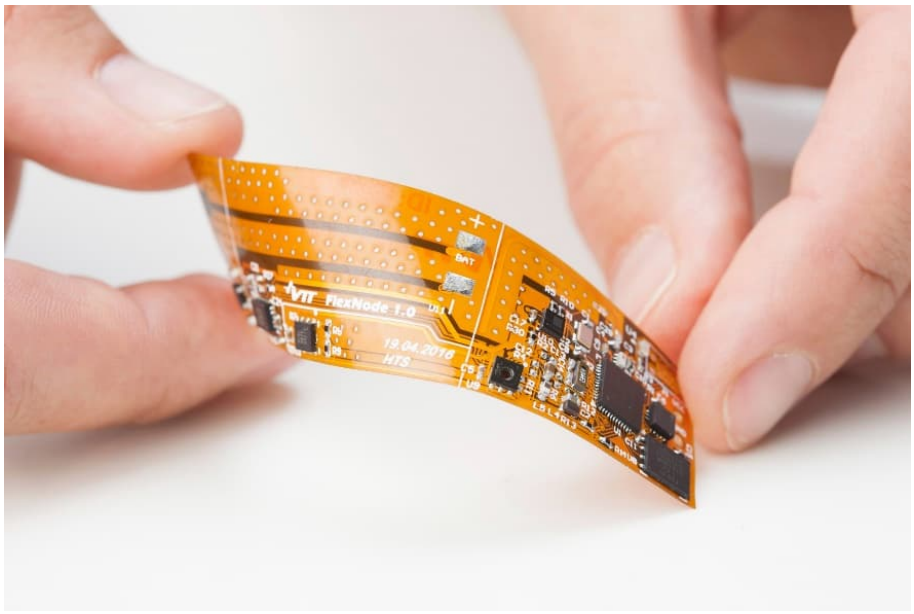
RESEARCH REPORT

VTT-R-00422-25



Performing computations for Digital I&C related CCFs in PRA

Authors: Kim Björkman

Confidentiality: VTT Public

Version: 5.11.2025

Report's title Performing computations for Digital I&C related CCFs in PRA	
Customer, contact person, address VYR	**Order reference** SAFER 6/2025
Project name Probabilistic Risk Assessment Labour, Improvements and Extensions	**Project number/Short name** 141374/PRALINE
Author(s) Kim Björkman	**Pages** 17/
Keywords probabilistic risk assessment, common cause failure, digital instrumentation and control system	**Report identification code** VTT-R-00422-25

Summary

In this study, we present a work process with tool support for a simplified common cause failure (CCF) modelling approach developed for digital instrumentation and control system related CCFs in probabilistic risk assessment (PRA). In the approach, CCF calculations are performed outside the PRA tool and only high-level CCF basic events are included in the PRA model.

The work process contains three high level steps: 1) hardware failure basic events probability computation, 2) CCF computation, and 3) incorporating the results to a PRA tool. Step 2 is further divided into several steps: a) identification of system level effects of different CCF combination events, b) combinatorial calculations, c) CCF calculations, and d) CCF basic event probability computations. For step 2, tool support is developed.

The prototype tool is a python program with an interface with Excel where the input data is defined and where results are written. Selected parts of the DIGMORE (international benchmark study) example case are used to test the prototype tool. The use of the tool simplifies the work process by decreasing both the amount of manual labour needed and the likelihood of errors in the actual computation and in incorporating the results in a PRA model.

Confidentiality	VTT Public

Espoo 5.11.2025

Written by	**Reviewed by**
Kim Björkman, Research Scientist	Tero Tyrväinen, Research Scientist

VTT's contact address
VTT Technical Research Centre of Finland Ltd, P.O. Box 1000, FI-02044 VTT, FINLAND

Distribution (customer and VTT)
SAFER2028 TAG1.1 members, VTT archive

Approval

VTT TECHNICAL RESEARCH CENTRE OF FINLAND LTD

Date:	06 November 2025
Signature:	DocuSigned by: Teemu Kärkelä E7B76042F134471...
Name:	Teemu Kärkelä
Title:	Research Team Leader

Contents

Abbreviations

Abbreviation	Meaning
APU	Acquisition and processing unit
CCCG	Common cause component group
CCF	Common cause failure
CL	Communication link
CSNI	Committee on the Safety of Nuclear Installations
DRPS	Diverse reactor protection system
HW	Hardware
I&C	Instrumentation and control
NEA	Nuclear Energy Agency
OECD	Organisation for Economic Co-operation and Development
PAC	Priority and actuation control
PM	Processor module
PRA	Probabilistic risk assessment
PRPS	Primary reactor protection system
VU	Voting unit

1. Introduction

Modelling digital instrumentation and control (I&C) systems in probabilistic risk assessment (PRA) is a challenging task. The challenge arises, e.g., from the complexity of the systems, the lack of available failure data, and from software. A particular challenge is the modelling of common cause failures (CCF), both hardware and software CCFs. Besides the lack of relevant data, the common cause component group (CCCG) sizes can be large and the systems including the components can have asymmetric success criteria.

