UNIVERSITY OF BRISTOL
DEPARTMENT OF COMPUTER SCIENCE
https://www.cs.bris.ac.uk

Computer Architecture (COMS10015)

Assessed coursework assignment
Encrypt

Note that:

1. This coursework assignment has a 30 percent weighting, i.e., it represents 30 percent of Credit Points (CPs) associated with COMS10015, and is assessed on an individual basis. The submission deadline is 21/11/24.
2. Before you start work, ensure you are aware of and/or adhere to various regulations[a] which govern coursework assessments: pertinant examples include those related to academic integrity (see, e.g., Sec. 3) and submission (see, e.g., Sec. 12).
3. There are numerous support resources available, for example:
   - via the unit forum, where you can get help and feedback via n-to-m, collective discussion,
   - via any lab. and/or drop-in slot(s), where you can get help and feedback via 1-to-1, personal discussion, or
   - via the staff responsible for this coursework assignment: although the above are often preferable, you can make contact in-person or online (e.g., via email).

[a] https://www.bristol.ac.uk/academic-quality/assessment/codeonline.html

1 Introduction

Imagine that two parties $\mathcal{A}$ and $\mathcal{B}$ engage in communication with each other over a public network (e.g., the Internet): a concrete example could be where they represent a web-browser and web-server respectively. Since the n_b-bit messages they communicate will potentially contain security-critical (e.g., identity-, location-, medical-, or finance-related) information, it is important to prevent a third-party $\mathcal{E}$ having access to them. Assuming $\mathcal{A}$ and $\mathcal{B}$ have agreed on some n_k-bit key k before they start communicating, having them use a block cipher[1] to encrypt and decrypt messages would be one approach to satisfying their requirement for secrecy. The idea is that $\mathcal{A}$ encrypts a plaintext message m to form the ciphertext message $c = \textsc{Enc}(k, m)$ which is sent to $\mathcal{B}$. Then, $\mathcal{B}$ decrypts the ciphertext message by computing $m' = \textsc{Dec}(k, c)$ and thereby recovers the same plaintext message, i.e., $m' = m$. This approach is effective because Enc and Dec are carefully designed so that 1) they act as each others inverse under k, and 2) security depends on k alone, not on knowledge of Enc and Dec: even if an attacker $\mathcal{E}$ intercepts c, they cannot easily recover m without also knowing k.

A given block cipher design specifies the algorithms Enc and Dec and associated parameters, e.g., n_k and n_b; numerous such designs exist, the de facto choice being the Advanced Encryption Standard (AES) [1]. PRESENT [2] is a block cipher design which can represents attractive alternative to AES: it is specifically designed to allow for a compact (i.e., low area overhead) hardware implementation, which can be an important metric in constrained use-cases.

2 Terms and conditions

- The assignment description may refer to the ASCII text file `question.txt`, or more generally "the marksheet": complete and include this file in your submission. This is important, in the sense that 1) it offers *you* clarity with respect to the marking process, e.g., via a marking scheme, and 2) it offers *us* useful (meta-)information about your submission. Keep in mind that *if* separate *assessment* units exist, they may have different assessment criteria and so marking scheme.
- Certain aspects of the assignment have a (potentially large) design space of possible approaches. Where there is some debate about the correct or "best" approach, the assignment demands *you* make an informed decision *yourself*: it is therefore not (purely) a programming exercise such that blindly implementing *an* approach will be enough. Such decisions should ideally be based on a reasoned argument formed via your *own* background research (versus relying exclusively on the teaching material provided), and clearly documented (e.g., using the marksheet).
- The assignment design includes some heavily supported, closed initial stages which reflect a lower mark, and some mostly unsupported, open later stages which reflects a higher mark. This suggests the marking scale is non-linear: it is clearly easier to obtain X marks in the initial stages than in the final stage. The term open (resp. closed) should be understood as meaning flexibility with respect to options for work, *not* non-specificity with respect to workload: each stage has a clear success criteria that limit the functionality you implement, meaning you can (and should) stop work once they have been satisfied.
- In some, specific instances the required style of Verilog will be dictated by the assignment. If no such requirement exists, however, *you* can select whatever style is appropriate: gate-, RTL-, and behavioural-level Verilog styles are all viable in general. However, whatever style you select, you must consider how your solution relates to real hardware versus purely whether or not it functions correctly in simulation.
- You should submit your work into the correct component via

  `https://www.ole.bris.ac.uk`

  Include any a) source code files, b) text or PDF files, (e.g., documentation) and c) auxiliary files (e.g., example output), either as required or that *you* feel are relevant. Keep in mind that *if* separate *teaching* and *assessment* units exist, you should submit via the latter *not* the former.
- To make the submission process easier, the recommended approach is to develop your solution within the *same* directory structure as the material provided. This will allow you to first create then submit a *single* archive (e.g., `solution.zip` using `zip`, or `solution.tar.gz` using `tar` and `gzip`) of your entire solution, rather than *multiple* separate files.
- Any implementations produced as part of the assignment will be marked using a platform equivalent to the MVB Linux lab(s). (e.g., MVB-1.15 or MVB-2.11). As such, they *must* compile, execute, and be thoroughly tested using both the operating system and development tool-chain versions available by default.

[1] `https://en.wikipedia.org/wiki/Block_cipher`

```
${ARCHIVE}
  ├── question.txt
  ├── Makefile
  ├── params.h
  ├── vectors_k.txt
  ├── vectors_m.txt
  ├── vectors_c.txt
  ├── encrypt_v1.v
  ├── encrypt_v1_test.v
  ├── encrypt_v2.v
  ├── encrypt_v2_test.v
  ├── encrypt_v3.v
  ├── encrypt_v3_test.v
  ├── sbox.v
  ├── sbox_test.v
  ├── key_schedule.v
  ├── key_schedule_test.v
  ├── round.v
  ├── round_test.v
  └── util.v
```

Figure 1: *A diagrammatic description of the material in* `question.tar.gz`*.*

- Although you can *definitely* expect to receive partial marks for a partial solution, it will be marked *as is*. This means a) there will be no effort to enable either optional or commented functionality (e.g., by uncommenting it, or via specification of compile-time or run-time parameters), and b) submitting multiple variant solutions is strongly discouraged, but would be dealt with by considering the variant which yields the highest single mark.

3 Description

3.1 Material

Download and unarchive the file

https://assets.phoo.org/COMS10015_2024_TB-4/csdsp/cw/Encrypt/question.tar.gz

somewhere secure in your file system: from here on, we assume `${ARCHIVE}` denotes a path to the resulting, unarchived content illustrated by Figure 1. In particular, you should find

- `question.txt`†, the marksheet mentioned in the assessment terms and conditions,
- `Makefile`, a GNU `make` based build system described in more detail by Appendix C.
- `params.h`, a header file that provides a symbolic definition of parameters such as n_k, n_b, and n_r,
- `vectors_k.txt`, `vectors_m.txt` and `vectors_c.txt`, ASCII text files representing the test vectors described in more detail by Appendix D.
- `encrypt_v1.v`†, `encrypt_v2.v`†, and `encrypt_v3.v`†, incomplete implementations of modules called `encrypt_v1`, `encrypt_v2`, and `encrypt_v3`.
- `sbox.v`†, an incomplete implementation of a module called `sbox`.
- `key_schedule.v`†, an incomplete implementation of a module called `key_schedule`.
- `round.v`†, an incomplete implementation of a module called `round`.
- `util.v`, a set of pre-defined support modules including
  - `split_0`, a module that splits a single input into multiple outputs,
  - `merge_0`, a module that merges multiple inputs into a single output,
  - `key_addition`, an implementation of the PRESENT key addition operation,
  - `perm`, an implementation of the PRESENT permutation operation.

plus a set of test stimuli: for a module `X` as defined in `X.v`, the corresponding test stimulus is `X_test` as defined in `X_test.v`. Viewed at face value, that is a *lot* of files! However, it is vital to understand that you can complete the assignment by altering *only* those files marked with a † symbol. Because you do not need to, you should not alter any *other* files: if you *do*, those alterations will be reverted before (and so therefore ignored during) the marking process.

3.2 Overview

Consider an example scenario, where you join the development team for a device $\mathcal{T}$; to address the challenge of secure communication, $\mathcal{T}$ integrates and makes use of a hardware implementation of PRESENT. This assignment models aspects of the scenario outlined above, using Verilog as a vehicle to do so. More specifically, it tasks you with implementing the Enc algorithm for PRESENT in Verilog. Remember that, in essence, Verilog simply offers a neat way to express and experiment with (i.e., simulate) a design you could also write down on paper and reason about in theory: a sensible strategy throughout is to establish understanding "on paper" before then applying it "in practice" (e.g., via source code).

Selection of PRESENT[2] implies $n_k = 80$ and $n_b = 64$, meaning a 80-bit cipher key k is used to encrypt a 64-bit plaintext message m into a 64-bit ciphertext message c. PRESENT is an iterative block cipher, meaning that a full encryption operation involves successively applying partial encryption rounds (or steps): Figure 2 illustrates this using a block diagram, noting that a total of $n_r = 31$ rounds (numbered 1 to 31 inclusive) are followed by a post-processing step.

3.3 Detail

Stage 1. The goal of this stage is to implement modules that, when combined, act to support some i-th round of encryption. The idea is that by using these modules, the subsequent stages can then explore different implementation strategies for a full encryption operation.

(a) PRESENT uses a single S-box[3], which could be described as mapping a 4-bit input `x` to (resp. for) a 4-bit output `r`. The `sbox` module implements this S-box: the required implementation is described in a tabular form by Figure 3 , which is basically a compact truth table for the Boolean function

$$\texttt{sbox} : \{0,1\}^4 \rightarrow \{0,1\}^4.$$

Complete the module implementation, using Appendix D to verify (as far as possible) that it functions as expected: your implementation should express the S-box as a (set of) Boolean expressions, e.g., using a gate- or RTL-level Verilog style.

(b) With reference to Figure 2, the `key_schedule` module implements some i-th round of the key schedule. The required implementation is described in a diagrammatic form by Figure 4, although arguably even more clearly using words alone: the output `r` can be viewed as being produced by

- taking `x` and left-rotating it by a distance of 61 bits, then
- taking that intermediate value, say `t0`, and updating bits 76 to 79 only (i.e., leaving all others unaltered) using an instance of the `sbox` module (i.e., passing those bits through the S-box), then
- taking that intermediate value, say `t1`, and updating bits 15 to 19 only (i.e., leaving all others unaltered) by XOR'ing them with `i`.

Complete the module implementation, using Appendix D to verify (as far as possible) that it functions as expected.

(c) With reference to Figure 2, the `round` module implements some i-th round of encryption: the required implementation is described in a diagrammatic form by Figure 5.

Complete the module implementation, using Appendix D to verify (as far as possible) that it functions as expected.

Stage 2. The `encrypt_v1` module accepts

- a 80-bit value called `k`, a cipher key, and
- a 64-bit value called `m`, a plaintext message,

as input, and produces

- a 64-bit value called `c`, a ciphertext message,

as output: as such, it computes a full encryption operation.

The goal of this stage is to implement the `encrypt_v1` module, using a combinatorial design strategy which replicates Figure 2 in a fairly direct manner. Adopting this strategy trades-off higher area (since n_r rounds are instantiated) in favour of lower latency (since the number of clock cycles for each encryption is 1), in relative terms, and makes the resulting implementation easier to use: the output is computed continuously from the input.

Complete the module implementation, using Appendix D to verify (as far as possible) that it functions as expected.

[2] An accessible introduction to PRESENT is provided by https://en.wikipedia.org/wiki/PRESENT for example. Keep in mind, however, that various technical details relating to PRESENT are out of scope given the task at hand: where that is the case, we simply ignore them. For example, note that PRESENT represents a specific instance of a more general block cipher design principle, i.e., a Substitution-Permutation (SP) network. Also note that PRESENT actually permits either a 80- or 128-bit cipher key to be used; we focus on the former only.

[3] https://en.wikipedia.org/wiki/S-box

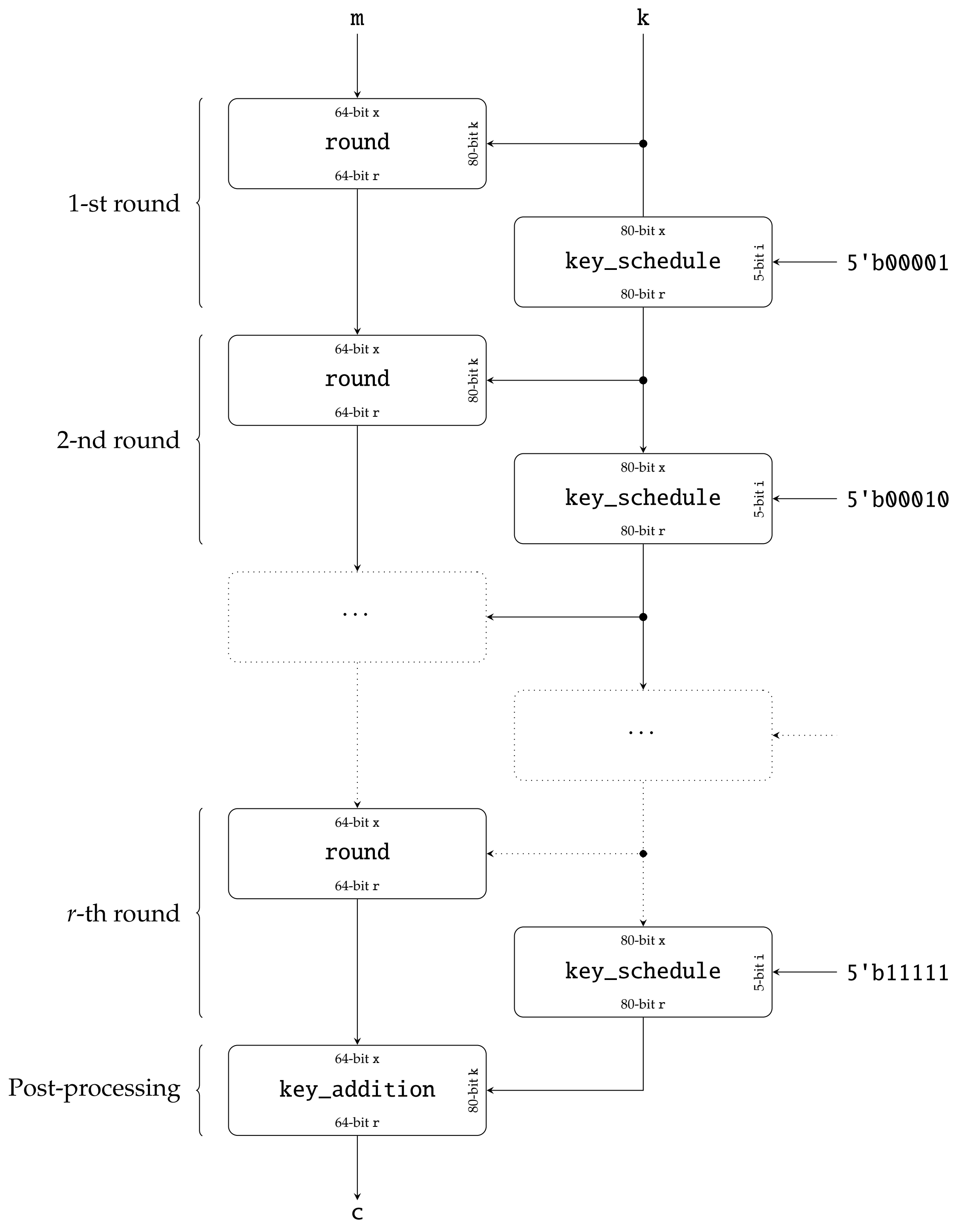


Figure 2: *A block diagram illustrating a full encryption operation using PRESENT, i.e., computation of* $c = \textsc{Enc}(k, m)$.

