED property is due to a mismatch between values in two architectural registers[9].

Now, let T_F[10] be the time when the first QED check fails. Then,

$$\exists\ (R_i,\ R_i')\ \text{s.t.}\ val(T_F,\ R_i) \neq val(T_F,\ R_i').$$

Here R_i, R_i' are registers which cause the QED check to fail, and $val(T,\ R)$ represents the value stored in register R at a given time T. Because Constraint C-2 is satisfied, we have $val(T_C,\ R_i) = val(T_C,\ R_i')$. Then, it must be true that

$$\exists a \in \{1,\dots,n\}\ \text{s.t.}\ \left(val\left(T_{O_a},\ R_i\right) \neq val\left(T_{D_a},\ R_i'\right)\right)$$
$$\wedge \left(\bigwedge_{b\in\{0,\dots,a-1\}} val\left(T_{O_b},\ R_i\right) = val\left(T_{D_b},\ R_i'\right) \right), \qquad (1)$$

where T_{O_a} is the time where O_a writes (commits) to its destination register R_i and T_{D_a} is the time where D_a writes to its destination register R_i'. Explicitly, (O_a, D_a) is the first pair of instructions within ▯ that force the architectural state to be QED inconsistent. The equation (1) is true because there must exist some Symbolic QED instructions which write to the register pair R_i, R_i' such that the QED check fails for these registers, as Constraint C-1 ensures that all SIF instructions complete by T_C.

We can abstractly represent instructions O_a and D_a as below:

$$O_a : R_i\ \ \text{op}\ \ s_1,\ s_2, \dots,\ s_m$$
$$D_a : R_i'\ \ \text{op}\ \ s_1',\ s_2', \dots,\ s_m'.$$

Here, op represents an operation performed on data stored in some source locations[11] (s_1, s_1') etc., and each instruction writes a computed result to its destination register. Because (O_a, D_a) are a pair of Symbolic QED instructions, they implement the same operation. Also, at any time point when the architectural state is QED-consistent (e.g., T_C from C-2), we must have $val(T_C,\ R_i) = val(T_C,\ R_i'), val(T_C,\ s_1) = val(T_C,\ s_1')$ etc.

When O_a and D_a execute, *only one* of two conditions are satisfied:

$$\textbf{(A)}\ \forall j \in \{1,\dots,m\},\ data(s_j,\ O_a) = data(s_j',\ D_a)$$

$$\textbf{(B)}\ \exists j \in \{1,\dots,m\}\ s.t.\ \ data(s_j,\ O_a) \neq data(s_j',\ D_a).$$

Here, $data(s_j,\ O_a)$ represents the data which instruction O_a uses for its operand s_j during computation (as defined in Sec. 2.1.).

First, assume **(A)** holds. Then all operand data of O_a and D_a match:

$$O_a:\ R_i\ \text{op}\ data_1,\ data_2,\dots,\ data_m$$
$$D_a:\ R_i'\ \text{op}\ data_1,\ data_2,\dots,\ data_m.$$

However, we also know that $val(T_{O_a},\ R_i) \neq val(T_{D_a},\ R_i')$. This implies that the same operation (op) performed on the same data results in two different values when executed at two different times, contradicting *Axiom-1* of any bug-free design. Therefore, for case **(A)**, Theorem-1 holds.

Next, assume **(B)** holds instead of **(A)**. Depending on when the operands' data are available for the instruction to compute on, we have five mutually disjoint subcases for **(B)** (for each subcase, we show that Theorem-1 holds):

(B.1) At T_C, data for both operands s_j and s_j' is available.

As Constraint C-3 holds, we know that $data(s_j,\ O_a) = val(T_C,\ s_j)$ and $data(s_j',\ D_a) = val(T_C,\ s_j')$. From Constraint C-2, the architectural state at T_C is QED-consistent. Therefore, $val(T_C,\ s_j) = val(T_C,\ s_j')$, which contradicts the assumption of **(B)**, since $data(s_j,\ O_a) \neq data(s_j',\ D_a)$. Thus, **(B.1)** cannot arise when QED constraints are in place.

(B.2) At T_C, data for only one operand (pick s_j without loss of generality) is available.

If **(B.2)** holds, *only one* of the following two cases must arise: **(B.2.1)** There are no earlier Symbolic QED instructions writing to s_j'; **(B.2.2)** There is at least one earlier Symbolic QED instruction with which D_a has RAW dependency for source operand s_j'. Assume case **(B.2.1)** holds. Then, from Constraints C-2 and C-3, at T_C, $data(s_j,\ O_a) = val(T_C,\ s_j)$ and $val(T_C,\ s_j) = val(T_C,\ s_j')$. If *Axiom-2* holds, we have $data(s_j',\ D_a) = val(T_C,\ s_j')$, which implies by transitivity of equality that $data(s_j,\ O_a) = data(s_j',\ D_a)$. However, this contradicts the assumption of **(B)** that $data(s_j,\ O_a) \neq data(s_j',\ D_a)$. Thus, for case **(B.2.1)**, if a QED property fails, it must be caused by a bug in the design.