The PRA reference case in Organisation for Economic Co-operation and Development (OECD) Nuclear Energy Agency (NEA) Committee on the Safety of Nuclear Installations (CSNI) Working Group on Risk Assessment task called DIGMORE (A realistic comparative application of digital I&C modelling approaches for probabilistic safety assessment) considers an I&C architecture covering several subsystems that include many identical components (OECD NEA CSNI, 2025). Other aspects included in the reference case are, e.g., priority logic and spurious actuations.

In (Tyrväinen & Björkman, 2024), a PRA model for the DIGMORE reference case was developed. Because of the challenges related to the CCF calculations (e.g., some CCF groups include more than 8 components), a simplified PRA model where CCFs were included as high-level failure events was developed, and the complex calculations were performed in the background. In the base case, (Tyrväinen & Björkman, 2024) applied the alpha-factor model for hardware CCFs. Since the computations were quite complex, also the use of the modified beta-factor model in complementary analyses was studied. A challenge in performing the CCF computations was the lack of tool support. In the case study, Excel was used for the computations but without a structured work process performing the computations and importing the results to a PRA tool was a cumbersome and error prone task.

In this work, we develop a structured work process with tool support to perform the CCF computations and to import the results to a PRA tool. The rest of the document is structured as follows. We summarize the CCF modelling approach and present the developed work process in section 2. In section 3, we discuss tool support for the work process. In section 4, we conclude this study.

2. Method description and work process

Tyrväinen & Björkman (2024) present a simplified CCF modelling approach for the DIGMORE reference case (OECD NEA CSNI, 2025). The reference case contains CCF groups with large number of components, e.g., there are 28 Priority and actuation control (PAC) units of two diverse types. If the alpha factor model is used (as recommended in the reference case description), performing such CCF calculations within the PRA model is not feasible.

In the developed approach, CCF calculations (including, e.g., the probabilities of the hardware (HW) CCF basic events) are performed outside the PRA tool (e.g. in Excel) and only high-level CCF basic events are included in the PRA model. Only CCF basic events that cause one or several safety functions to fail are explicitly included in the PRA model. The CCFs that have the same system level effect are merged into the same basic event. In addition to normal alpha-factor computations, quite complex combinatorial calculations are required to manage the CCF combinations with group sizes of, e.g., 8 and 16.

Based on the experience from (Tyrväinen & Björkman, 2024), the application of the simplified approach may not be completely straightforward. A challenge in performing CCF computations is the lack of tool support, and, thus, a lot of manual work is needed that can be prone to errors. In addition, the required combinatorial calculations are not trivial. Excel was used for the computations but without a structured work process performing the computations and importing the results to a PRA tool can be demanding. This is highlighted when performing sensitivity studies.

The following subchapters present the steps of the work process we developed to manage the application of the method (see Figure 1). The main steps are (the focus is on hardware failure events):

1. Hardware failure basic events probability computation
2. Common cause failure computation
3. Incorporating the results to a PRA tool

For a more in-depth description of the simplified approach, we refer the reader to (Tyrväinen & Björkman, 2024).