x	r	x	r
$0_{(16)}$	$C_{(16)}$	$8_{(16)}$	$3_{(16)}$
$1_{(16)}$	$5_{(16)}$	$9_{(16)}$	$E_{(16)}$
$2_{(16)}$	$6_{(16)}$	$A_{(16)}$	$F_{(16)}$
$3_{(16)}$	$B_{(16)}$	$B_{(16)}$	$8_{(16)}$
$4_{(16)}$	$9_{(16)}$	$C_{(16)}$	$4_{(16)}$
$5_{(16)}$	$0_{(16)}$	$D_{(16)}$	$7_{(16)}$
$6_{(16)}$	$A_{(16)}$	$E_{(16)}$	$1_{(16)}$
$7_{(16)}$	$D_{(16)}$	$F_{(16)}$	$2_{(16)}$

Figure 3: *The* sbox *module, as used to form the i-th round of PRESENT.*

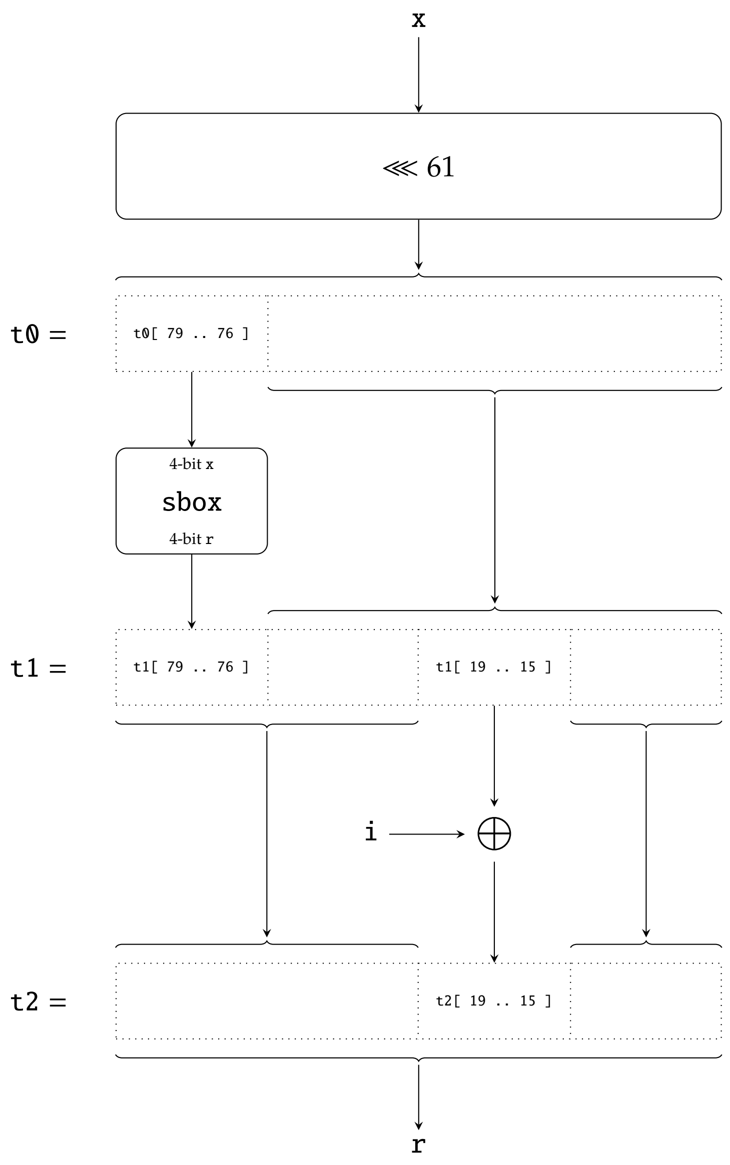


Figure 4: *The* `key_schedule` *module, as used to form the i-th round of PRESENT (note that ⋘ denotes left-rotate, in this case by a fixed distance of* 61 *bits).*

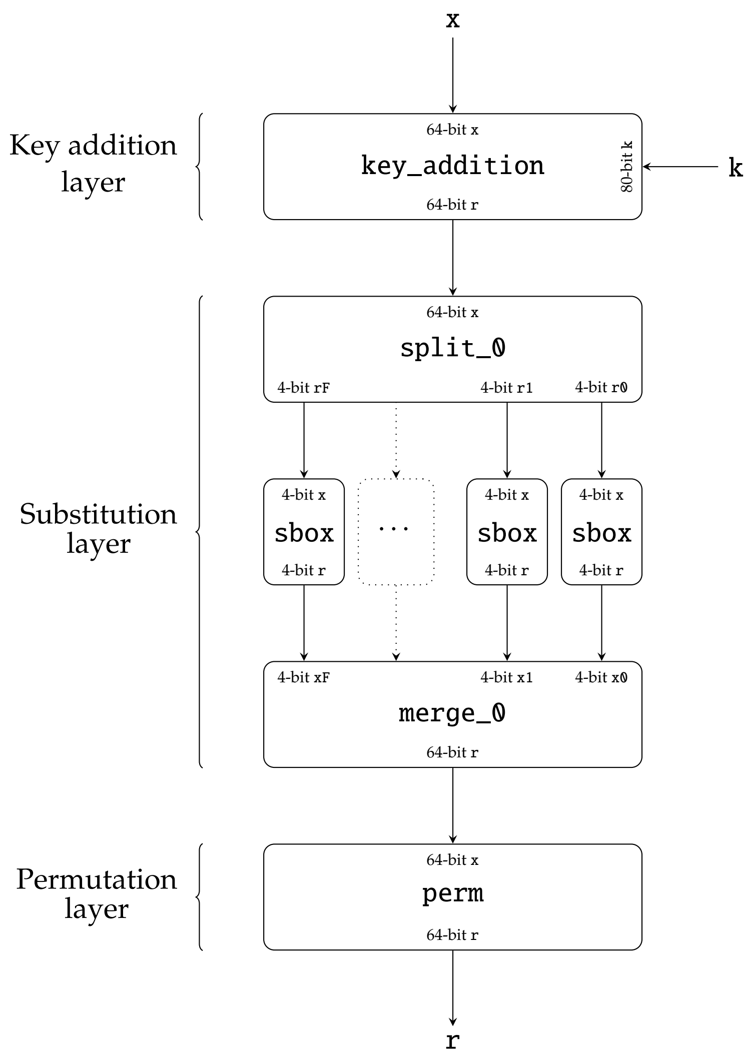


Figure 5: *The* `round` *module, as used to form the i-th round of PRESENT.*

Stage 3. The `encrypt_v2` module has an interface matching that of `encrypt_v1` except for some additional inputs and outputs, namely

- a 1-bit value called `clk`, a clock signal,
- a 1-bit value called `req`, a request signal, and
- a 1-bit value called `ack`, an acknowledge signal,

and also computes a full encryption operation.

The goal of this stage is to implement the `encrypt_v2` module, using a different, iterative design strategy. Adopting this strategy trades-off higher latency (since the number of clock cycles for each encryption is $\sim n_r$), in favour of lower area (since 1 round is instantiated), in relative terms, and makes the resulting implementation harder to use: a control protocol outlined by Appendix A must be followed to provide input and accept output correctly.

Complete the module implementation, using Appendix D to verify (as far as possible) that it functions as expected.

Advice. The natural way to design and implement the control protocol is by treating it as a Finite State Machine (FSM). More so than other stages, it is important to explain the design of said FSM: use either `question.txt` and/or comments in your source code to do so.

Advice. Until you implement the module, simulating it will "hang" as a result of incorrect interaction with the test stimulus. In more detail, the test stimulus follows the control protocol and hence waits for the module to set `ack` to 1 (resp. 0) at the start (resp. end) of computation: `ack` is not updated by the incomplete implementation, so the test stimulus ends up waiting forever.

Stage 4. The `encrypt_v3` module has an interface matching that of `encrypt_v1` except for some additional inputs and outputs, namely

- a 1-bit value called `clk`, a clock signal,

and also computes a full encryption operation.

The goal of this stage is to implement the `encrypt_v3` module, using a different, pipelined design strategy which delivers a different trade-off: this strategy focuses on maximising throughput, rather than minimising latency (as with `encrypt_v1`) or area (as with `encrypt_v2`). Note that a control protocol outlined by Appendix B must be followed to provide input and accept output correctly.

Complete the module implementation, using Appendix D to verify (as far as possible) that it functions as expected.

Advice. Before investing significant effort in design or implementation tasks, it is crucial you first conduct some background research in order to understand the concept of pipelining.

Advice. In general, the number of pipeline stages represents a parameter for which a concrete value must be selected. Here, however, the assumption is that a fully pipelined design strategy is employed. This implies 1) there are n_r pipeline stages, so 2) computation of an output from the associate inputs has an n_r clock cycle latency.

References

[1] *Advanced Encryption Standard (AES)*. National Institute of Standards and Technology (NIST) Federal Information Processing Standard (FIPS) 197. 2001. URL: https://doi.org/10.6028/NIST.FIPS.197 (see p. 2).

[2] A. Bogdanov et al. "PRESENT: An Ultra-Lightweight Block Cipher". In: *Cryptographic Hardware and Embedded Systems (CHES)*. LNCS 4727. Springer-Verlag, 2007, pp. 450–466 (see p. 2).

A Control protocol for the iterative design strategy

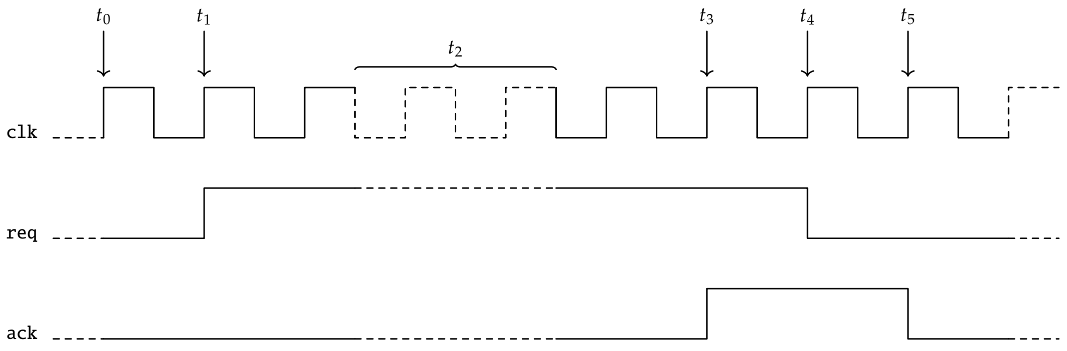
A.1 Problem

The iterative design strategy requires an understanding of how the module (i.e., `encrypt_v2`) and the user *of* said module (e.g., the test stimulus `encrypt_v2_test`) interact. Two underlying problems exist, namely 1) the module does not know know when to start computation (i.e., when input is available), and 2) the user does not know know when computation has finished (i.e., when output is available).

The solution of both problems is for both parties to follow a protocol: this is based on use of the request signal `req` and acknowledge signal `ack`, and driven by (positive edges of) the clock signal `clk`. You could frame interaction between the module and user as communication between (i.e., of input and output to and from) them, and so a "conversation" controlled (or structured) by the protocol: the rules of said protocol essentially mean, e.g., one party cannot be "confused" as a result of communication by the other.

A.2 Protocol

Based on a diagrammatic description



the (intentionally simple) procotol can be explained as follows:

- At some positive edge on `clk` (labelled t_0), both `req` and `ack` are initially 0.
- At some positive edge on `clk` (labelled t_1), the user wants the module start a computation. It proceeds by 1) driving values onto any inputs (i.e., `k` and `m`), then 2) changing `req` from 0 to 1.
- The module notices the change to (e.g., positive edge on) `req`, and concludes that the inputs are available. Note that, in general, it would need to store the inputs ready for subsequent use. The reason for storing them internally within the module stems from a need to be pessimistic: the module must pessimistically assume any externally provided input *may* be changed *during* computation which uses them. However, given the assignment remit, we relax the requirement to do so by guaranteeing all inputs are stable throughout the computation (i.e., they will not change until the computation is complete, so there is no need to store them internally).
- During some period (labelled t_2), the module computes the outputs from the inputs using `clk` to trigger each constituent step; during this period, the user is essentially waiting for the computation to finish.
- At some positive edge on `clk` (labelled t_3), the module finishes the computation. It proceeds by 1) driving values onto any outputs (i.e., `c`), then 2) changing `ack` from 0 to 1.
- The user notices the change to (e.g., positive edge on) `ack`, and concludes that the outputs are available. It proceeds by 1) storing any outputs ready for subsequent use, then 2) changing `req` from 1 to 0.
- The module notices the change to (e.g., negative edge on) `req`, and concludes that the interaction is finished. It proceeds by changing `ack` from 1 to 0.
- The user notices the change to (e.g., negative edge on) `ack` and concludes that the interaction is finished.
- Since both `req` and `ack` are 0 again, the module and user are ready to engage in successive interactions if/when need be.

B Control protocol for the pipelined design strategy

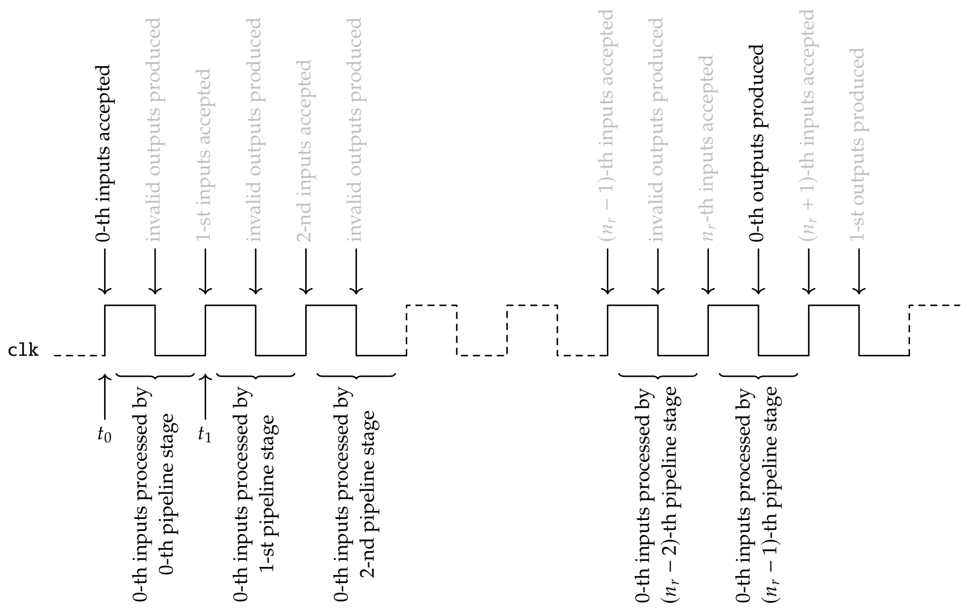
B.1 Problem

In the same way as described in Chapter A, the pipelined design strategy requires careful control (or structure) with respect to interaction between the module and user; this is realised by having them adhere to a protocol.

Similar underlying problems exist, but they now stem from the fact that, at a given instant, a pipeline with n_r stages could be described as simultaneously processing n_r independent computations, each of which is at a different stage of completeness i.e., less (resp. more) complete within initial (resp. latter) stages. This description implies a need to control what the pipeline does, and when it does it, that that, e.g., inputs and outputs are accepted and produced correctly, and computations of one from the other progresses correctly.

B.2 Protocol

Based on a diagrammatic description


0-th inputs accepted
invalid outputs produced
1-st inputs accepted
invalid outputs produced
2-nd inputs accepted
invalid outputs produced
$(n_r - 1)$-th inputs accepted
invalid outputs produced
n_r-th inputs accepted
0-th outputs produced
$(n_r + 1)$-th inputs accepted
1-st outputs produced
clk
t_0
t_1
0-th inputs processed by 0-th pipeline stage
0-th inputs processed by 1-st pipeline stage
0-th inputs processed by 2-nd pipeline stage
0-th inputs processed by $(n_r - 2)$-th pipeline stage
0-th inputs processed by $(n_r - 1)$-th pipeline stage

the (intentionally simple) procotol can be explained as follows:

- at every positive edge on `clk`, the module accepts inputs (i.e., `k` and `m`) from the user and
- at every negative edge on `clk`, the user accepts outputs (i.e., `c`) from the module.