Now assume that **(B.2.2)** holds instead of **(B.2.1)**. Let D_x be the last instruction that D_a has RAW dependency on for source operand s_j'. From *Axiom-2*, we have $data(s_j',\ D_a) = val(T_{D_x},\ s_j')$. Let O_x be the corresponding original QED instruction for D_x. Note that O_a must have RAW dependency with O_x. Therefore, from *Axiom-2* we again have $data(s_j,\ O_a) = val(T_{O_x},\ s_j)$. We also have from (2), $val(T_{O_x},\ s_j) = val(T_{D_x},\ s_j')$. Hence, by transitivity, $data(s_j,\ O_a) = data(s_j',\ D_a)$. However, this contradicts the assumption of **(B)**, i.e., $data(s_j,\ O_a) \neq data(s_j',\ D_a)$. Thus, *Axiom-2* cannot hold and for case **(B.2.2)**, if a QED property fails, it is caused by a bug in the design.

(B.3) At T_C, data for both operands s_j and s_j' are not available.

If case **(B.3)** holds, *only one* of two cases must arise: **(B.3.1)** There is no earlier Symbolic QED instruction pair that write to s_j, s_j'; **(B.3.2)** There is an earlier Symbolic QED instruction pair O_x, D_x that write to s_j, s_j' that is the last pair O_a, D_a have RAW dependencies on for these operands.

Next, assume that **(B.3.1)** holds. From *Axiom-2*, we have $data(s_j,\ O_a) = val(T_C,\ s_j)$ and $data(s_j',\ D_a) = val(T_C,\ s_j')$. From Constraint C-2, we also know that $val(T_C,\ s_j) = val(T_C,\ s_j')$. Thus, by transitivity, we have $data(s_j, O_a) = data(s'_j, D_a)$, but this contradicts **(B)**. Hence, in case **(B.3.1)** we also conclude that if a QED property fails, it is caused by a bug in the design.

Finally, assume that **(B.3.2)** holds. From *Axiom-2* we know that $data(s_j, O_a) = val(T_{O_x}, s_j)$ and $data(s'_j, D_a) = val(T_{D_x}, s'_j)$. From (3) we have $val(T_{O_x}, s_j) = val(T_{D_x}, s'_j)$, and then by transitivity, we have $data(s_j, O_a) = data(s'_j, D_a)$. This contradicts the assumption of **(B)**. Thus, for **(B.3.2)** we also conclude that if a QED property fails, it is caused by a bug inside the design.

We have shown that in each of the six possible mutually disjoint cases, a QED property failure can only be caused by a bug in the design. This proves Theorem-1. ∎

Corollary-1 (partially-correct designs): Let Constraints C-1, C-2, and C-3 be satisfied by a symbolic initial state of a processor core. If *Axiom-1* holds after T_C for any bug-free design of the core and a QED property fails, then the design cannot satisfy *Axiom-2*. Likewise, if *Axiom-2* holds after T_C for any bug-free design of the core and a QED property fails, then the design cannot satisfy *Axiom-1*.

Proof: Assume a QED property fails and again consider the setup in the proof of Theorem-1 for the QED-compatible bug trace. Again, let (O_a, D_a) be the first pair of instructions within that force the architectural state to be QED inconsistent. Consider again the representation of these two instructions as O_a and D_a as below:

$$\begin{aligned} O_a &: R_i \ \ op \ \ s_1,\ s_2, \ldots,\ s_m \\ D_a &: R'_i \ \ op \ \ s'_1,\ s'_2, \ldots,\ s'_m. \end{aligned}$$

Here, op represents an operation performed on data stored in some source locations (s_1, s'_1) etc., and each instruction writes a computed result to its destination register. Because (O_a, D_a) are a pair of Symbolic QED instructions, they implement the same operation. Also, at any time point when the architectural state is QED-consistent (e.g., T_C from C-2), we must have $val(T_C, R_i) = val(T_C, R'_i), val(T_C, s_1) = val(T_C, s'_1)$ etc.

When O_a and D_a execute, *only one* of two conditions are satisfied:

$$\textbf{(A)}\ \forall j \in \{1,\ldots,m\},\ data(s_j, O_a) = data(s'_j, D_a)$$

$$\textbf{(B)}\ \exists j \in \{1,\ldots,m\}\ s.t.\ \ data(s_j, O_a) \neq data(s'_j, D_a).$$

In the proof of Theorem-1, we showed that a QED failure in condition **(A)** cannot happen if *Axiom-1* holds, and a QED failure in any of the five subcases of **(B)** cannot happen if *Axiom-2* holds. Therefore, in designs where *Axiom-1* must hold, only condition **(B)** can occur if a QED failure happens, and in designs where *Axiom-2* must hold, only condition **(A)** occurs if a QED failure happens. This proves the result. ∎

[8]Interleaving can be [O1, O2, …, On, D1, D2, …, Dn], [O1, D1, O2, D2, …, On, Dn], etc. If the relative orderings of subsequences $\{O_i\}_{i=1}^{n}$ and $\{D_i\}_{i=1}^{n}$ is preserved, there is no impact on proof.

[9]It could alternatively be due to a mismatch in data at two memory locations, with no impact on proof.

[10]T_F is a later time than T_C, as T_F is when all the Symbolic QED instructions commit.

[11]Register or memory locations.