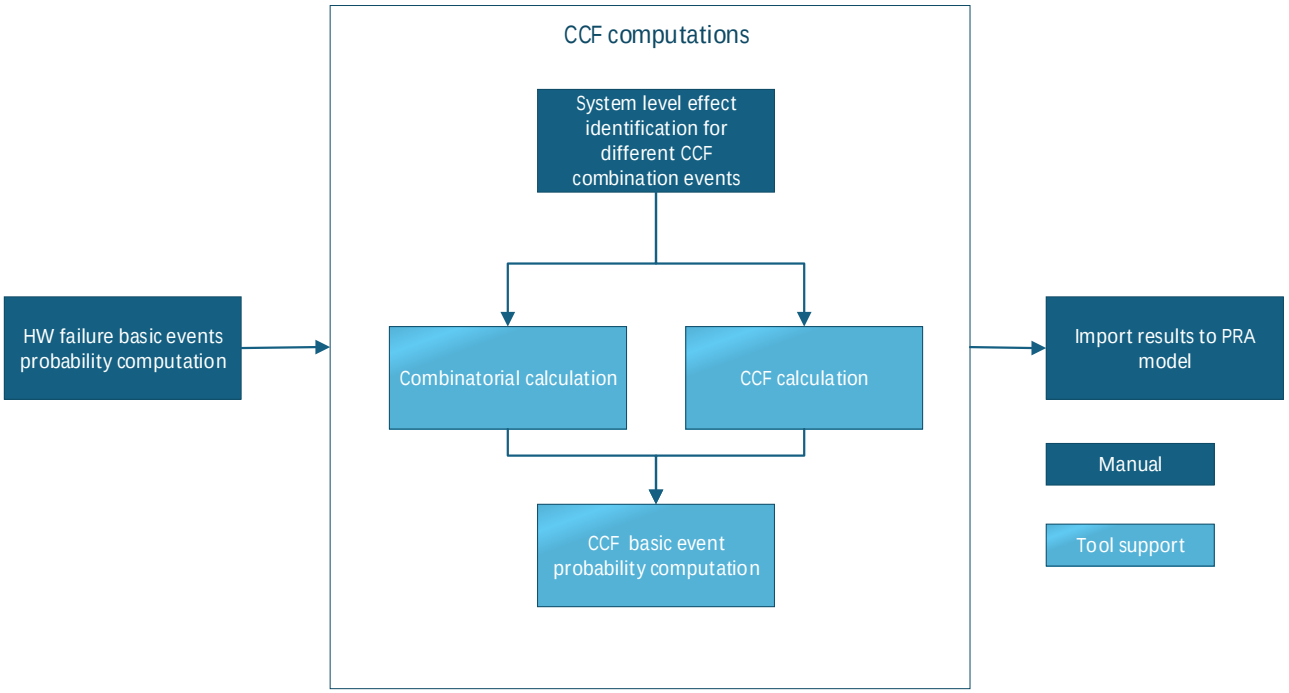


Figure 1. CCF computation steps.

2.1 Computing probabilities for hardware failure basic events

Hardware basic event failure probabilities are an input for the HW CCF computation. The approach to compute the probabilities depends, e.g., on the available data and the selected modelling approach.

In (Tyrväinen & Björkman, 2024), the computation of HW failure probability was divided into two parts; 1) unavailability before and 2) unavailability after detection. In the computation of unavailability before detection, it might be necessary to compute several probabilities to account for different

detection mechanisms, their detection coverage, and their failures. The total failure probability for HW failure is the sum of unavailability after detection and unavailabilities before detection.

The different unavailabilities could alternatively be modelled with separate basic events. In addition, they could at least in theory have different CCF parameters.

2.2 Performing common cause failure computations

2.2.1 CCF calculations for large common cause component groups

The CCF calculation process in divided into further steps that are presented in the following subchapters.

1. Identification of system level effects of different CCF combination events
2. Combinatorial calculations
3. CCF calculations
4. CCF basic event probability computations

In the description of the steps, the alpha-factor model is used, but the same steps are applicable for other commonly used CCF-models as well.

The CCFs related to the primary reactor protection system (PRPS) voting unit (VU) communication link (CL) modules of the DIGMORE reference case are used as an example in the description. The PRPS consists of two subsystems (PRPS-A and PRPS-B) both containing four divisions. Each division contains a voting unit, and each VU contains a communication link module (each division contains also other units, and the units contains also other modules). Thus, each subsystem contains four VU CL modules. The failure criterion in one subsystem is 3 out of 4. Since both subsystems are based on the same technology, the size of the PRPS VU CL common cause component group is eight.

2.2.1.1 Identification of system level effects of different CCF combination events

In this step, possible system level effects of different CCF combination events are identified. CCF combination events with the same system level effect will be combined for PRA modelling.

For PRPS VU CL module CCF, three relevant system level effects are identified for different CCF combination events:

1. PRPS-A fails but not PRPS-B (at least three VU CL modules fail in PRPS-A but not in PRPS-B)
2. PRPS-B fails but not PRPS-A (at least three VU CL modules fail in PRPS-B but not in PRPS-A)
3. Both PRPS-A and PRPS-B fail (at least three VU CL modules fail in both in PRPS-A and PRPS-B)

2.2.1.2 Combinatorial calculations

Combinatorial calculations are performed for CCF combination event groups with the same system level effect. For each possible number of failed components in a CCF, the purpose is to compute the number of all combinations that have the same (sub)system level effect. We denote this value as $C_{s_j \ldots s_l / k}$, where s_i means that i:th subsystem must fail in combination, and *k* is the total number of failures in the system.

Table 1 shows the number of PRPS VU CL CCF combination events causing failure of one PRPS subsystem or both subsystems. Each CCF combination event is included only in one $C_{s_j \ldots s_l / k}$ value (the one with the most subsystem failures).

Table 1. Numbers of PRPS VU CL CCF combination events causing failure of one PRPS subsystem or both with 3-o-o-4 criterion (modified from (Tyrväinen & Björkman, 2024)).