Despite being (or perhaps because it is) so simple, several subtle but important points need to be considered: these points are reflected by the test stimulus, but warrant discussion and explanation. First, the outputs associated with specific inputs will be spaced n_r clock cycles apart; this fact stems from the number of stages in and hence computational latency of the pipeline. Second, the outputs will be invalid in some clock cycles; this fact stems from their not corresponding to any associated inputs. For example, the 0-th inputs are provided to the pipeline on the clock edge labelled t_0. On the clock edge labelled t_1, those inputs have only been processed by 1 or n_r stages: this means the associated outputs are not ready, and so what the pipeline produces as output is therefore is invalid (since *no* inputs have been fully processed) at that instant. Third, `clk` is now the *only* form of synchronisation between the module and user: in contrast to Chapter A, for example, `req` and `ack` no longer exist. This places stricter limits on the module and user, in the sense that the inputs (resp. outputs) needs to be ready at each positive (resp. negative) edge on `clk`, so that the module (resp. user) can store and then use them correctly; they cannot simply wait if the inputs (resp. outputs) are *not* ready, for example.

C Developing your solution

C.1 Workflow

To use the content provided, and thus support development of your solution based on it, you can and so should adopt the same approach as presented in the lab. worksheet(s).

Before you start:

1. update your `${PATH}`[4] environment variable by executing

   ```
   export PATH="${PATH}:/opt/iverilog/12.0/bin"
   ```

2. check said update worked correctly by executing

   ```
   which iverilog
   which gtkwave
   ```

   noting that any reported error (e.g., `no iverilog in ...` or similar) suggests it did not: ask for help!

Imagine you have 1) `X.v`, which implements a module `X`, plus 2) `X_test.v`, which implements a module `X_test`, where the latter acts as a test stimulus for the former. Although all of the steps related to use of `X.v` and `X_test.v` could be performed manually, the `Makefile` provided represents an automated build system which implies less effort *and* less chance of error. Based on the use of `Makefile`, an edit-compile-execute style design cycle can be roughly summerised as follows:

1. Edit the Verilog source code file `X.v`.
2. Execute

   ```
   make clean
   ```

   to clean (i.e., remove) residual files stemming from previous compilation or simulation steps.
3. Execute

   ```
   make X_test.vvp
   ```

   to compile the design using `iverilog`: doing so combines the Verilog source code file `X.v` with the associated test stimulus `X_test.v` to produce the executable `X_test.vvp`.
4. Execute

   ```
   make X_test.vcd
   ```

   to simulate the design using `vvp`: doing so produces 1) some machine-readable output via a Value Change Dump (VCD)[5] file `X_test.vcd`, plus 2) some human-readable output via the terminal.

   Not all test stimuli are fully-automatic; some require arguments (or parameters) to control them. In such a case, `Makefile` uses the `ARGS` environment variable to capture arguments it then passes to `vvp`. For example, imagine `X_test` requires three 1-bit arguments called `x`, `y`, and `z`: instead of the above, one could instead execute

   ```
   make ARGS="+x=0 +y=1 +z=0" X_test.vcd
   ```

   to set `x` = 0, `y` = 1, and `z` = 0.
5. Execute

   ```
   gtkwave X_test.vcd
   ```

   to visualise the resulting VCD file using `gtkwave`.

Once you have completed your solution, execute

```
make solution.tar.gz
```

to automatically create a single archive, i.e., the file, `solution.tar.gz`, consisting of *all* files in the current working directory.

[4] http://en.wikipedia.org/wiki/PATH_(variable)
[5] https://en.wikipedia.org/wiki/Value_change_dump

C.2 Example

In more concrete terms, one might consider the case where `X = encrypt_v1` which means that the Verilog source code files `encrypt_v1.v` and `encrypt_v1_test.v` describe the modules `encrypt_v1`, and `encrypt_v1_test` respectively. The development workflow would then be

1. edit `encrypt_v1.v` to complete the module implementation,
2. execute `make clean`,
3. execute `make encrypt_v1_test.vvp`,
4. execute `make encrypt_v1_test.vcd`,
5. execute `gtkwave encrypt_v1_test.vcd`.

D Testing your solution

D.1 Test vectors

The process of testing and debugging an implementation of some arithmetic operation (e.g., a ripple-carry adder) is arguably made easier by the fact we can (and sometimes do) compute the same results manually; this fact affords some intuition about whether the correct result is produced, and, crucially, the reason why if not. The same is not true of a block cipher implementation, where, by design, there is no analogous intuition for what the correct result should be (in the sense it should "look" random).

This problem demands a considered approach to testing and debugging. A number of different options exist, for example:

1. Given an implementation of only Enc, we can test whether $c \overset{?}{=} \textsc{Enc}(k, m)$ if provided with a test vector comprising k and m plus the corresponding, known to be correct c.
2. Given an implementation of both Enc and Dec, we can apply a consistency check by testing whether $m \overset{?}{=} \textsc{Dec}(k, \textsc{Enc}(k, m))$ for some random k and m.

Each has advantages and disadvantages, and one might sensibly argue that a combination of these (and others) would be ideal. For this assignment however, the later is more attractive in the sense it requires less work by you. As such, the test stimuli provided rely on a set of test vectors: each i-th line of the ASCII text file

- `vectors_k.txt` contains $k[i]$, the i-th 80-bit hexadecimal cipher key,
- `vectors_m.txt` contains $m[i]$, the i-th 64-bit hexadecimal plaintext message, and
- `vectors_c.txt` contains $c[i]$, the i-th 64-bit hexadecimal ciphertext message,

such that $c[i] = \textsc{Enc}(k[i], m[i])$. For example, the first lines contain

$$\begin{array}{rcrcr} k[0] & = & DC8770E93EA141E1FC67_{(16)} & \mapsto & \texttt{80'hDC8770E93EA141E1FC67} \\ m[0] & = & 54110E827441213D_{(16)} & \mapsto & \texttt{64'h54110E827441213D} \\ c[0] & = & 02DEBC8CB87BC942_{(16)} & \mapsto & \texttt{64'h02DEBC8CB87BC942} \end{array}$$

A given test stimuli loads this content into a corresponding memory using the `readmemh` system task, then tests whether $c[i] \overset{?}{=} \textsc{Enc}(k[i], m[i])$ for each i-th test vector. This allows the test process to be automatic, and avoids the test stimuli itself becoming too verbose (e.g., due to expression of the test vectors as Verilog source code).

D.2 Fully worked example

Although the test vectors described above are useful for testing whether an *overall* result is correct, when said result is *in*correct they offer little or no insight into what or where the problem might be. To help address this limitation, the following represents a fully worked example for test vector $i = 0$. Note that it includes all intermediate inputs and outputs for every operation within every round: although this makes it extremely verbose, it should, e.g., allow you identify exactly where your implementation and the example differ.

Input

```
k   =   80'hdc8770e93ea141e1fc67
m   =   64'h54110e827441213d
```

1-st round

- `key_schedule`

```
x   =   80'hdc8770e93ea141e1fc67
i   =   5'h01
r   =   80'hbf8cfb90ee1d27d4a83c
```

- `round`

```
x   =   64'h54110e827441213d
k   =   80'hdc8770e93ea141e1fc67
r   =   64'h3bdc2857f24acce2
```

- key_addition

```
x    =   64'h54110e827441213d
k    =   80'hdc8770e93ea141e1fc67
r    =   64'h88967e6b4ae060dc
```

- split_0

```
x    =   64'h88967e6b4ae060dc
r0   =   4'hc
r1   =   4'hd
r2   =   4'h0
r3   =   4'h6
r4   =   4'h0
r5   =   4'he
r6   =   4'ha
r7   =   4'h4
r8   =   4'hb
r9   =   4'h6
rA   =   4'he
rB   =   4'h7
rC   =   4'h6
rD   =   4'h9
rE   =   4'h8
rF   =   4'h8
```

- sbox (0-th instance)

```
x    =   4'hc
r    =   4'h4
```

- sbox (1-st instance)

```
x    =   4'hd
r    =   4'h7
```

- sbox (2-nd instance)

```
x    =   4'h0
r    =   4'hc
```

- sbox (3-rd instance)

```
x    =   4'h6
r    =   4'ha
```

- sbox (4-th instance)

```
x    =   4'h0
r    =   4'hc
```

- sbox (5-th instance)

```
x    =   4'he
r    =   4'h1
```

- sbox (6-th instance)

```
x    =   4'ha
r    =   4'hf
```

- sbox (7-th instance)

```
x    =   4'h4
r    =   4'h9
```

- sbox (8-th instance)

```
x    =   4'hb
r    =   4'h8
```

- sbox (9-th instance)

```
x    =   4'h6
r    =   4'ha
```

- sbox (10-th instance)

```
x    =   4'he
r    =   4'h1
```

- sbox (11-st instance)

```
x    =   4'h7
r    =   4'hd
```

- sbox (12-nd instance)

```
x     =    4'h6
r     =    4'ha
```

- sbox (13-rd instance)

```
x     =    4'h9
r     =    4'he
```

- sbox (14-th instance)

```
x     =    4'h8
r     =    4'h3
```

- sbox (15-th instance)

```
x     =    4'h8
r     =    4'h3
```

- merge_0

```
x0    =    4'h4
x1    =    4'h7
x2    =    4'hc
x3    =    4'ha
x4    =    4'hc
x5    =    4'h1
x6    =    4'hf
x7    =    4'h9
x8    =    4'h8
x9    =    4'ha
xA    =    4'h1
xB    =    4'hd
xC    =    4'ha
xD    =    4'he
xE    =    4'h3
xF    =    4'h3
r     =    64'h33ead1a89f1cac74
```

- perm

```
x     =    64'h33ead1a89f1cac74
r     =    64'h3bdc2857f24acce2
```

2-nd round

- key_schedule

```
x     =    80'hbf8cfb90ee1d27d4a83c
i     =    5'h02
r     =    80'he50797f19f721dc2a4fa
```

- round

```
x     =    64'h3bdc2857f24acce2
k     =    80'hbf8cfb90ee1d27d4a83c
r     =    64'h55171bd08c03cd9a
```

  - key_addition

```
x     =    64'h3bdc2857f24acce2
k     =    80'hbf8cfb90ee1d27d4a83c
r     =    64'h8450d3c71c57eb36
```

  - split_0

```
x     =   64'h8450d3c71c57eb36
r0    =   4'h6
r1    =   4'h3
r2    =   4'hb
r3    =   4'he
r4    =   4'h7
r5    =   4'h5
r6    =   4'hc
r7    =   4'h1
r8    =   4'h7
r9    =   4'hc
rA    =   4'h3
rB    =   4'hd
rC    =   4'h0
rD    =   4'h5
rE    =   4'h4
rF    =   4'h8
```

- sbox (0-th instance)
```
x     =   4'h6
r     =   4'ha
```
- sbox (1-st instance)
```
x     =   4'h3
r     =   4'hb
```
- sbox (2-nd instance)
```
x     =   4'hb
r     =   4'h8
```
- sbox (3-rd instance)
```
x     =   4'he
r     =   4'h1
```
- sbox (4-th instance)
```
x     =   4'h7
r     =   4'hd
```
- sbox (5-th instance)
```
x     =   4'h5
r     =   4'h0
```
- sbox (6-th instance)
```
x     =   4'hc
r     =   4'h4
```
- sbox (7-th instance)
```
x     =   4'h1
r     =   4'h5
```
- sbox (8-th instance)
```
x     =   4'h7
r     =   4'hd
```
- sbox (9-th instance)
```
x     =   4'hc
r     =   4'h4
```
- sbox (10-th instance)
```
x     =   4'h3
r     =   4'hb
```
- sbox (11-st instance)
```
x     =   4'hd
r     =   4'h7
```
- sbox (12-nd instance)
```
x     =   4'h0
r     =   4'hc
```

- sbox (13-rd instance)

```
x    =   4'h5
r    =   4'h0
```

- sbox (14-th instance)

```
x    =   4'h4
r    =   4'h9
```

- sbox (15-th instance)

```
x    =   4'h8
r    =   4'h3
```

- merge_0

```
x0   =   4'ha
x1   =   4'hb
x2   =   4'h8
x3   =   4'h1
x4   =   4'hd
x5   =   4'h0
x6   =   4'h4
x7   =   4'h5
x8   =   4'hd
x9   =   4'h4
xA   =   4'hb
xB   =   4'h7
xC   =   4'hc
xD   =   4'h0
xE   =   4'h9
xF   =   4'h3
r    =   64'h390c7b4d540d18ba
```

- perm

```
x    =   64'h390c7b4d540d18ba
r    =   64'h55171bd08c03cd9a
```

3-rd round

• key_schedule

```
x    =   80'he50797f19f721dc2a4fa
i    =   5'h03
r    =   80'h049f5ca0f2fe33efc3b8
```

• round

```
x    =   64'h55171bd08c03cd9a
k    =   80'he50797f19f721dc2a4fa
r    =   64'hd06477bc0a4929f9
```

- key_addition

```
x    =   64'h55171bd08c03cd9a
k    =   80'he50797f19f721dc2a4fa
r    =   64'hb0108c211371d058
```

- split_0

```
x     =   64'hb0108c211371d058
r0    =   4'h8
r1    =   4'h5
r2    =   4'h0
r3    =   4'hd
r4    =   4'h1
r5    =   4'h7
r6    =   4'h3
r7    =   4'h1
r8    =   4'h1
r9    =   4'h2
rA    =   4'hc
rB    =   4'h8
rC    =   4'h0
rD    =   4'h1
rE    =   4'h0
rF    =   4'hb
```

– sbox (0-th instance)

```
x     =   4'h8
r     =   4'h3
```

– sbox (1-st instance)

```
x     =   4'h5
r     =   4'h0
```

– sbox (2-nd instance)

```
x     =   4'h0
r     =   4'hc
```

– sbox (3-rd instance)

```
x     =   4'hd
r     =   4'h7
```

– sbox (4-th instance)

```
x     =   4'h1
r     =   4'h5
```

– sbox (5-th instance)

```
x     =   4'h7
r     =   4'hd
```

– sbox (6-th instance)

```
x     =   4'h3
r     =   4'hb
```

– sbox (7-th instance)

```
x     =   4'h1
r     =   4'h5
```

– sbox (8-th instance)

```
x     =   4'h1
r     =   4'h5
```

– sbox (9-th instance)

```
x     =   4'h2
r     =   4'h6
```

– sbox (10-th instance)

```
x     =   4'hc
r     =   4'h4
```

– sbox (11-st instance)

```
x     =   4'h8
r     =   4'h3
```

– sbox (12-nd instance)

```
x     =   4'h0
r     =   4'hc
```

- sbox (13-rd instance)

```
x    =   4'h1
r    =   4'h5
```

- sbox (14-th instance)

```
x    =   4'h0
r    =   4'hc
```

- sbox (15-th instance)

```
x    =   4'hb
r    =   4'h8
```

- merge_0

```
x0   =   4'h3
x1   =   4'h0
x2   =   4'hc
x3   =   4'h7
x4   =   4'h5
x5   =   4'hd
x6   =   4'hb
x7   =   4'h5
x8   =   4'h5
x9   =   4'h6
xA   =   4'h4
xB   =   4'h3
xC   =   4'hc
xD   =   4'h5
xE   =   4'hc
xF   =   4'h8
r    =   64'h8c5c34655bd57c03
```

- perm

```
x    =   64'h8c5c34655bd57c03
r    =   64'hd06477bc0a4929f9
```

4-th round

- key_schedule

```
x    =   80'h049f5ca0f2fe33efc3b8
i    =   5'h04
r    =   80'h28770093eb941e5dc67d
```

- round

```
x    =   64'hd06477bc0a4929f9
k    =   80'h049f5ca0f2fe33efc3b8
r    =   64'h54358b1ea8c5c25e
```

  - key_addition

```
x    =   64'hd06477bc0a4929f9
k    =   80'h049f5ca0f2fe33efc3b8
r    =   64'hd4fb2b1cf8b71a16
```