Number of failures (k)	Only PRPS-A fails ($C_{s_A/k}$)	Only PRPS-B fails ($C_{s_B/k}$)	Both PRPS-A and PRPS-B fails ($C_{s_A s_{AB}/k}$)
1			
2			
3	4	4	
4	17	17	
5	28	28	
6	6	6	16
7			8
8			1

2.2.1.3 CCF calculations

CCF calculations are performed according to the selected CCF model. For the alpha factor model, the calculations are performed according to $Q_k = \frac{n}{\binom{n}{k}} \frac{\alpha_k}{\mu_\alpha} Q_t$, $\mu_\alpha = \sum_{k=1}^{n} k\alpha_k$, where α_k is the total probability of having a failure of multiplicity *k*, *n* is the size of the CCF group, Q_t is the total probability of component failure (covering both single failure and CCFs), and Q_k is the probability of k specific components failing in a CCF.

2.2.1.4 CCF basic event probability computations

The system level effect probability is computed according to $P_{s_j \ldots s_l} = \sum_{k=1}^{n} C_{s_j \ldots s_l / k} Q_k$, where *n* is the size of the CCF group. The probability $P_{s_j \ldots s_l}$ is multiplied with a conservative factor. For example, in (Tyrväinen & Björkman, 2024) the factor 1.1 was used, i.e. 10% was added to the probability.

The conservative factor is used to ensure that the risk is not underestimated because minimal cut sets with single failures or two or more CCFs are left out, e.g., minimal cut sets including a CCF of two PRPS VU CLs and a single failure in the same subsystem of a PRPS VU CL or a processor module (PM). Alternatively, it would also be possible to account for these combinations in the computation (see also section 3.2), but it would add complexity.

2.2.2 Modified beta-factor model

Especially, for large CCF groups (more than 16 components) the modified beta-factor model and partial beta-factor method can be considered. The modified beta-factor model differs from the traditional beta-factor model by allowing a component to belong to several CCF groups.

The application of the modified beta-factor model is much simpler than the alpha-factor model because the number of combinations to be analyzed is much smaller. Step 2.2 (see section 2.2.1.2) can be completely skipped. Otherwise, the work process is quite similar, but the probability calculations are obviously different.

The approach presented in the previous subsection and the modified beta-factor method are actually quite similar. The basic events included in the PRA model may be identical using both approaches. One could even argue that the approach presented in the previous subsection translates the alpha-factor model into a modified beta-factor model (even though the basic event probabilities are directly calculated instead of the beta-factors).

Bao et al. (2022) propose the partial beta-factor method for digital I&C CCF parameter estimation. In the partial beta-factor method, the analyst gives scores (A, B, C, etc.) to several subfactors (e.g., Redundancy (& diversity) and separation) that affect the CCF probability depending on how good the defense against CCFs is. After that, the beta-factor is calculated simply by summing table values related to the scores of the subfactors.

In (Tyrväinen & Björkman, 2024), the modified beta-factor model and the partial beta-factor method were applied to specific cases where the CCF groups included more than 16 components. The report included also a separate study where all CCFs were modelled using that method instead of the alpha-factor model.

Software CCFs can be modelled assuming complete dependency (beta-factor 1) as in (Tyrväinen & Björkman, 2024) with separate failure probabilities for application software (AS) CCFs and operating system/platform software. Alternatively, the partial beta-factor method for software CCFs can be used.

Bao et al. (2022) have developed variants of the partial beta-factor method specifically to estimate beta-factors for software CCFs in non-diverse and diverse configurations. If the contribution of software CCFs to the results is not negligible when assuming complete dependency, the use of the partial beta-factor method can be considered at least in sensitivity studies.

2.2.3 CCF calculations for diverse CCF groups with large number of components

The PAC units modelling in (Tyrväinen & Björkman, 2024) was a special case that required a customized approach compared to the CCF calculations presented in section 2.2.1. In the DIGMORE reference case, there were two diverse types of PAC units: PAC-A and PAC-B. There were 14 units of each type. For each of the seven safety systems there were two PAC-A and two PAC-B units, and the failure criterion was 3 out of 4. Therefore, the approach presented in section 2.2.1 was not directly applicable.

To perform the required computations, a two-phased process (applicable also to other similar cases) was developed. In the first phase, component failures within a PAC were identified and combined with the same PAC level effect and improbable component failures were screened out. Then normal alpha-factor computations were performed for CCF events.

In the second phase, a visual studio script was used to go through all the combinations with a CCF or single failure from both groups. In addition, software failures were considered in the script. The result of the analysis is the PAC (3-out-of-4) failure probability for one specific system, for two specific systems etc.