  - split_0

```
x    =   64'hd4fb2b1cf8b71a16
r0   =   4'h6
r1   =   4'h1
r2   =   4'ha
r3   =   4'h1
r4   =   4'h7
r5   =   4'hb
r6   =   4'h8
r7   =   4'hf
r8   =   4'hc
r9   =   4'h1
rA   =   4'hb
rB   =   4'h2
rC   =   4'hb
rD   =   4'hf
rE   =   4'h4
rF   =   4'hd
```

– sbox (0-th instance)

```
x    =   4'h6
r    =   4'ha
```

– sbox (1-st instance)

```
x    =   4'h1
r    =   4'h5
```

– sbox (2-nd instance)

```
x    =   4'ha
r    =   4'hf
```

– sbox (3-rd instance)

```
x    =   4'h1
r    =   4'h5
```

– sbox (4-th instance)

```
x    =   4'h7
r    =   4'hd
```

– sbox (5-th instance)

```
x    =   4'hb
r    =   4'h8
```

– sbox (6-th instance)

```
x    =   4'h8
r    =   4'h3
```

– sbox (7-th instance)

```
x    =   4'hf
r    =   4'h2
```

– sbox (8-th instance)

```
x    =   4'hc
r    =   4'h4
```

– sbox (9-th instance)

```
x    =   4'h1
r    =   4'h5
```

– sbox (10-th instance)

```
x    =   4'hb
r    =   4'h8
```

– sbox (11-st instance)

```
x    =   4'h2
r    =   4'h6
```

– sbox (12-nd instance)

```
x    =   4'hb
r    =   4'h8
```

- sbox (13-rd instance)

```
x    =   4'hf
r    =   4'h2
```

- sbox (14-th instance)

```
x    =   4'h4
r    =   4'h9
```

- sbox (15-th instance)

```
x    =   4'hd
r    =   4'h7
```

- merge_0

```
x0   =   4'ha
x1   =   4'h5
x2   =   4'hf
x3   =   4'h5
x4   =   4'hd
x5   =   4'h8
x6   =   4'h3
x7   =   4'h2
x8   =   4'h4
x9   =   4'h5
xA   =   4'h8
xB   =   4'h6
xC   =   4'h8
xD   =   4'h2
xE   =   4'h9
xF   =   4'h7
r    =   64'h79286854238d5f5a
```

- perm

```
x    =   64'h79286854238d5f5a
r    =   64'h54358b1ea8c5c25e
```

5-th round

- key_schedule

```
x    =   80'h28770093eb941e5dc67d
i    =   5'h05
r    =   80'h88cfa50ee0127d7003cb
```

- round

```
x    =   64'h54358b1ea8c5c25e
k    =   80'h28770093eb941e5dc67d
r    =   64'ha4c3d11e1b49abd9
```

  - key_addition

```
x    =   64'h54358b1ea8c5c25e
k    =   80'h28770093eb941e5dc67d
r    =   64'h7c428b8d4351dc03
```

  - split_0

```
x     =   64'h7c428b8d4351dc03
r0    =   4'h3
r1    =   4'h0
r2    =   4'hc
r3    =   4'hd
r4    =   4'h1
r5    =   4'h5
r6    =   4'h3
r7    =   4'h4
r8    =   4'hd
r9    =   4'h8
rA    =   4'hb
rB    =   4'h8
rC    =   4'h2
rD    =   4'h4
rE    =   4'hc
rF    =   4'h7
```

– sbox (0-th instance)

```
x     =   4'h3
r     =   4'hb
```

– sbox (1-st instance)

```
x     =   4'h0
r     =   4'hc
```

– sbox (2-nd instance)

```
x     =   4'hc
r     =   4'h4
```

– sbox (3-rd instance)

```
x     =   4'hd
r     =   4'h7
```

– sbox (4-th instance)

```
x     =   4'h1
r     =   4'h5
```

– sbox (5-th instance)

```
x     =   4'h5
r     =   4'h0
```

– sbox (6-th instance)

```
x     =   4'h3
r     =   4'hb
```

– sbox (7-th instance)

```
x     =   4'h4
r     =   4'h9
```

– sbox (8-th instance)

```
x     =   4'hd
r     =   4'h7
```

– sbox (9-th instance)

```
x     =   4'h8
r     =   4'h3
```

– sbox (10-th instance)

```
x     =   4'hb
r     =   4'h8
```

– sbox (11-st instance)

```
x     =   4'h8
r     =   4'h3
```

– sbox (12-nd instance)

```
x     =   4'h2
r     =   4'h6
```

- sbox (13-rd instance)

```
x   =  4'h4
r   =  4'h9
```

- sbox (14-th instance)

```
x   =  4'hc
r   =  4'h4
```

- sbox (15-th instance)

```
x   =  4'h7
r   =  4'hd
```

- merge_0

```
x0  =  4'hb
x1  =  4'hc
x2  =  4'h4
x3  =  4'h7
x4  =  4'h5
x5  =  4'h0
x6  =  4'hb
x7  =  4'h9
x8  =  4'h7
x9  =  4'h3
xA  =  4'h8
xB  =  4'h3
xC  =  4'h6
xD  =  4'h9
xE  =  4'h4
xF  =  4'hd
r   =  64'hd49638379b0574cb
```

- perm

```
x   =  64'hd49638379b0574cb
r   =  64'ha4c3d11e1b49abd9
```

6-th round

- key_schedule

```
x   =  80'h88cfa50ee0127d7003cb
i   =  5'h06
r   =  80'hc0797119f4a1dc014fae
```

- round

```
x   =  64'ha4c3d11e1b49abd9
k   =  80'h88cfa50ee0127d7003cb
r   =  64'h2d57fb0b808f0e0a
```

  - key_addition

```
x   =  64'ha4c3d11e1b49abd9
k   =  80'h88cfa50ee0127d7003cb
r   =  64'h2c0c7410fb5bd6a9
```

  - split_0

```
x    =   64'h2c0c7410fb5bd6a9
r0   =   4'h9
r1   =   4'ha
r2   =   4'h6
r3   =   4'hd
r4   =   4'hb
r5   =   4'h5
r6   =   4'hb
r7   =   4'hf
r8   =   4'h0
r9   =   4'h1
rA   =   4'h4
rB   =   4'h7
rC   =   4'hc
rD   =   4'h0
rE   =   4'hc
rF   =   4'h2
```

– sbox (0-th instance)

```
x    =   4'h9
r    =   4'he
```

– sbox (1-st instance)

```
x    =   4'ha
r    =   4'hf
```

– sbox (2-nd instance)

```
x    =   4'h6
r    =   4'ha
```

– sbox (3-rd instance)

```
x    =   4'hd
r    =   4'h7
```

– sbox (4-th instance)

```
x    =   4'hb
r    =   4'h8
```

– sbox (5-th instance)

```
x    =   4'h5
r    =   4'h0
```

– sbox (6-th instance)

```
x    =   4'hb
r    =   4'h8
```

– sbox (7-th instance)

```
x    =   4'hf
r    =   4'h2
```

– sbox (8-th instance)

```
x    =   4'h0
r    =   4'hc
```

– sbox (9-th instance)

```
x    =   4'h1
r    =   4'h5
```

– sbox (10-th instance)

```
x    =   4'h4
r    =   4'h9
```

– sbox (11-st instance)

```
x    =   4'h7
r    =   4'hd
```

– sbox (12-nd instance)

```
x    =   4'hc
r    =   4'h4
```

- sbox (13-rd instance)

```
x    =   4'h0
r    =   4'hc
```

- sbox (14-th instance)

```
x    =   4'hc
r    =   4'h4
```

- sbox (15-th instance)

```
x    =   4'h2
r    =   4'h6
```

- merge_0

```
x0   =   4'he
x1   =   4'hf
x2   =   4'ha
x3   =   4'h7
x4   =   4'h8
x5   =   4'h0
x6   =   4'h8
x7   =   4'h2
x8   =   4'hc
x9   =   4'h5
xA   =   4'h9
xB   =   4'hd
xC   =   4'h4
xD   =   4'hc
xE   =   4'h4
xF   =   4'h6
r    =   64'h64c4d95c28087afe
```

- perm

```
x    =   64'h64c4d95c28087afe
r    =   64'h2d57fb0b808f0e0a
```

7-th round

- key_schedule

```
x    =   80'hc0797119f4a1dc014fae
i    =   5'h07
r    =   80'h69f5d80f2e233e97bb80
```

- round

```
x    =   64'h2d57fb0b808f0e0a
k    =   80'hc0797119f4a1dc014fae
r    =   64'h04c367ae6d2cded8
```

  - key_addition

```
x    =   64'h2d57fb0b808f0e0a
k    =   80'hc0797119f4a1dc014fae
r    =   64'hed2e8a12742ed20b
```

  - split_0

```
x     =   64'hed2e8a12742ed20b
r0    =   4'hb
r1    =   4'h0
r2    =   4'h2
r3    =   4'hd
r4    =   4'he
r5    =   4'h2
r6    =   4'h4
r7    =   4'h7
r8    =   4'h2
r9    =   4'h1
rA    =   4'ha
rB    =   4'h8
rC    =   4'he
rD    =   4'h2
rE    =   4'hd
rF    =   4'he
```

- sbox (0-th instance)

```
x     =   4'hb
r     =   4'h8
```

- sbox (1-st instance)

```
x     =   4'h0
r     =   4'hc
```

- sbox (2-nd instance)

```
x     =   4'h2
r     =   4'h6
```

- sbox (3-rd instance)

```
x     =   4'hd
r     =   4'h7
```

- sbox (4-th instance)

```
x     =   4'he
r     =   4'h1
```

- sbox (5-th instance)

```
x     =   4'h2
r     =   4'h6
```

- sbox (6-th instance)

```
x     =   4'h4
r     =   4'h9
```

- sbox (7-th instance)

```
x     =   4'h7
r     =   4'hd
```

- sbox (8-th instance)

```
x     =   4'h2
r     =   4'h6
```

- sbox (9-th instance)

```
x     =   4'h1
r     =   4'h5
```

- sbox (10-th instance)

```
x     =   4'ha
r     =   4'hf
```

- sbox (11-st instance)

```
x     =   4'h8
r     =   4'h3
```

- sbox (12-nd instance)

```
x     =   4'he
r     =   4'h1
```

- sbox (13-rd instance)

```
x    =   4'h2
r    =   4'h6
```

- sbox (14-th instance)

```
x    =   4'hd
r    =   4'h7
```

- sbox (15-th instance)

```
x    =   4'he
r    =   4'h1
```

- merge_0

```
x0   =   4'h8
x1   =   4'hc
x2   =   4'h6
x3   =   4'h7
x4   =   4'h1
x5   =   4'h6
x6   =   4'h9
x7   =   4'hd
x8   =   4'h6
x9   =   4'h5
xA   =   4'hf
xB   =   4'h3
xC   =   4'h1
xD   =   4'h6
xE   =   4'h7
xF   =   4'h1
r    =   64'h17613f56d96176c8
```

- perm

```
x    =   64'h17613f56d96176c8
r    =   64'h04c367ae6d2cded8
```

8-th round

- key_schedule

```
x    =   80'h69f5d80f2e233e97bb80
i    =   5'h08
r    =   80'h27700d3ebb01e5c067d2
```

- round

```
x    =   64'h04c367ae6d2cded8
k    =   80'h69f5d80f2e233e97bb80
r    =   64'hbae64324f65163ca
```

  - key_addition

```
x    =   64'h04c367ae6d2cded8
k    =   80'h69f5d80f2e233e97bb80
r    =   64'h6d36bfa1430fe04f
```

  - split_0

```
x    =   64'h6d36bfa1430fe04f
r0   =   4'hf
r1   =   4'h4
r2   =   4'h0
r3   =   4'he
r4   =   4'hf
r5   =   4'h0
r6   =   4'h3
r7   =   4'h4
r8   =   4'h1
r9   =   4'ha
rA   =   4'hf
rB   =   4'hb
rC   =   4'h6
rD   =   4'h3
rE   =   4'hd
rF   =   4'h6
```

– sbox (0-th instance)

```
x    =   4'hf
r    =   4'h2
```

– sbox (1-st instance)

```
x    =   4'h4
r    =   4'h9
```

– sbox (2-nd instance)

```
x    =   4'h0
r    =   4'hc
```

– sbox (3-rd instance)

```
x    =   4'he
r    =   4'h1
```

– sbox (4-th instance)

```
x    =   4'hf
r    =   4'h2
```

– sbox (5-th instance)

```
x    =   4'h0
r    =   4'hc
```

– sbox (6-th instance)

```
x    =   4'h3
r    =   4'hb
```

– sbox (7-th instance)

```
x    =   4'h4
r    =   4'h9
```

– sbox (8-th instance)

```
x    =   4'h1
r    =   4'h5
```

– sbox (9-th instance)

```
x    =   4'ha
r    =   4'hf
```

– sbox (10-th instance)

```
x    =   4'hf
r    =   4'h2
```

– sbox (11-st instance)

```
x    =   4'hb
r    =   4'h8
```

– sbox (12-nd instance)

```
x    =   4'h6
r    =   4'ha
```

- sbox (13-rd instance)

```
x    =   4'h3
r    =   4'hb
```

- sbox (14-th instance)

```
x    =   4'hd
r    =   4'h7
```

- sbox (15-th instance)

```
x    =   4'h6
r    =   4'ha
```

- merge_0

```
x0   =   4'h2
x1   =   4'h9
x2   =   4'hc
x3   =   4'h1
x4   =   4'h2
x5   =   4'hc
x6   =   4'hb
x7   =   4'h9
x8   =   4'h5
x9   =   4'hf
xA   =   4'h2
xB   =   4'h8
xC   =   4'ha
xD   =   4'hb
xE   =   4'h7
xF   =   4'ha
r    =   64'ha7ba82f59bc21c92
```

- perm

```
x    =   64'ha7ba82f59bc21c92
r    =   64'hbae64324f65163ca
```

9-th round

- key_schedule

```
x    =   80'h27700d3ebb01e5c067d2
i    =   5'h09
r    =   80'hccfa44ee01a7d764bcb8
```

- round

```
x    =   64'hbae64324f65163ca
k    =   80'h27700d3ebb01e5c067d2
r    =   64'hb997e353f14d4fc9
```

  - key_addition

```
x    =   64'hbae64324f65163ca
k    =   80'h27700d3ebb01e5c067d2
r    =   64'h9d964e1a4d50860a
```

  - split_0

```
x    =   64'h9d964e1a4d50860a
r0   =   4'ha
r1   =   4'h0
r2   =   4'h6
r3   =   4'h8
r4   =   4'h0
r5   =   4'h5
r6   =   4'hd
r7   =   4'h4
r8   =   4'ha
r9   =   4'h1
rA   =   4'he
rB   =   4'h4
rC   =   4'h6
rD   =   4'h9
rE   =   4'hd
rF   =   4'h9
```

- sbox (0-th instance)

```
x    =   4'ha
r    =   4'hf
```

- sbox (1-st instance)

```
x    =   4'h0
r    =   4'hc
```

- sbox (2-nd instance)

```
x    =   4'h6
r    =   4'ha
```

- sbox (3-rd instance)

```
x    =   4'h8
r    =   4'h3
```

- sbox (4-th instance)

```
x    =   4'h0
r    =   4'hc
```

- sbox (5-th instance)

```
x    =   4'h5
r    =   4'h0
```

- sbox (6-th instance)

```
x    =   4'hd
r    =   4'h7
```

- sbox (7-th instance)

```
x    =   4'h4
r    =   4'h9
```

- sbox (8-th instance)

```
x    =   4'ha
r    =   4'hf
```

- sbox (9-th instance)

```
x    =   4'h1
r    =   4'h5
```

- sbox (10-th instance)

```
x    =   4'he
r    =   4'h1
```

- sbox (11-st instance)

```
x    =   4'h4
r    =   4'h9
```

- sbox (12-nd instance)

```
x    =   4'h6
r    =   4'ha
```

- sbox (13-rd instance)

```
x     =    4'h9
r     =    4'he
```

- sbox (14-th instance)

```
x     =    4'hd
r     =    4'h7
```

- sbox (15-th instance)

```
x     =    4'h9
r     =    4'he
```

- merge_0

```
x0    =    4'hf
x1    =    4'hc
x2    =    4'ha
x3    =    4'h3
x4    =    4'hc
x5    =    4'h0
x6    =    4'h7
x7    =    4'h9
x8    =    4'hf
x9    =    4'h5
xA    =    4'h1
xB    =    4'h9
xC    =    4'ha
xD    =    4'he
xE    =    4'h7
xF    =    4'he
r     =    64'he7ea915f970c3acf
```

- perm

```
x     =    64'he7ea915f970c3acf
r     =    64'hb997e353f14d4fc9
```

10-th round

- key_schedule

```
x     =    80'hccfa44ee01a7d764bcb8
i     =    5'h0a
r     =    80'he797199f489dc031faec
```

- round

```
x     =    64'hb997e353f14d4fc9
k     =    80'hccfa44ee01a7d764bcb8
r     =    64'hae5a9d5b399f9d37
```

  - key_addition

```
x     =    64'hb997e353f14d4fc9
k     =    80'hccfa44ee01a7d764bcb8
r     =    64'h756da7bdf0ea98ad
```

  - split_0

```
x     =   64'h756da7bdf0ea98ad
r0    =   4'hd
r1    =   4'ha
r2    =   4'h8
r3    =   4'h9
r4    =   4'ha
r5    =   4'he
r6    =   4'h0
r7    =   4'hf
r8    =   4'hd
r9    =   4'hb
rA    =   4'h7
rB    =   4'ha
rC    =   4'hd
rD    =   4'h6
rE    =   4'h5
rF    =   4'h7
```

- sbox (0-th instance)

```
x     =   4'hd
r     =   4'h7
```

- sbox (1-st instance)

```
x     =   4'ha
r     =   4'hf
```

- sbox (2-nd instance)

```
x     =   4'h8
r     =   4'h3
```

- sbox (3-rd instance)

```
x     =   4'h9
r     =   4'he
```

- sbox (4-th instance)

```
x     =   4'ha
r     =   4'hf
```

- sbox (5-th instance)

```
x     =   4'he
r     =   4'h1
```

- sbox (6-th instance)

```
x     =   4'h0
r     =   4'hc
```

- sbox (7-th instance)

```
x     =   4'hf
r     =   4'h2
```

- sbox (8-th instance)

```
x     =   4'hd
r     =   4'h7
```

- sbox (9-th instance)

```
x     =   4'hb
r     =   4'h8
```

- sbox (10-th instance)

```
x     =   4'h7
r     =   4'hd
```

- sbox (11-st instance)

```
x     =   4'ha
r     =   4'hf
```

- sbox (12-nd instance)

```
x     =   4'hd
r     =   4'h7
```

- sbox (13-rd instance)

```
x     =    4'h6
r     =    4'ha
```

- sbox (14-th instance)

```
x     =    4'h5
r     =    4'h0
```

- sbox (15-th instance)

```
x     =    4'h7
r     =    4'hd
```

- merge_0

```
x0    =    4'h7
x1    =    4'hf
x2    =    4'h3
x3    =    4'he
x4    =    4'hf
x5    =    4'h1
x6    =    4'hc
x7    =    4'h2
x8    =    4'h7
x9    =    4'h8
xA    =    4'hd
xB    =    4'hf
xC    =    4'h7
xD    =    4'ha
xE    =    4'h0
xF    =    4'hd
r     =    64'hd0a7fd872c1fe3f7
```

- perm

```
x     =    64'hd0a7fd872c1fe3f7
r     =    64'hae5a9d5b399f9d37
```

11-st round

- key_schedule

```
x     =    80'he797199f489dc031faec
i     =    5'h0b
r     =    80'hbf5d9cf2e333e9163806
```

- round

```
x     =    64'hae5a9d5b399f9d37
k     =    80'he797199f489dc031faec
r     =    64'hc5a372f658159dc4
```

  - key_addition

```
x     =    64'hae5a9d5b399f9d37
k     =    80'he797199f489dc031faec
r     =    64'h49cd84c471025d06
```

  - split_0

```
x     =   64'h49cd84c471025d06
r0    =   4'h6
r1    =   4'h0
r2    =   4'hd
r3    =   4'h5
r4    =   4'h2
r5    =   4'h0
r6    =   4'h1
r7    =   4'h7
r8    =   4'h4
r9    =   4'hc
rA    =   4'h4
rB    =   4'h8
rC    =   4'hd
rD    =   4'hc
rE    =   4'h9
rF    =   4'h4
```

- sbox (0-th instance)

```
x     =   4'h6
r     =   4'ha
```

- sbox (1-st instance)

```
x     =   4'h0
r     =   4'hc
```

- sbox (2-nd instance)

```
x     =   4'hd
r     =   4'h7
```

- sbox (3-rd instance)

```
x     =   4'h5
r     =   4'h0
```

- sbox (4-th instance)

```
x     =   4'h2
r     =   4'h6
```

- sbox (5-th instance)

```
x     =   4'h0
r     =   4'hc
```

- sbox (6-th instance)

```
x     =   4'h1
r     =   4'h5
```

- sbox (7-th instance)

```
x     =   4'h7
r     =   4'hd
```

- sbox (8-th instance)

```
x     =   4'h4
r     =   4'h9
```

- sbox (9-th instance)

```
x     =   4'hc
r     =   4'h4
```

- sbox (10-th instance)

```
x     =   4'h4
r     =   4'h9
```

- sbox (11-st instance)

```
x     =   4'h8
r     =   4'h3
```

- sbox (12-nd instance)

```
x     =   4'hd
r     =   4'h7
```

- sbox (13-rd instance)

```
x    =   4'hc
r    =   4'h4
```

- sbox (14-th instance)

```
x    =   4'h9
r    =   4'he
```

- sbox (15-th instance)

```
x    =   4'h4
r    =   4'h9
```

- merge_0

```
x0   =   4'ha
x1   =   4'hc
x2   =   4'h7
x3   =   4'h0
x4   =   4'h6
x5   =   4'hc
x6   =   4'h5
x7   =   4'hd
x8   =   4'h9
x9   =   4'h4
xA   =   4'h9
xB   =   4'h3
xC   =   4'h7
xD   =   4'h4
xE   =   4'he
xF   =   4'h9
r    =   64'h9e473949d5c607ca
```

- perm

```
x    =   64'h9e473949d5c607ca
r    =   64'hc5a372f658159dc4
```

12-nd round

- key_schedule

```
x    =   80'hbf5d9cf2e333e9163806
i    =   5'h0c
r    =   80'h4700d7ebb39e5c607d22
```

- round

```
x    =   64'hc5a372f658159dc4
k    =   80'hbf5d9cf2e333e9163806
r    =   64'hc3dcc22b6033dd0e
```

  - key_addition

```
x    =   64'hc5a372f658159dc4
k    =   80'hbf5d9cf2e333e9163806
r    =   64'h7afeee04bb2674d2
```

  - split_0

```
x     =   64'h7afeee04bb2674d2
r0    =   4'h2
r1    =   4'hd
r2    =   4'h4
r3    =   4'h7
r4    =   4'h6
r5    =   4'h2
r6    =   4'hb
r7    =   4'hb
r8    =   4'h4
r9    =   4'h0
rA    =   4'he
rB    =   4'he
rC    =   4'he
rD    =   4'hf
rE    =   4'ha
rF    =   4'h7
```

– sbox (0-th instance)

```
x     =   4'h2
r     =   4'h6
```

– sbox (1-st instance)

```
x     =   4'hd
r     =   4'h7
```

– sbox (2-nd instance)

```
x     =   4'h4
r     =   4'h9
```

– sbox (3-rd instance)

```
x     =   4'h7
r     =   4'hd
```

– sbox (4-th instance)

```
x     =   4'h6
r     =   4'ha
```

– sbox (5-th instance)

```
x     =   4'h2
r     =   4'h6
```

– sbox (6-th instance)

```
x     =   4'hb
r     =   4'h8
```

– sbox (7-th instance)

```
x     =   4'hb
r     =   4'h8
```

– sbox (8-th instance)

```
x     =   4'h4
r     =   4'h9
```

– sbox (9-th instance)

```
x     =   4'h0
r     =   4'hc
```

– sbox (10-th instance)

```
x     =   4'he
r     =   4'h1
```

– sbox (11-st instance)

```
x     =   4'he
r     =   4'h1
```

– sbox (12-nd instance)

```
x     =   4'he
r     =   4'h1
```

  - sbox (13-rd instance)

    ```
    x     =    4'hf
    r     =    4'h2
    ```

  - sbox (14-th instance)

    ```
    x     =    4'ha
    r     =    4'hf
    ```

  - sbox (15-th instance)

    ```
    x     =    4'h7
    r     =    4'hd
    ```

  - merge_0

    ```
    x0    =    4'h6
    x1    =    4'h7
    x2    =    4'h9
    x3    =    4'hd
    x4    =    4'ha
    x5    =    4'h6
    x6    =    4'h8
    x7    =    4'h8
    x8    =    4'h9
    x9    =    4'hc
    xA    =    4'h1
    xB    =    4'h1
    xC    =    4'h1
    xD    =    4'h2
    xE    =    4'hf
    xF    =    4'hd
    r     =    64'hdf2111c9886ad976
    ```

  - perm

    ```
    x     =    64'hdf2111c9886ad976
    r     =    64'hc3dcc22b6033dd0e
    ```

13-rd round

- key_schedule

  ```
  x     =    80'h4700d7ebb39e5c607d22
  i     =    5'h0d
  r     =    80'hcfa448e01afd76754b8c
  ```

- round

  ```
  x     =    64'hc3dcc22b6033dd0e
  k     =    80'h4700d7ebb39e5c607d22
  r     =    64'h41623bb4a0fae8fd
  ```

  - key_addition

    ```
    x     =    64'hc3dcc22b6033dd0e
    k     =    80'h4700d7ebb39e5c607d22
    r     =    64'h84dc15c0d3ad816e
    ```

  - split_0

```
x    =   64'h84dc15c0d3ad816e
r0   =   4'he
r1   =   4'h6
r2   =   4'h1
r3   =   4'h8
r4   =   4'hd
r5   =   4'ha
r6   =   4'h3
r7   =   4'hd
r8   =   4'h0
r9   =   4'hc
rA   =   4'h5
rB   =   4'h1
rC   =   4'hc
rD   =   4'hd
rE   =   4'h4
rF   =   4'h8
```

– sbox (0-th instance)

```
x    =   4'he
r    =   4'h1
```

– sbox (1-st instance)

```
x    =   4'h6
r    =   4'ha
```

– sbox (2-nd instance)

```
x    =   4'h1
r    =   4'h5
```

– sbox (3-rd instance)

```
x    =   4'h8
r    =   4'h3
```

– sbox (4-th instance)

```
x    =   4'hd
r    =   4'h7
```

– sbox (5-th instance)

```
x    =   4'ha
r    =   4'hf
```

– sbox (6-th instance)

```
x    =   4'h3
r    =   4'hb
```

– sbox (7-th instance)

```
x    =   4'hd
r    =   4'h7
```

– sbox (8-th instance)

```
x    =   4'h0
r    =   4'hc
```

– sbox (9-th instance)

```
x    =   4'hc
r    =   4'h4
```

– sbox (10-th instance)

```
x    =   4'h5
r    =   4'h0
```

– sbox (11-st instance)

```
x    =   4'h1
r    =   4'h5
```

– sbox (12-nd instance)

```
x    =   4'hc
r    =   4'h4
```

- sbox (13-rd instance)

```
x    =    4'hd
r    =    4'h7
```

- sbox (14-th instance)

```
x    =    4'h4
r    =    4'h9
```

- sbox (15-th instance)

```
x    =    4'h8
r    =    4'h3
```

- merge_0

```
x0   =    4'h1
x1   =    4'ha
x2   =    4'h5
x3   =    4'h3
x4   =    4'h7
x5   =    4'hf
x6   =    4'hb
x7   =    4'h7
x8   =    4'hc
x9   =    4'h4
xA   =    4'h0
xB   =    4'h5
xC   =    4'h4
xD   =    4'h7
xE   =    4'h9
xF   =    4'h3
r    =    64'h3974504c7bf735a1
```

- perm

```
x    =    64'h3974504c7bf735a1
r    =    64'h41623bb4a0fae8fd
```

14-th round

- key_schedule

```
x    =    80'hcfa448e01afd76754b8c
i    =    5'h0e
r    =    80'hf97199f4891c0358aece
```

- round

```
x    =    64'h41623bb4a0fae8fd
k    =    80'hcfa448e01afd76754b8c
r    =    64'h1df82878944bcd57
```

  - key_addition

```
x    =    64'h41623bb4a0fae8fd
k    =    80'hcfa448e01afd76754b8c
r    =    64'h8ec67354ba079e88
```

  - split_0

```
x     =   64'h8ec67354ba079e88
r0    =   4'h8
r1    =   4'h8
r2    =   4'he
r3    =   4'h9
r4    =   4'h7
r5    =   4'h0
r6    =   4'ha
r7    =   4'hb
r8    =   4'h4
r9    =   4'h5
rA    =   4'h3
rB    =   4'h7
rC    =   4'h6
rD    =   4'hc
rE    =   4'he
rF    =   4'h8
```

– sbox (0-th instance)

```
x     =   4'h8
r     =   4'h3
```

– sbox (1-st instance)

```
x     =   4'h8
r     =   4'h3
```

– sbox (2-nd instance)

```
x     =   4'he
r     =   4'h1
```

– sbox (3-rd instance)

```
x     =   4'h9
r     =   4'he
```

– sbox (4-th instance)

```
x     =   4'h7
r     =   4'hd
```

– sbox (5-th instance)

```
x     =   4'h0
r     =   4'hc
```

– sbox (6-th instance)

```
x     =   4'ha
r     =   4'hf
```

– sbox (7-th instance)

```
x     =   4'hb
r     =   4'h8
```

– sbox (8-th instance)

```
x     =   4'h4
r     =   4'h9
```

– sbox (9-th instance)

```
x     =   4'h5
r     =   4'h0
```

– sbox (10-th instance)

```
x     =   4'h3
r     =   4'hb
```

– sbox (11-st instance)

```
x     =   4'h7
r     =   4'hd
```

– sbox (12-nd instance)

```
x     =   4'h6
r     =   4'ha
```

- sbox (13-rd instance)

```
x    =   4'hc
r    =   4'h4
```

- sbox (14-th instance)

```
x    =   4'he
r    =   4'h1
```

- sbox (15-th instance)

```
x    =   4'h8
r    =   4'h3
```

- merge_0

```
x0   =   4'h3
x1   =   4'h3
x2   =   4'h1
x3   =   4'he
x4   =   4'hd
x5   =   4'hc
x6   =   4'hf
x7   =   4'h8
x8   =   4'h9
x9   =   4'h0
xA   =   4'hb
xB   =   4'hd
xC   =   4'ha
xD   =   4'h4
xE   =   4'h1
xF   =   4'h3
r    =   64'h314adb098fcde133
```

- perm

```
x    =   64'h314adb098fcde133
r    =   64'h1df82878944bcd57
```

15-th round

- key_schedule

```
x    =   80'hf97199f4891c0358aece
i    =   5'h0f
r    =   80'h55d9df2e333e9124006b
```

- round

```
x    =   64'h1df82878944bcd57
k    =   80'hf97199f4891c0358aece
r    =   64'h581215da3241e6d4
```

  - key_addition

```
x    =   64'h1df82878944bcd57
k    =   80'hf97199f4891c0358aece
r    =   64'he489b18c1d57ce0f
```

  - split_0

```
x     =   64'he489b18c1d57ce0f
r0    =   4'hf
r1    =   4'h0
r2    =   4'he
r3    =   4'hc
r4    =   4'h7
r5    =   4'h5
r6    =   4'hd
r7    =   4'h1
r8    =   4'hc
r9    =   4'h8
rA    =   4'h1
rB    =   4'hb
rC    =   4'h9
rD    =   4'h8
rE    =   4'h4
rF    =   4'he
```