On a general level, the analysis process presented in section 2.2.1 is also applicable in this case. However, in this case it is necessary to consider the combination of two CCF groups. In the script, steps 2-4 are considered iteratively one combination at a time.

2.3 Including results to a PRA tool

When the basic events and CCF groups have been defined (and the probabilities have been computed), the basic events need to be included in the fault trees. Fault tree modelling guidelines are out of the scope of this work.

CCF combination events with the same system level effect are merged into the same basic event. For example, for PRPS VU communication link HW there are three CCF basic events that are modelled: CCF in PRPS-A (but not in B), CCF in PRPS-B (but not in A), and CCF in both subsystems. In the DIGMORE model, the CCF in both systems is modelled by creating a CCF group and using the Q-factor model (i.e., CCF combination event probabilities are explicitly given), but a separate basic event could also have been used in the fault trees. The components of the new CCF group are subsystem specific CCF basic events.

The same approach is applicable when there are multiple diverse CCF groups with large number of components. In the PAC example case, failure of a single front-line system was modelled as a merged basic event due to PAC failure. A CCF group consisting of the single front-line system failures was used to model CCF between the front-line systems.

3. Tool support

To streamline the work process, tool support is needed. We have implemented a prototype tool to perform the common cause failure computations. The tool is currently developed to demonstrate the applicability of the work process using the DIGMORE example case (OECD NEA CSNI, 2025). We have also set requirements to extend the prototype tool towards a more general level application CCF tool and we list approach independent requirements for the tool.

3.1 DIGMORE case

3.1.1 Prototype tool

The prototype tool is a Python program with an interface with Excel to define inputs and to write results. The tool currently supports the CCF calculations (steps 2.2-2.4) presented in section 2.2.1 using the alpha-factor model both for symmetrical subgroups (i.e., subgroups include the same number of components, and the failure criterion is the same) and asymmetrical subgroups (i.e., subgroups include the different number of components, and/or the failure criterion is different).

In the input Excel, HW failure probabilities (see Figure 2), CCF-parameters, and CCF groups, their subgroups, and their failure criteria are defined (see Figure 3). Besides including the main results of the CCF computation, the results Excel includes Q_k values for each CCF group. The main results are split to basic event probabilities and CCF group definitions (for symmetrical cases) to be used in the main PRA model. A config file is used to define different analysis cases (i.e. which Excel file to use as input and where output is written), and to specify wanted output format to facilitate result import to a PRA tool (see Figure 4).

System	Unit	Module	BE/Group	Total
PRPS	APU	AI	PRPS_APU_AI	9,02E-04
PRPS	APU	PM	PRPS_APU_PM	4,68E-04
PRPS	APU	CL	PRPS_APU_CL	2,33E-03

Figure 2. Input for hardware basic event probabilities

System	Unit	Module	Group	Subgroup	Subgroup size	Subgroup Failure criterion
PRPS	APU	AI	PRPS_APU_AI	1	4	3
PRPS	APU	AI	PRPS_APU_AI	2	4	3
PRPS	APU	AI	PRPS_APU_AI	3	4	3
PRPS	APU	AI	PRPS_APU_AI	4	4	3
PRPS	APU	PM	PRPS_APU_PM	1	4	3
PRPS	APU	PM	PRPS_APU_PM	2	4	3
PRPS	APU	CL	PRPS_APU_CL	1	4	3
PRPS	APU	CL	PRPS APU CL	2	4	3

Figure 3. Input for CCF subgroup definitions.

```
{"result_format": {
    "FinPSA" : {
        "BE_DATA" : {
            "HEADER" : " ---------- Data records <transfer data> ----------",
            "FIELDS" : [
                {"name" : "Name"},
                {"name" : "RelP1"},
                {"name" : "CCF Group"}
```

Figure 4. Config file output format definition for CCF basic event results.

The current tool supports also the computation of hardware basic event probabilities, as presented performed in (Tyrväinen & Björkman, 2024). In this case, the data needed in the computations are defined in the input Excel (e.g., failure rate and failure detection coverages). The use of precomputed HW basic event probabilities is defined in the input Excel.

3.1.2 Computational algorithm

The algorithm is structured according to steps 2.2-2.4 of the work process (see section 2.2.1). The steps are performed one CCCG at a time.

In the combinatorial computations, all combinations of component failures within the CCF group are gone through one by one. For each combination,

1. The number of component failures of each subgroup and the total number of failures are collected.
2. The combination of subgroups (can contain only one subgroup) whose failure criterion is fulfilled is collected.
3. The subgroup combination is added to the results vector based on the total number of failures.

The result is the number of combinations for different subgroup combinations for different number of failures (see Table 1 for an example). The algorithm for combinatorial computations is presented in Appendix A.