– sbox (0-th instance)

```
x     =   4'hf
r     =   4'h2
```

– sbox (1-st instance)

```
x     =   4'h0
r     =   4'hc
```

– sbox (2-nd instance)

```
x     =   4'he
r     =   4'h1
```

– sbox (3-rd instance)

```
x     =   4'hc
r     =   4'h4
```

– sbox (4-th instance)

```
x     =   4'h7
r     =   4'hd
```

– sbox (5-th instance)

```
x     =   4'h5
r     =   4'h0
```

– sbox (6-th instance)

```
x     =   4'hd
r     =   4'h7
```

– sbox (7-th instance)

```
x     =   4'h1
r     =   4'h5
```

– sbox (8-th instance)

```
x     =   4'hc
r     =   4'h4
```

– sbox (9-th instance)

```
x     =   4'h8
r     =   4'h3
```

– sbox (10-th instance)

```
x     =   4'h1
r     =   4'h5
```

– sbox (11-st instance)

```
x     =   4'hb
r     =   4'h8
```

– sbox (12-nd instance)

```
x     =   4'h9
r     =   4'he
```

- sbox (13-rd instance)

```
x    =    4'h8
r    =    4'h3
```

- sbox (14-th instance)

```
x    =    4'h4
r    =    4'h9
```

- sbox (15-th instance)

```
x    =    4'he
r    =    4'h1
```

- merge_0

```
x0   =    4'h2
x1   =    4'hc
x2   =    4'h1
x3   =    4'h4
x4   =    4'hd
x5   =    4'h0
x6   =    4'h7
x7   =    4'h5
x8   =    4'h4
x9   =    4'h3
xA   =    4'h5
xB   =    4'h8
xC   =    4'he
xD   =    4'h3
xE   =    4'h9
xF   =    4'h1
r    =    64'h193e8534570d41c2
```

- perm

```
x    =    64'h193e8534570d41c2
r    =    64'h581215da3241e6d4
```

16-th round

- key_schedule

```
x    =    80'h55d9df2e333e9124006b
i    =    5'h10
r    =    80'h300d6abb3be5c66fd224
```

- round

```
x    =    64'h581215da3241e6d4
k    =    80'h55d9df2e333e9124006b
r    =    64'h95adeced4612456c
```

  - key_addition

```
x    =    64'h581215da3241e6d4
k    =    80'h55d9df2e333e9124006b
r    =    64'h0dcbcaf4017f77f0
```

  - split_0

```
x     =   64'h0dcbcaf4017f77f0
r0    =   4'h0
r1    =   4'hf
r2    =   4'h7
r3    =   4'h7
r4    =   4'hf
r5    =   4'h7
r6    =   4'h1
r7    =   4'h0
r8    =   4'h4
r9    =   4'hf
rA    =   4'ha
rB    =   4'hc
rC    =   4'hb
rD    =   4'hc
rE    =   4'hd
rF    =   4'h0
```

– sbox (0-th instance)

```
x     =   4'h0
r     =   4'hc
```

– sbox (1-st instance)

```
x     =   4'hf
r     =   4'h2
```

– sbox (2-nd instance)

```
x     =   4'h7
r     =   4'hd
```

– sbox (3-rd instance)

```
x     =   4'h7
r     =   4'hd
```

– sbox (4-th instance)

```
x     =   4'hf
r     =   4'h2
```

– sbox (5-th instance)

```
x     =   4'h7
r     =   4'hd
```

– sbox (6-th instance)

```
x     =   4'h1
r     =   4'h5
```

– sbox (7-th instance)

```
x     =   4'h0
r     =   4'hc
```

– sbox (8-th instance)

```
x     =   4'h4
r     =   4'h9
```

– sbox (9-th instance)

```
x     =   4'hf
r     =   4'h2
```

– sbox (10-th instance)

```
x     =   4'ha
r     =   4'hf
```

– sbox (11-st instance)

```
x     =   4'hc
r     =   4'h4
```

– sbox (12-nd instance)

```
x     =   4'hb
r     =   4'h8
```

- sbox (13-rd instance)

```
x     =    4'hc
r     =    4'h4
```

- sbox (14-th instance)

```
x     =    4'hd
r     =    4'h7
```

- sbox (15-th instance)

```
x     =    4'h0
r     =    4'hc
```

- merge_0

```
x0    =    4'hc
x1    =    4'h2
x2    =    4'hd
x3    =    4'hd
x4    =    4'h2
x5    =    4'hd
x6    =    4'h5
x7    =    4'hc
x8    =    4'h9
x9    =    4'h2
xA    =    4'hf
xB    =    4'h4
xC    =    4'h8
xD    =    4'h4
xE    =    4'h7
xF    =    4'hc
r     =    64'hc7484f29c5d2dd2c
```

- perm

```
x     =    64'hc7484f29c5d2dd2c
r     =    64'h95adeced4612456c
```

17-th round

- key_schedule

```
x     =    80'h300d6abb3be5c66fd224
i     =    5'h11
r     =    80'h2a448601ad57677438cd
```

- round

```
x     =    64'h95adeced4612456c
k     =    80'h300d6abb3be5c66fd224
r     =    64'hb597b0d2ad6da8dd
```

  - key_addition

```
x     =    64'h95adeced4612456c
k     =    80'h300d6abb3be5c66fd224
r     =    64'ha5a086567df78303
```

  - split_0

```
x     =   64'ha5a086567df78303
r0    =   4'h3
r1    =   4'h0
r2    =   4'h3
r3    =   4'h8
r4    =   4'h7
r5    =   4'hf
r6    =   4'hd
r7    =   4'h7
r8    =   4'h6
r9    =   4'h5
rA    =   4'h6
rB    =   4'h8
rC    =   4'h0
rD    =   4'ha
rE    =   4'h5
rF    =   4'ha
```

– sbox (0-th instance)

```
x     =   4'h3
r     =   4'hb
```

– sbox (1-st instance)

```
x     =   4'h0
r     =   4'hc
```

– sbox (2-nd instance)

```
x     =   4'h3
r     =   4'hb
```

– sbox (3-rd instance)

```
x     =   4'h8
r     =   4'h3
```

– sbox (4-th instance)

```
x     =   4'h7
r     =   4'hd
```

– sbox (5-th instance)

```
x     =   4'hf
r     =   4'h2
```

– sbox (6-th instance)

```
x     =   4'hd
r     =   4'h7
```

– sbox (7-th instance)

```
x     =   4'h7
r     =   4'hd
```

– sbox (8-th instance)

```
x     =   4'h6
r     =   4'ha
```

– sbox (9-th instance)

```
x     =   4'h5
r     =   4'h0
```

– sbox (10-th instance)

```
x     =   4'h6
r     =   4'ha
```

– sbox (11-st instance)

```
x     =   4'h8
r     =   4'h3
```

– sbox (12-nd instance)

```
x     =   4'h0
r     =   4'hc
```

- sbox (13-rd instance)

```
x    =   4'ha
r    =   4'hf
```

- sbox (14-th instance)

```
x    =   4'h5
r    =   4'h0
```

- sbox (15-th instance)

```
x    =   4'ha
r    =   4'hf
```

- merge_0

```
x0   =   4'hb
x1   =   4'hc
x2   =   4'hb
x3   =   4'h3
x4   =   4'hd
x5   =   4'h2
x6   =   4'h7
x7   =   4'hd
x8   =   4'ha
x9   =   4'h0
xA   =   4'ha
xB   =   4'h3
xC   =   4'hc
xD   =   4'hf
xE   =   4'h0
xF   =   4'hf
r    =   64'hf0fc3a0ad72d3bcb
```

- perm

```
x    =   64'hf0fc3a0ad72d3bcb
r    =   64'hb597b0d2ad6da8dd
```

18-th round

- key_schedule

```
x    =   80'h2a448601ad57677438cd
i    =   5'h12
r    =   80'h3719a54890c035a3ecee
```

- round

```
x    =   64'hb597b0d2ad6da8dd
k    =   80'h2a448601ad57677438cd
r    =   64'h9df3a2dbff373b32
```

  - key_addition

```
x    =   64'hb597b0d2ad6da8dd
k    =   80'h2a448601ad57677438cd
r    =   64'h9fd336d3003acfa9
```

  - split_0

```
x    =   64'h9fd336d3003acfa9
r0   =   4'h9
r1   =   4'ha
r2   =   4'hf
r3   =   4'hc
r4   =   4'ha
r5   =   4'h3
r6   =   4'h0
r7   =   4'h0
r8   =   4'h3
r9   =   4'hd
rA   =   4'h6
rB   =   4'h3
rC   =   4'h3
rD   =   4'hd
rE   =   4'hf
rF   =   4'h9
```

– sbox (0-th instance)

```
x    =   4'h9
r    =   4'he
```

– sbox (1-st instance)

```
x    =   4'ha
r    =   4'hf
```

– sbox (2-nd instance)

```
x    =   4'hf
r    =   4'h2
```

– sbox (3-rd instance)

```
x    =   4'hc
r    =   4'h4
```

– sbox (4-th instance)

```
x    =   4'ha
r    =   4'hf
```

– sbox (5-th instance)

```
x    =   4'h3
r    =   4'hb
```

– sbox (6-th instance)

```
x    =   4'h0
r    =   4'hc
```

– sbox (7-th instance)

```
x    =   4'h0
r    =   4'hc
```

– sbox (8-th instance)

```
x    =   4'h3
r    =   4'hb
```

– sbox (9-th instance)

```
x    =   4'hd
r    =   4'h7
```

– sbox (10-th instance)

```
x    =   4'h6
r    =   4'ha
```

– sbox (11-st instance)

```
x    =   4'h3
r    =   4'hb
```

– sbox (12-nd instance)

```
x    =   4'h3
r    =   4'hb
```

- sbox (13-rd instance)

```
x   =  4'hd
r   =  4'h7
```

- sbox (14-th instance)

```
x   =  4'hf
r   =  4'h2
```

- sbox (15-th instance)

```
x   =  4'h9
r   =  4'he
```

- merge_0

```
x0  =  4'he
x1  =  4'hf
x2  =  4'h2
x3  =  4'h4
x4  =  4'hf
x5  =  4'hb
x6  =  4'hc
x7  =  4'hc
x8  =  4'hb
x9  =  4'h7
xA  =  4'ha
xB  =  4'hb
xC  =  4'hb
xD  =  4'h7
xE  =  4'h2
xF  =  4'he
r   =  64'he27bba7bccbf42fe
```

- perm

```
x   =  64'he27bba7bccbf42fe
r   =  64'h9df3a2dbff373b32
```

19-th round

- key_schedule

```
x   =  80'h3719a54890c035a3ecee
i   =  5'h13
r   =  80'hdd9dc6e334a9121186b4
```

- round

```
x   =  64'h9df3a2dbff373b32
k   =  80'h3719a54890c035a3ecee
r   =  64'hdf9ade1bd3e2f515
```

  - key_addition

```
x   =  64'h9df3a2dbff373b32
k   =  80'h3719a54890c035a3ecee
r   =  64'haaea07936ff70e91
```

  - split_0

```
x    =   64'haaea07936ff70e91
r0   =   4'h1
r1   =   4'h9
r2   =   4'he
r3   =   4'h0
r4   =   4'h7
r5   =   4'hf
r6   =   4'hf
r7   =   4'h6
r8   =   4'h3
r9   =   4'h9
rA   =   4'h7
rB   =   4'h0
rC   =   4'ha
rD   =   4'he
rE   =   4'ha
rF   =   4'ha
```

– sbox (0-th instance)

```
x    =   4'h1
r    =   4'h5
```

– sbox (1-st instance)

```
x    =   4'h9
r    =   4'he
```

– sbox (2-nd instance)

```
x    =   4'he
r    =   4'h1
```

– sbox (3-rd instance)

```
x    =   4'h0
r    =   4'hc
```

– sbox (4-th instance)

```
x    =   4'h7
r    =   4'hd
```

– sbox (5-th instance)

```
x    =   4'hf
r    =   4'h2
```

– sbox (6-th instance)

```
x    =   4'hf
r    =   4'h2
```

– sbox (7-th instance)

```
x    =   4'h6
r    =   4'ha
```

– sbox (8-th instance)

```
x    =   4'h3
r    =   4'hb
```

– sbox (9-th instance)

```
x    =   4'h9
r    =   4'he
```

– sbox (10-th instance)

```
x    =   4'h7
r    =   4'hd
```

– sbox (11-st instance)

```
x    =   4'h0
r    =   4'hc
```

– sbox (12-nd instance)

```
x    =   4'ha
r    =   4'hf
```

- sbox (13-rd instance)

```
x   =  4'he
r   =  4'h1
```

- sbox (14-th instance)

```
x   =  4'ha
r   =  4'hf
```

- sbox (15-th instance)

```
x   =  4'ha
r   =  4'hf
```

- merge_0

```
x0  =  4'h5
x1  =  4'he
x2  =  4'h1
x3  =  4'hc
x4  =  4'hd
x5  =  4'h2
x6  =  4'h2
x7  =  4'ha
x8  =  4'hb
x9  =  4'he
xA  =  4'hd
xB  =  4'hc
xC  =  4'hf
xD  =  4'h1
xE  =  4'hf
xF  =  4'hf
r   =  64'hff1fcdeba22dc1e5
```

- perm

```
x   =  64'hff1fcdeba22dc1e5
r   =  64'hdf9ade1bd3e2f515
```

20-th round

- key_schedule

```
x   =  80'hdd9dc6e334a9121186b4
i   =  5'h14
r   =  80'hb0d69bb3b8dc669f2242
```

- round

```
x   =  64'hdf9ade1bd3e2f515
k   =  80'hdd9dc6e334a9121186b4
r   =  64'hb077f84647001ded
```

  - key_addition

```
x   =  64'hdf9ade1bd3e2f515
k   =  80'hdd9dc6e334a9121186b4
r   =  64'h020718f8e74be704
```

  - split_0

```
x    =   64'h020718f8e74be704
r0   =   4'h4
r1   =   4'h0
r2   =   4'h7
r3   =   4'he
r4   =   4'hb
r5   =   4'h4
r6   =   4'h7
r7   =   4'he
r8   =   4'h8
r9   =   4'hf
rA   =   4'h8
rB   =   4'h1
rC   =   4'h7
rD   =   4'h0
rE   =   4'h2
rF   =   4'h0
```

– sbox (0-th instance)

```
x    =   4'h4
r    =   4'h9
```

– sbox (1-st instance)

```
x    =   4'h0
r    =   4'hc
```

– sbox (2-nd instance)

```
x    =   4'h7
r    =   4'hd
```

– sbox (3-rd instance)

```
x    =   4'he
r    =   4'h1
```

– sbox (4-th instance)

```
x    =   4'hb
r    =   4'h8
```

– sbox (5-th instance)

```
x    =   4'h4
r    =   4'h9
```

– sbox (6-th instance)

```
x    =   4'h7
r    =   4'hd
```

– sbox (7-th instance)

```
x    =   4'he
r    =   4'h1
```

– sbox (8-th instance)

```
x    =   4'h8
r    =   4'h3
```

– sbox (9-th instance)

```
x    =   4'hf
r    =   4'h2
```

– sbox (10-th instance)

```
x    =   4'h8
r    =   4'h3
```

– sbox (11-st instance)

```
x    =   4'h1
r    =   4'h5
```

– sbox (12-nd instance)

```
x    =   4'h7
r    =   4'hd
```

- sbox (13-rd instance)

```
x   =   4'h0
r   =   4'hc
```

- sbox (14-th instance)

```
x   =   4'h2
r   =   4'h6
```

- sbox (15-th instance)

```
x   =   4'h0
r   =   4'hc
```

- merge_0

```
x0  =   4'h9
x1  =   4'hc
x2  =   4'hd
x3  =   4'h1
x4  =   4'h8
x5  =   4'h9
x6  =   4'hd
x7  =   4'h1
x8  =   4'h3
x9  =   4'h2
xA  =   4'h3
xB  =   4'h5
xC  =   4'hd
xD  =   4'hc
xE  =   4'h6
xF  =   4'hc
r   =   64'hc6cd53231d981dc9
```