CCF calculations are performed according to the used CCF model. In the DIGMORE case, alpha-factor model is used. The CCF basic event probability computations are performed according to section 2.2.1.4.

3.1.3 Results

We evaluated the prototype tool by performing CCF calculations for PRPS and diverse reactor protection system (DRPS) hardware modules (excluding spurious failures). Different types of CCF cases are presented in Table 2.

Table 2. CCF cases, number of subgroups and subgroup size. All subgroups are symmetrical with each other, and the failure criterion is 3 out of 4 for each subgroup.

CCF case	Subgroups	Subgroup size
CCFs between identical modules in the PRPS (except AI modules)	2	4
CCFs between AI modules in the PRPS	4	4
CCFs between identical PRPS sensors	1	4
CCFs between identical modules and sensors in the DRPS (except for sensor CL modules)	1	4

In total, we performed calculations for 21 CCF cases. The computation time was ca. 1 s. In all cases, the probabilities were identical with those computed with Excel in (Tyrväinen & Björkman, 2024). Figure 5 shows an example of the final output. The data records part shows the (CCF) basic events (failures of single subgroups), their probability (RelP1), and the CCF group they belong to (to model failures of multiple subgroups). The common cause failure part illustrates the CCF groups, their CCF model (see section 2.3), and parameter values (CCFP1-CCFP3). For example, basic event named PRPS_APU_PM_1 represents CCF in PRPS-A acquisition and processing unit (APU) PM (but not in B), and CCF group PRPS_APU_PM_CCF represents CCFs in both subsystems.

---------- Data records <transfer data> ----------				---------- Common cause failures <transfer data> ----------				
Name	RelP1	CCF Group		Name	CCF mode	CCFP1	CCFP2	CCFP3
PRPS_APU_AI_1	2.00E-5	PRPS_APU_AI_CCF		PRPS_APU_AI_CCF	Q-factor	3.05E-6	2.27E-6	8.00E-6
PRPS_APU_AI_2	2.00E-5	PRPS_APU_AI_CCF		PRPS_APU_PM_CCF	Q-factor	7.87E-6	0	0
PRPS_APU_AI_3	2.00E-5	PRPS_APU_AI_CCF		PRPS_APU_CL_CCF	Q-factor	3.92E-5	0	0
PRPS_APU_AI_4	2.00E-5	PRPS_APU_AI_CCF		PRPS_VU_DO_CCF	Q-factor	1.57E-5	0	0
PRPS_APU_PM_1	1.50E-5	PRPS_APU_PM_CCF		PRPS_VU_PM_CCF	Q-factor	7.76E-6	0	0
PRPS_APU_PM_2	1.50E-5	PRPS_APU_PM_CCF		PRPS_VU_CL_CCF	Q-factor	3.92E-5	0	0
PRPS_APU_CL_1	7.49E-5	PRPS_APU_CL_CCF		PRPS_SR_CCF	Q-factor	3.92E-7	0	0

Figure 5. An example of the output of the prototype tool. The results are written in a format that can directly be copied into FinPSA tool.

CCFs between DRPS sensor CL modules were excluded from the analysis because the DIGMORE case does not specify enough alpha-factor parameters to perform the calculations. There are 20 CL

modules related to the DPRS sensors. The partial beta-factor method was applied in (Tyrväinen & Björkman, 2024).

Performing the combinatorial computations is the most time-consuming phase. To test the scalability of the algorithm, we performed the combinatorial computations for the 20 DPRS CL sensor modules case and for a hypothetical CCCG of size 24. The calculation times were ca. 2 seconds and ca. 45 seconds respectively.

3.2 CCF tool requirements

In the development of a more generally applicable tool, several general level requirements need to be accounted for. These include, e.g., traceability of results, ease of use, extendibility, and support for sensitivity studies and documentation. In addition, the approach discussed in section 2 could probably be applied in multi-module small modular reactor PRA CCF modelling context (e.g., inter-module CCFs). New targets for application should be accounted for in the development and, thus, the terminology should be generalized. For example, some of the headings in Figure 2 and Figure 3 are quite I&C specific an not as such applicable in, e.g., multi-module PRA context. It should be possible to represent failures of several subgroups also as basic events and not just as CCF groups as in the DIGMORE example.

The approach presented in (Tyrväinen & Björkman, 2024) only accounts for CCF combination events that directly fulfil the failure criterion. Combinations of single failures and shorter CCF combination events are not considered, e.g., combination of a CCF of two VU CLs and a single failure of another VU CL. In the approach, a conservative factor is used to ensure that the risk is not underestimated. While these combinations are difficult to calculate manually, they could be included in the computation performed by the software tool. However, this could lead to a refinement of the work process. For example, it might be necessary to perform steps 2.2-2.4 iteratively.

The tool should support CCF calculations with other common CCF models (e.g. Multiple Greek Letter model). Support for the use of partial beta factor model (Bao et al. 2022) both for hardware and especially for software CCF can be considered. However, the software CCF approach may need some refinements to be more applicable in our tool framework.

In addition, tool support could be needed for performing CCF calculations for diverse CCF groups (i.e. generalization of the PAC case presented in section 2.2.3). For example, the failure criterion for a function provided by components ABCD is three out of four, where AB belongs to one CCF group and CD to another CCF group. The scripts developed in (Tyrväinen & Björkman, 2024) would provide a basis for this development.

3.3 Related approaches

Related research regarding probabilistic modelling of CCFs in digital I&C systems has been discussed in (Tyrväinen, 2021) and (Björkman, 2023). Our aim is not to repeat those reviews but to present some related approaches that consider CCFs with large CCCGs.

Directly related to our work are the benchmark studies performed in the DIGMORE project and in its predecessor the DIGMAP project (OECD NEA CSNI 2024). OECD NEA CSNI (2024) summarizes the different CCF modelling approaches used by the partner organizations in the benchmarks study in the DIGMAP project. Some of the partners modelled all combinations of CCF logic within a CCCG (for groups up to eight components), whereas others used an abstracted CCF model (i.e., CCF events with the same impact were merged and probabilities were calculated in the background) similarly to the approach discussed in section 2. For CCCG sizes larger than eight, a separate basic

event could not be created for each combination due to software tool limitations. RiskSpectrum software handled this kind of group in a too simplified and conservative manner by merging all the combinations with at least four failures. Another approach that was used was to merge together component pairs to reduce the group size from 16 to 8.

In (Mankamo 2017), the extended Common Load Model is discussed. The model is developed especially for the treatment of dependencies and failure probabilities for highly redundant systems (size of a CCCG is more than four). The model uses a physical analogy by expressing failure conditions by stress resistance. Multiple failures occur when the load (stress) exceeds component resistances. Subgroup failure probabilities are used to define the model. The model includes four parameters, which is a clear improvement compared to commonly used CCF models, such as the alpha-factor model that adds a new parameter for each degree of redundancy. However, estimating the four parameters may not be trivial. The model includes extensions, e.g., for time-dependent modelling and asymmetric CCF groups.

Challenges related to CCFs where the set of components can be partially diverse are discussed in (Stiller et al. 2015). An approach was developed, where separate CCF subgroups were specified for the partially diverse sets of components and CCFs between the subgroups were managed through correlation between the groups. The correlation could be accounted for in the post processing of the minimal cut sets. Cut sets including correlated CCFs are re-quantified, considering that the conditional probability of an additional CCF is larger than the unconditional probability of an additional CCF.

A general area where large CCCGs are considered is multi-unit PRA, especially from inter-unit CCF perspective. Approaches to manage inter-unit CCFs are suggested, e.g., in (Kim et al. 2020) and (Soga et al. 2021).

4. Conclusions

In this study, we have presented a work process with tool support for a simplified CCF modelling approach. In the approach, CCF calculations are performed outside the PRA tool and only high-level CCF basic events are included in the PRA model. Only CCF basic events that cause one or several safety functions to fail are explicitly included in the PRA model. The CCFs that have the same system level effect are merged into the same basic event.

The developed work process contained three high level steps: 1) hardware failure basic events probability computation, 2) CCF computation, and 3) incorporating the results to a PRA tool. Step 2 was further divided into several steps: a) identification of system level effects of different CCF combination events, b) combinatorial calculations, c) CCF calculations, and d) CCF basic event probability computations. For step 2, tool support was developed.

The implemented prototype tool is a Python program that has an interface with Excel where the input data is defined and where results are written. The selected parts of the DIGMORE example case were used to test the prototype tool. The use of the tool simplifies the work process by decreasing both the amount of manual labour needed and the likelihood of errors in the actual computation and in incorporating the results in a PRA model.

The developed work process covered mainly the basic case of the simplified CCF modelling approach. Further development is needed, e.g., for diverse CCF group modelling and modified beta-factor model. In parallel with work process development, we need to enhance the tool support. In addition, requirements for a more generally applicable tool were considered.

References

Bao, H., Zhang, S., Youngblood, R., Shorthill, T., Pandit, P., Chen, E., Park, J., Ban, H., Diaconeasa, M., Dinh, N., Lawrence, S. (2022). Risk analysis of various design architectures for high safety-significant safety-related digital instrumentation and control systems of nuclear power plants during accident scenarios, INL/RPT-22-70056, Idaho National Laboratory, Idaho Falls.