- perm

```
x   =   64'hc6cd53231d981dc9
r   =   64'hb077f84647001ded
```

21-st round

- key_schedule

```
x   =   80'hb0d69bb3b8dc669f2242
i   =   5'h15
r   =   80'h1448561ad37677110cd3
```

- round

```
x   =   64'hb077f84647001ded
k   =   80'hb0d69bb3b8dc669f2242
r   =   64'hec0ef03b2ee1342a
```

- key_addition

```
x   =   64'hb077f84647001ded
k   =   80'hb0d69bb3b8dc669f2242
r   =   64'h00a163f5ffdc7b72
```

- split_0

```
x    =   64'h00a163f5ffdc7b72
r0   =   4'h2
r1   =   4'h7
r2   =   4'hb
r3   =   4'h7
r4   =   4'hc
r5   =   4'hd
r6   =   4'hf
r7   =   4'hf
r8   =   4'h5
r9   =   4'hf
rA   =   4'h3
rB   =   4'h6
rC   =   4'h1
rD   =   4'ha
rE   =   4'h0
rF   =   4'h0
```

- sbox (0-th instance)

```
x    =   4'h2
r    =   4'h6
```

- sbox (1-st instance)

```
x    =   4'h7
r    =   4'hd
```

- sbox (2-nd instance)

```
x    =   4'hb
r    =   4'h8
```

- sbox (3-rd instance)

```
x    =   4'h7
r    =   4'hd
```

- sbox (4-th instance)

```
x    =   4'hc
r    =   4'h4
```

- sbox (5-th instance)

```
x    =   4'hd
r    =   4'h7
```

- sbox (6-th instance)

```
x    =   4'hf
r    =   4'h2
```

- sbox (7-th instance)

```
x    =   4'hf
r    =   4'h2
```

- sbox (8-th instance)

```
x    =   4'h5
r    =   4'h0
```

- sbox (9-th instance)

```
x    =   4'hf
r    =   4'h2
```

- sbox (10-th instance)

```
x    =   4'h3
r    =   4'hb
```

- sbox (11-st instance)

```
x    =   4'h6
r    =   4'ha
```

- sbox (12-nd instance)

```
x    =   4'h1
r    =   4'h5
```

- sbox (13-rd instance)

```
x   =   4'ha
r   =   4'hf
```

- sbox (14-th instance)

```
x   =   4'h0
r   =   4'hc
```

- sbox (15-th instance)

```
x   =   4'h0
r   =   4'hc
```

- merge_0

```
x0  =   4'h6
x1  =   4'hd
x2  =   4'h8
x3  =   4'hd
x4  =   4'h4
x5  =   4'h7
x6  =   4'h2
x7  =   4'h2
x8  =   4'h0
x9  =   4'h2
xA  =   4'hb
xB  =   4'ha
xC  =   4'h5
xD  =   4'hf
xE  =   4'hc
xF  =   4'hc
r   =   64'hccf5ab202274d8d6
```

- perm

```
x   =   64'hccf5ab202274d8d6
r   =   64'hec0ef03b2ee1342a
```

22-nd round

- key_schedule

```
x   =   80'h1448561ad37677110cd3
i   =   5'h16
r   =   80'h619a62890ac35a65cee2
```

- round

```
x   =   64'hec0ef03b2ee1342a
k   =   80'h1448561ad37677110cd3
r   =   64'h3c3f0b70dee6695e
```

  - key_addition

```
x   =   64'hec0ef03b2ee1342a
k   =   80'h1448561ad37677110cd3
r   =   64'hf846a621fd97433b
```

  - split_0

```
x    =   64'hf846a621fd97433b
r0   =   4'hb
r1   =   4'h3
r2   =   4'h3
r3   =   4'h4
r4   =   4'h7
r5   =   4'h9
r6   =   4'hd
r7   =   4'hf
r8   =   4'h1
r9   =   4'h2
rA   =   4'h6
rB   =   4'ha
rC   =   4'h6
rD   =   4'h4
rE   =   4'h8
rF   =   4'hf
```

– sbox (0-th instance)

```
x    =   4'hb
r    =   4'h8
```

– sbox (1-st instance)

```
x    =   4'h3
r    =   4'hb
```

– sbox (2-nd instance)

```
x    =   4'h3
r    =   4'hb
```

– sbox (3-rd instance)

```
x    =   4'h4
r    =   4'h9
```

– sbox (4-th instance)

```
x    =   4'h7
r    =   4'hd
```

– sbox (5-th instance)

```
x    =   4'h9
r    =   4'he
```

– sbox (6-th instance)

```
x    =   4'hd
r    =   4'h7
```

– sbox (7-th instance)

```
x    =   4'hf
r    =   4'h2
```

– sbox (8-th instance)

```
x    =   4'h1
r    =   4'h5
```

– sbox (9-th instance)

```
x    =   4'h2
r    =   4'h6
```

– sbox (10-th instance)

```
x    =   4'h6
r    =   4'ha
```

– sbox (11-st instance)

```
x    =   4'ha
r    =   4'hf
```

– sbox (12-nd instance)

```
x    =   4'h6
r    =   4'ha
```

- sbox (13-rd instance)

```
x    =   4'h4
r    =   4'h9
```

- sbox (14-th instance)

```
x    =   4'h8
r    =   4'h3
```

- sbox (15-th instance)

```
x    =   4'hf
r    =   4'h2
```

- merge_0

```
x0   =   4'h8
x1   =   4'hb
x2   =   4'hb
x3   =   4'h9
x4   =   4'hd
x5   =   4'he
x6   =   4'h7
x7   =   4'h2
x8   =   4'h5
x9   =   4'h6
xA   =   4'ha
xB   =   4'hf
xC   =   4'ha
xD   =   4'h9
xE   =   4'h3
xF   =   4'h2
r    =   64'h239afa6527ed9bb8
```

- perm

```
x    =   64'h239afa6527ed9bb8
r    =   64'h3c3f0b70dee6695e
```

23-rd round

- key_schedule

```
x    =   80'h619a62890ac35a65cee2
i    =   5'h17
r    =   80'h89dc4c334c512153eb4c
```

- round

```
x    =   64'h3c3f0b70dee6695e
k    =   80'h619a62890ac35a65cee2
r    =   64'h2d4f65a06fae60ce
```

  - key_addition

```
x    =   64'h3c3f0b70dee6695e
k    =   80'h619a62890ac35a65cee2
r    =   64'h5da569f9d425333b
```

  - split_0

```
x     =   64'h5da569f9d425333b
r0    =   4'hb
r1    =   4'h3
r2    =   4'h3
r3    =   4'h3
r4    =   4'h5
r5    =   4'h2
r6    =   4'h4
r7    =   4'hd
r8    =   4'h9
r9    =   4'hf
rA    =   4'h9
rB    =   4'h6
rC    =   4'h5
rD    =   4'ha
rE    =   4'hd
rF    =   4'h5
```

– sbox (0-th instance)

```
x     =   4'hb
r     =   4'h8
```

– sbox (1-st instance)

```
x     =   4'h3
r     =   4'hb
```

– sbox (2-nd instance)

```
x     =   4'h3
r     =   4'hb
```

– sbox (3-rd instance)

```
x     =   4'h3
r     =   4'hb
```

– sbox (4-th instance)

```
x     =   4'h5
r     =   4'h0
```

– sbox (5-th instance)

```
x     =   4'h2
r     =   4'h6
```

– sbox (6-th instance)

```
x     =   4'h4
r     =   4'h9
```

– sbox (7-th instance)

```
x     =   4'hd
r     =   4'h7
```

– sbox (8-th instance)

```
x     =   4'h9
r     =   4'he
```

– sbox (9-th instance)

```
x     =   4'hf
r     =   4'h2
```

– sbox (10-th instance)

```
x     =   4'h9
r     =   4'he
```

– sbox (11-st instance)

```
x     =   4'h6
r     =   4'ha
```

– sbox (12-nd instance)

```
x     =   4'h5
r     =   4'h0
```

- sbox (13-rd instance)

```
x    =   4'ha
r    =   4'hf
```

- sbox (14-th instance)

```
x    =   4'hd
r    =   4'h7
```

- sbox (15-th instance)

```
x    =   4'h5
r    =   4'h0
```

- merge_0

```
x0   =   4'h8
x1   =   4'hb
x2   =   4'hb
x3   =   4'hb
x4   =   4'h0
x5   =   4'h6
x6   =   4'h9
x7   =   4'h7
x8   =   4'he
x9   =   4'h2
xA   =   4'he
xB   =   4'ha
xC   =   4'h0
xD   =   4'hf
xE   =   4'h7
xF   =   4'h0
r    =   64'h07f0ae2e7960bbb8
```

- perm

```
x    =   64'h07f0ae2e7960bbb8
r    =   64'h2d4f65a06fae60ce
```

24-th round

- key_schedule

```
x    =   80'h89dc4c334c512153eb4c
i    =   5'h18
r    =   80'hdd69913b89866986242a
```

- round

```
x    =   64'h2d4f65a06fae60ce
k    =   80'h89dc4c334c512153eb4c
r    =   64'hf74aae87bff3d14d
```

  - key_addition

```
x    =   64'h2d4f65a06fae60ce
k    =   80'h89dc4c334c512153eb4c
r    =   64'ha493299323ff419d
```

  - split_0

```
x    =   64'ha493299323ff419d
r0   =   4'hd
r1   =   4'h9
r2   =   4'h1
r3   =   4'h4
r4   =   4'hf
r5   =   4'hf
r6   =   4'h3
r7   =   4'h2
r8   =   4'h3
r9   =   4'h9
rA   =   4'h9
rB   =   4'h2
rC   =   4'h3
rD   =   4'h9
rE   =   4'h4
rF   =   4'ha
```

– sbox (0-th instance)

```
x    =   4'hd
r    =   4'h7
```

– sbox (1-st instance)

```
x    =   4'h9
r    =   4'he
```

– sbox (2-nd instance)

```
x    =   4'h1
r    =   4'h5
```

– sbox (3-rd instance)

```
x    =   4'h4
r    =   4'h9
```

– sbox (4-th instance)

```
x    =   4'hf
r    =   4'h2
```

– sbox (5-th instance)

```
x    =   4'hf
r    =   4'h2
```

– sbox (6-th instance)

```
x    =   4'h3
r    =   4'hb
```

– sbox (7-th instance)

```
x    =   4'h2
r    =   4'h6
```

– sbox (8-th instance)

```
x    =   4'h3
r    =   4'hb
```

– sbox (9-th instance)

```
x    =   4'h9
r    =   4'he
```

– sbox (10-th instance)

```
x    =   4'h9
r    =   4'he
```

– sbox (11-st instance)

```
x    =   4'h2
r    =   4'h6
```

– sbox (12-nd instance)

```
x    =   4'h3
r    =   4'hb
```

- sbox (13-rd instance)

```
x     =    4'h9
r     =    4'he
```

- sbox (14-th instance)

```
x     =    4'h4
r     =    4'h9
```

- sbox (15-th instance)

```
x     =    4'ha
r     =    4'hf
```

- merge_0

```
x0    =    4'h7
x1    =    4'he
x2    =    4'h5
x3    =    4'h9
x4    =    4'h2
x5    =    4'h2
x6    =    4'hb
x7    =    4'h6
x8    =    4'hb
x9    =    4'he
xA    =    4'he
xB    =    4'h6
xC    =    4'hb
xD    =    4'he
xE    =    4'h9
xF    =    4'hf
r     =    64'hf9eb6eeb6b2295e7
```

- perm

```
x     =    64'hf9eb6eeb6b2295e7
r     =    64'hf74aae87bff3d14d
```

25-th round

- key_schedule

```
x     =    80'hdd69913b89866986242a
i     =    5'h19
r     =    80'h44855bad3227713c4d30
```

- round

```
x     =    64'hf74aae87bff3d14d
k     =    80'hdd69913b89866986242a
r     =    64'h5ae9e122fcc458a4
```

  - key_addition

```
x     =    64'hf74aae87bff3d14d
k     =    80'hdd69913b89866986242a
r     =    64'h2a233fbc3675b8cb
```

  - split_0

```
x     =   64'h2a233fbc3675b8cb
r0    =   4'hb
r1    =   4'hc
r2    =   4'h8
r3    =   4'hb
r4    =   4'h5
r5    =   4'h7
r6    =   4'h6
r7    =   4'h3
r8    =   4'hc
r9    =   4'hb
rA    =   4'hf
rB    =   4'h3
rC    =   4'h3
rD    =   4'h2
rE    =   4'ha
rF    =   4'h2
```

– sbox (0-th instance)

```
x     =   4'hb
r     =   4'h8
```

– sbox (1-st instance)

```
x     =   4'hc
r     =   4'h4
```

– sbox (2-nd instance)

```
x     =   4'h8
r     =   4'h3
```

– sbox (3-rd instance)

```
x     =   4'hb
r     =   4'h8
```

– sbox (4-th instance)

```
x     =   4'h5
r     =   4'h0
```

– sbox (5-th instance)

```
x     =   4'h7
r     =   4'hd
```

– sbox (6-th instance)

```
x     =   4'h6
r     =   4'ha
```

– sbox (7-th instance)

```
x     =   4'h3
r     =   4'hb
```

– sbox (8-th instance)

```
x     =   4'hc
r     =   4'h4
```

– sbox (9-th instance)

```
x     =   4'hb
r     =   4'h8
```

– sbox (10-th instance)

```
x     =   4'hf
r     =   4'h2
```

– sbox (11-st instance)

```
x     =   4'h3
r     =   4'hb
```

– sbox (12-nd instance)

```
x     =   4'h3
r     =   4'hb
```

- sbox (13-rd instance)

```
x     =    4'h2
r     =    4'h6
```

- sbox (14-th instance)

```
x     =    4'ha
r     =    4'hf
```

- sbox (15-th instance)

```
x     =    4'h2
r     =    4'h6
```

- merge_0

```
x0    =    4'h8
x1    =    4'h4
x2    =    4'h3
x3    =    4'h8
x4    =    4'h0
x5    =    4'hd
x6    =    4'ha
x7    =    4'hb
x8    =    4'h4
x9    =    4'h8
xA    =    4'h2
xB    =    4'hb
xC    =    4'hb
xD    =    4'h6
xE    =    4'hf
xF    =    4'h6
r     =    64'h6f6bb284bad08348
```

- perm

```
x     =    64'h6f6bb284bad08348
r     =    64'h5ae9e122fcc458a4
```

26-th round

- key_schedule

```
x     =    80'h44855bad3227713c4d30
i     =    5'h1a
r     =    80'h39a60890ab75a649ee27
```

- round

```
x     =    64'h5ae9e122fcc458a4
k     =    80'h44855bad3227713c4d30
r     =    64'h2c16948e271fc671
```

  - key_addition

```
x     =    64'h5ae9e122fcc458a4
k     =    80'h44855bad3227713c4d30
r     =    64'h1e6cba8fcee32998
```

  - split_0

```
x    =  64'h1e6cba8fcee32998
r0   =  4'h8
r1   =  4'h9
r2   =  4'h9
r3   =  4'h2
r4   =  4'h3
r5   =  4'he
r6   =  4'he
r7   =  4'hc
r8   =  4'hf
r9   =  4'h8
rA   =  4'ha
rB   =  4'hb
rC   =  4'hc
rD   =  4'h6
rE   =  4'he
rF   =  4'h1
```

– sbox (0-th instance)

```
x    =  4'h8
r    =  4'h3
```

– sbox (1-st instance)

```
x    =  4'h9
r    =  4'he
```

– sbox (2-nd instance)

```
x    =  4'h9
r    =  4'he
```

– sbox (3-rd instance)

```
x    =  4'h2
r    =  4'h6
```

– sbox (4-th instance)

```
x    =  4'h3
r    =  4'hb
```

– sbox (5-th instance)

```
x    =  4'he
r    =  4'h1
```

– sbox (6-th instance)

```
x    =  4'he
r    =  4'h1
```

– sbox (7-th instance)

```
x    =  4'hc
r    =  4'h4
```

– sbox (8-th instance)

```
x    =  4'hf
r    =  4'h2
```

– sbox (9-th instance)

```
x    =  4'h8
r    =  4'h3
```

– sbox (10-th instance)

```
x    =  4'ha
r    =  4'hf
```

– sbox (11-st instance)

```
x    =  4'hb
r    =  4'h8
```

– sbox (12-nd instance)

```
x    =  4'hc
r    =  4'h4
```

- sbox (13-rd instance)

```
x    =   4'h6
r    =   4'ha
```

- sbox (14-th instance)

```
x    =   4'he
r    =   4'h1
```

- sbox (15-th instance)

```
x    =   4'h1
r    =   4'h5
```

- merge_0

```
x0   =   4'h3
x1   =   4'he
x2   =   4'he
x3   =   4'h6
x4   =   4'hb
x5   =   4'h1
x6   =   4'h1
x7   =   4'h4
x8   =   4'h2
x9   =   4'h3
xA   =   4'hf
xB   =   4'h8
xC   =   4'h4
xD   =   4'ha
xE   =   4'h1
xF   =   4'h5
r    =   64'h51a48f32411b6ee3
```

- perm

```
x    =   64'h51a48f32411b6ee3
r    =   64'h2c16948e271fc671
```

27-th round

- key_schedule

```
x    =   80'h39a60890ab75a649ee27
i    =   5'h1b
r    =   80'hbdc4e734c112156334c9
```

- round

```
x    =   64'h2c16948e271fc671
k    =   80'h39a60890ab75a649ee27
r    =   64'h383e9e5408bb8393
```

  - key_addition

```
x    =   64'h2c16948e271fc671
k    =   80'h39a60890ab75a649ee27
r    =   64'h15b09c1e8c6a6038
```

  - split_0

```
x     =   64'h15b09c1e8c6a6038
r0    =   4'h8
r1    =   4'h3
r2    =   4'h0
r3    =   4'h6
r4    =   4'ha
r5    =   4'h6
r6    =   4'hc
r7    =   4'h8
r8    =   4'he
r9    =   4'h1
rA    =   4'hc
rB    =   4'h9
rC    =   4'h0
rD    =   4'hb
rE    =   4'h5
rF    =   4'h1
```

– sbox (0-th instance)

```
x     =   4'h8
r     =   4'h3
```

– sbox (1-st instance)

```
x     =   4'h3
r     =   4'hb
```

– sbox (2-nd instance)

```
x     =   4'h0
r     =   4'hc
```

– sbox (3-rd instance)

```
x     =   4'h6
r     =   4'ha
```

– sbox (4-th instance)

```
x     =   4'ha
r     =   4'hf
```

– sbox (5-th instance)

```
x     =   4'h6
r     =   4'ha
```

– sbox (6-th instance)

```
x     =   4'hc
r     =   4'h4
```

– sbox (7-th instance)

```
x     =   4'h8
r     =   4'h3
```

– sbox (8-th instance)

```
x     =   4'he
r     =   4'h1
```

– sbox (9-th instance)

```
x     =   4'h1
r     =   4'h5
```

– sbox (10-th instance)

```
x     =   4'hc
r     =   4'h4
```

– sbox (11-st instance)

```
x     =   4'h9
r     =   4'he
```

– sbox (12-nd instance)

```
x     =   4'h0
r     =   4'hc
```

- sbox (13-rd instance)

```
x    =   4'hb
r    =   4'h8
```

- sbox (14-th instance)

```
x    =   4'h5
r    =   4'h0
```

- sbox (15-th instance)

```
x    =   4'h1
r    =   4'h5
```

- merge_0

```
x0   =   4'h3
x1   =   4'hb
x2   =   4'hc
x3   =   4'ha
x4   =   4'hf
x5   =   4'ha
x6   =   4'h4
x7   =   4'h3
x8   =   4'h1
x9   =   4'h5
xA   =   4'h4
xB   =   4'he
xC   =   4'hc
xD   =   4'h8
xE   =   4'h0
xF   =   4'h5
r    =   64'h508ce45134afacb3
```

- perm

```
x    =   64'h508ce45134afacb3
r    =   64'h383e9e5408bb8393
```

28-th round

- key_schedule

```
x    =   80'hbdc4e734c112156334c9
i    =   5'h1c
r    =   80'ha69937b89ce6982c42ac
```

- round

```
x    =   64'h383e9e5408bb8393
k    =   80'hbdc4e734c112156334c9
r    =   64'h1f7d1df9b67e9820
```

- key_addition

```
x    =   64'h383e9e5408bb8393
k    =   80'hbdc4e734c112156334c9
r    =   64'h85fa7960c9a996f0
```

- split_0

```
x     =   64'h85fa7960c9a996f0
r0    =   4'h0
r1    =   4'hf
r2    =   4'h6
r3    =   4'h9
r4    =   4'h9
r5    =   4'ha
r6    =   4'h9
r7    =   4'hc
r8    =   4'h0
r9    =   4'h6
rA    =   4'h9
rB    =   4'h7
rC    =   4'ha
rD    =   4'hf
rE    =   4'h5
rF    =   4'h8
```

– sbox (0-th instance)

```
x     =   4'h0
r     =   4'hc
```

– sbox (1-st instance)

```
x     =   4'hf
r     =   4'h2
```

– sbox (2-nd instance)

```
x     =   4'h6
r     =   4'ha
```

– sbox (3-rd instance)

```
x     =   4'h9
r     =   4'he
```

– sbox (4-th instance)

```
x     =   4'h9
r     =   4'he
```

– sbox (5-th instance)

```
x     =   4'ha
r     =   4'hf
```

– sbox (6-th instance)

```
x     =   4'h9
r     =   4'he
```

– sbox (7-th instance)

```
x     =   4'hc
r     =   4'h4
```

– sbox (8-th instance)

```
x     =   4'h0
r     =   4'hc
```

– sbox (9-th instance)

```
x     =   4'h6
r     =   4'ha
```

– sbox (10-th instance)

```
x     =   4'h9
r     =   4'he
```

– sbox (11-st instance)

```
x     =   4'h7
r     =   4'hd
```

– sbox (12-nd instance)

```
x     =   4'ha
r     =   4'hf
```

- sbox (13-rd instance)

```
x    =   4'hf
r    =   4'h2
```

- sbox (14-th instance)

```
x    =   4'h5
r    =   4'h0
```

- sbox (15-th instance)

```
x    =   4'h8
r    =   4'h3
```

- merge_0

```
x0   =   4'hc
x1   =   4'h2
x2   =   4'ha
x3   =   4'he
x4   =   4'he
x5   =   4'hf
x6   =   4'he
x7   =   4'h4
x8   =   4'hc
x9   =   4'ha
xA   =   4'he
xB   =   4'hd
xC   =   4'hf
xD   =   4'h2
xE   =   4'h0
xF   =   4'h3
r    =   64'h302fdeac4efeea2c
```

- perm

```
x    =   64'h302fdeac4efeea2c
r    =   64'h1f7d1df9b67e9820
```

29-th round

- key_schedule

```
x    =   80'ha69937b89ce6982c42ac
i    =   5'h1d
r    =   80'h385594d326f713925305
```

- round

```
x    =   64'h1f7d1df9b67e9820
k    =   80'ha69937b89ce6982c42ac
r    =   64'hd66e4def4cf03750
```

  - key_addition

```
x    =   64'h1f7d1df9b67e9820
k    =   80'ha69937b89ce6982c42ac
r    =   64'hb9e42a412a98000c
```

  - split_0

```
x    =   64'hb9e42a412a98000c
r0   =   4'hc
r1   =   4'h0
r2   =   4'h0
r3   =   4'h0
r4   =   4'h8
r5   =   4'h9
r6   =   4'ha
r7   =   4'h2
r8   =   4'h1
r9   =   4'h4
rA   =   4'ha
rB   =   4'h2
rC   =   4'h4
rD   =   4'he
rE   =   4'h9
rF   =   4'hb
```

- sbox (0-th instance)

```
x    =   4'hc
r    =   4'h4
```

- sbox (1-st instance)

```
x    =   4'h0
r    =   4'hc
```

- sbox (2-nd instance)

```
x    =   4'h0
r    =   4'hc
```

- sbox (3-rd instance)

```
x    =   4'h0
r    =   4'hc
```

- sbox (4-th instance)

```
x    =   4'h8
r    =   4'h3
```

- sbox (5-th instance)

```
x    =   4'h9
r    =   4'he
```

- sbox (6-th instance)

```
x    =   4'ha
r    =   4'hf
```

- sbox (7-th instance)

```
x    =   4'h2
r    =   4'h6
```

- sbox (8-th instance)

```
x    =   4'h1
r    =   4'h5
```

- sbox (9-th instance)

```
x    =   4'h4
r    =   4'h9
```

- sbox (10-th instance)

```
x    =   4'ha
r    =   4'hf
```

- sbox (11-st instance)

```
x    =   4'h2
r    =   4'h6
```

- sbox (12-nd instance)

```
x    =   4'h4
r    =   4'h9
```

- sbox (13-rd instance)

```
x    =   4'he
r    =   4'h1
```

- sbox (14-th instance)

```
x    =   4'h9
r    =   4'he
```

- sbox (15-th instance)

```
x    =   4'hb
r    =   4'h8
```

- merge_0

```
x0   =   4'h4
x1   =   4'hc
x2   =   4'hc
x3   =   4'hc
x4   =   4'h3
x5   =   4'he
x6   =   4'hf
x7   =   4'h6
x8   =   4'h5
x9   =   4'h9
xA   =   4'hf
xB   =   4'h6
xC   =   4'h9
xD   =   4'h1
xE   =   4'he
xF   =   4'h8
r    =   64'h8e196f956fe3ccc4
```

- perm

```
x    =   64'h8e196f956fe3ccc4
r    =   64'hd66e4def4cf03750
```

30-th round

- key_schedule

```
x    =   80'h385594d326f713925305
i    =   5'h1e
r    =   80'h9a60a70ab29a64d1e272
```

- round

```
x    =   64'hd66e4def4cf03750
k    =   80'h385594d326f713925305
r    =   64'h36f40d7b2ec9ea54
```

  - key_addition

```
x    =   64'hd66e4def4cf03750
k    =   80'h385594d326f713925305
r    =   64'hee3bd93c6a0724c2
```

  - split_0

```
x    =   64'hee3bd93c6a0724c2
r0   =   4'h2
r1   =   4'hc
r2   =   4'h4
r3   =   4'h2
r4   =   4'h7
r5   =   4'h0
r6   =   4'ha
r7   =   4'h6
r8   =   4'hc
r9   =   4'h3
rA   =   4'h9
rB   =   4'hd
rC   =   4'hb
rD   =   4'h3
rE   =   4'he
rF   =   4'he
```

– sbox (0-th instance)

```
x    =   4'h2
r    =   4'h6
```

– sbox (1-st instance)

```
x    =   4'hc
r    =   4'h4
```

– sbox (2-nd instance)

```
x    =   4'h4
r    =   4'h9
```

– sbox (3-rd instance)

```
x    =   4'h2
r    =   4'h6
```

– sbox (4-th instance)

```
x    =   4'h7
r    =   4'hd
```

– sbox (5-th instance)

```
x    =   4'h0
r    =   4'hc
```

– sbox (6-th instance)

```
x    =   4'ha
r    =   4'hf
```

– sbox (7-th instance)

```
x    =   4'h6
r    =   4'ha
```

– sbox (8-th instance)

```
x    =   4'hc
r    =   4'h4
```

– sbox (9-th instance)

```
x    =   4'h3
r    =   4'hb
```

– sbox (10-th instance)

```
x    =   4'h9
r    =   4'he
```

– sbox (11-st instance)

```
x    =   4'hd
r    =   4'h7
```

– sbox (12-nd instance)

```
x    =   4'hb
r    =   4'h8
```

- sbox (13-rd instance)

```
x    =   4'h3
r    =   4'hb
```

- sbox (14-th instance)

```
x    =   4'he
r    =   4'h1
```

- sbox (15-th instance)

```
x    =   4'he
r    =   4'h1
```

- merge_0

```
x0   =   4'h6
x1   =   4'h4
x2   =   4'h9
x3   =   4'h6
x4   =   4'hd
x5   =   4'hc
x6   =   4'hf
x7   =   4'ha
x8   =   4'h4
x9   =   4'hb
xA   =   4'he
xB   =   4'h7
xC   =   4'h8
xD   =   4'hb
xE   =   4'h1
xF   =   4'h1
r    =   64'h11b87eb4afcd6946
```

- perm

```
x    =   64'h11b87eb4afcd6946
r    =   64'h36f40d7b2ec9ea54
```

31-st round

- key_schedule

```
x    =   80'h9a60a70ab29a64d1e272
i    =   5'h1f
r    =   80'hbc4e534c14e1565ccc9a
```

- round

```
x    =   64'h36f40d7b2ec9ea54
k    =   80'h9a60a70ab29a64d1e272
r    =   64'hbe90efc0ac9a9f1e
```

  - key_addition

```
x    =   64'h36f40d7b2ec9ea54
k    =   80'h9a60a70ab29a64d1e272
r    =   64'hac94aa719c538e85
```

  - split_0

```
x    =   64'hac94aa719c538e85
r0   =   4'h5
r1   =   4'h8
r2   =   4'he
r3   =   4'h8
r4   =   4'h3
r5   =   4'h5
r6   =   4'hc
r7   =   4'h9
r8   =   4'h1
r9   =   4'h7
rA   =   4'ha
rB   =   4'ha
rC   =   4'h4
rD   =   4'h9
rE   =   4'hc
rF   =   4'ha
```

- sbox (0-th instance)

```
x    =   4'h5
r    =   4'h0
```

- sbox (1-st instance)

```
x    =   4'h8
r    =   4'h3
```

- sbox (2-nd instance)

```
x    =   4'he
r    =   4'h1
```

- sbox (3-rd instance)

```
x    =   4'h8
r    =   4'h3
```

- sbox (4-th instance)

```
x    =   4'h3
r    =   4'hb
```

- sbox (5-th instance)

```
x    =   4'h5
r    =   4'h0
```

- sbox (6-th instance)

```
x    =   4'hc
r    =   4'h4
```

- sbox (7-th instance)

```
x    =   4'h9
r    =   4'he
```

- sbox (8-th instance)

```
x    =   4'h1
r    =   4'h5
```

- sbox (9-th instance)

```
x    =   4'h7
r    =   4'hd
```

- sbox (10-th instance)

```
x    =   4'ha
r    =   4'hf
```

- sbox (11-st instance)

```
x    =   4'ha
r    =   4'hf
```

- sbox (12-nd instance)

```
x    =   4'h4
r    =   4'h9
```

- sbox (13-rd instance)

```
x    =   4'h9
r    =   4'he
```

- sbox (14-th instance)

```
x    =   4'hc
r    =   4'h4
```

- sbox (15-th instance)

```
x    =   4'ha
r    =   4'hf
```

- merge_0

```
x0   =   4'h0
x1   =   4'h3
x2   =   4'h1
x3   =   4'h3
x4   =   4'hb
x5   =   4'h0
x6   =   4'h4
x7   =   4'he
x8   =   4'h5
x9   =   4'hd
xA   =   4'hf
xB   =   4'hf
xC   =   4'h9
xD   =   4'he
xE   =   4'h4
xF   =   4'hf
r    =   64'hf4e9ffd5e40b3130
```

- perm

```
x    =   64'hf4e9ffd5e40b3130
r    =   64'hbe90efc0ac9a9f1e
```

Post-processing

- key_addition

```
x    =   64'hbe90efc0ac9a9f1e
k    =   80'hbc4e534c14e1565ccc9a
r    =   64'h02debc8cb87bc942
```

Output

```
c    =   64'h02debc8cb87bc942
```