Björkman, K. (2023). I&C system architecture PRA – Literature review. VTT Technical Research Centre of Finland Ltd. VTT Research Report No. VTT-R-00677-23.

Kim, D-S., Park, J. H., Lim, H-G. (2020). A pragmatic approach to modeling common cause failures in multi-unit PSA for nuclear power plant sites with a large number of units. Reliability Engineering & System Safety, Volume 195, 2020, 106739, ISSN 0951-8320, https://doi.org/10.1016/j.ress.2019.106739.

Mankamo, T. (2017). Extended Common Load Model: A tool for dependent failure modelling in highly redundant structures. Swedish Radiation Safety Authority. SSM 2017:11.

Organisation for Economic Co-operation and Development (OECD) Nuclear Energy Agency (NEA) Committee on the Safety of Nuclear Installations (CSNI). (2024). Digital I&C PSA – Comparative Application of Digital I&C Modelling Approaches for PSA, Volume 1: Main Report and Appendix A, NEA/CSNI/R(2021)14, Paris, France.

Organisation for Economic Co-operation and Development (OECD) Nuclear Energy Agency (NEA) Committee on the Safety of Nuclear Installations (CSNI). (2025). DIGMORE – a realistic comparative application of DI&C modelling approaches for PSA, Appendix A: Complete reference case descriptions. DRAFT.

Soga, S., Higo, E., Hiromichi, M. (2021). A systematic approach to estimate an inter-unit common-cause failure probability. Reliability Engineering & System Safety, Volume 215, 2021, 107802, ISSN 0951-8320, https://doi.org/10.1016/j.ress.2021.107802.

Stiller, J., Leberecht, M., Gänßmantel, G., Wielenberg, Andreas, Kreuser, A., Verstegen, C. (2015). Common cause failures exceeding ccf groups. Proceedings of the international topical meeting on probabilistic safety assessment and analysis (PSA), Sun Valley, Idaho April 26-30, 2015.

Tyrväinen, T. (2021). Probabilistic modelling of common cause failures in digital I&C systems - Literature review. VTT Technical Research Centre of Finland. VTT Research Report No. VTT-R-00728-21.

Tyrväinen, T., Björkman, K. (2024). Probabilistic risk model for digital I&C architecture. Research Report, VTT, VTT-R-00646-24. 45 p. + app. 7 p.

Appendix A: Algorithm for combinatorial computations

A Python implementation of the algorithm is shown in Figure 6 and an example of an input for the implementation is shown in Figure 7. The algorithm utilizes bitwise operations to enhance performance.

```
""" Performs combinatorial calculations """
def perform_combo_calc_asymmetrical(self, ccf_group):

    num_of_combos = int(math.pow(2, ccf_group['size']))
    combo_list = [{}] * ccf_group['size']

    """ for each combinations """
    for i in range(num_of_combos):

        tot_fails = i.bit_count() #compute failures in combination
        if tot_fails < 1:  # nothing is failed
            continue
        old_ssc = 0        # managed group size for this combination
        failed = False
        sgs_failed  =""
        """ for each sub group """
        for sub_group in ccf_group['subgroups']:
            sg_size = sub_group["size"]
            """ choose the failure bits that are relevant for this group """
            group_bits = int(math.pow(2, sg_size+old_ssc)-math.pow(2, old_ssc))
            """ get only the set bits for this group in combination i """
            sg_failure_bits = i & group_bits
            """ get number of failures (i.e. set bits for this group)"""
            sg_failures = sg_failure_bits.bit_count()
            old_ssc += sg_size
            """ check if failure criterion is  fulfilled """
            if sg_failures >= sub_group['failure_criterion']:
                if sgs_failed == "":
                    sgs_failed = f'{sub_group['name']}'
                else:
                    sgs_failed += f',{sub_group['name']}'
                failed = True
        if not failed:
            continue
        """ add failed sg into total combinations list.  """
        if combo_list[tot_fails-1] == {}:
            combo_list[tot_fails-1] = {sgs_failed:1}
        else:
            old_fails = combo_list[tot_fails - 1]
            if sgs_failed in old_fails.keys():
                old_fails.update({sgs_failed : old_fails[sgs_failed]+1})
                combo_list[tot_fails - 1] = old_fails
            else:
                combo_list[tot_fails - 1].update({sgs_failed: 1})

    return combo_list
```

Figure 6. Python implementation of the combinatorial calculations algorithm.

```
{'name' : 'prps-apu-pm',
    'size' : 8,
    'subgroups' : [{
        'name' : '1',
        'size' : 4,
        'failure_criterion' : 3
    },{
        'name' : '2',
        'size' : 4,
        'failure_criterion' : 3
    }]
}
```

Figure 7. An input example for the combinatorial computation function (ccf_group in Figure 6).