Modelling and Verification of Biphase Mark Protocols in Duration Calculus Using PVS

Dang Van Hung*
The United Nations University
International Institute for Software Technology
P.O.Box 3058, Macau

Abstract

The paper presents a model of Biphase Mark Protocols (BMP) using Duration Calculus, which seems to be more general and more intuitive than the others in the literature [9]. With Duration Calculus we can model the behaviour of the bus in a natural way and in more detail. The model makes it possible to specify and verify BMP using PVS/DC$^-$ [8] tool. The mechanical verification not only ensures the correctness of the protocol, but also helps engineers to choose the optimal values for the parameters given the ratio between the clock rates of the sender and the receiver and the time for changing the signal from high to low and low to high.

1 Introduction

Using formal methods is a key solution for building a correct system. In this paper, we apply formal methods to a simple and well-known industrial case study of real time systems, namely the Biphase Mark Protocols.

In a previous paper [6], we used Duration Calculus to model communication protocols taking into account how long it takes for the sender to change the signal from high to low and vice-versa. The DC model of communication protocols given in that paper is proved to be more general and intuitive than the others in the literature [9, 1].

Duration Calculus (DC) [2] is a logic to reason about boolean functions based on interval temporal logics. This makes it one of the most suitable logics for specifying the communication protocols because the signal sent and received are usually modelled by boolean functions of time. In a previous paper we found that using DC we can model some other aspects of the protocol that we have to abstract away in other formalisms such as edges and time distance between them. Because of these advantages, the results of analysing the protocols in the model using DC may have some contributions to an improvement of the protocols. Hence, in this paper we discuss in detail the clock rate difference and optimal design of the protocols. Before doing so, in order to be sure whether the analysis in the model is correct, we mechanise the verification of the protocols in our model. Fortunately, the syntax, semantics and proof system of DC have been encoded in PVS [8] as a PVS based proof checker for Duration Calculus called PVS/DC$^-$ which is well suitable for our purpose.

The paper is organised as follows. In Section 2, we present our model of communication protocols and their specification in PVS. A formal specification of the Biphase Mark Protocols is presented in Section 3. The formal verification of the protocols is given in Section 4. Section 5 is for our discussion about optimal design of BMP.

2 Modelling Communication Protocols in Duration Calculus

In this section, we present a model for the communication protocols using Duration Calculus (DC).

Consider a model for communication at the physical layer (see Fig. 1). A sender and a receiver are connected via a bus. Their clocks are running at different rates. We refer to the clock of the receiver as the time reference. The receiver receives signals by digitising. Since the signals sent by the sender and the signals received by the receiver are functions from the set (R) of nonnegative real numbers (representing time) to $\{0, 1\}$ (1 represents that the signal is *high*, and 0 represents that the signal is *low*), we can consider them as boolean functions of time, which are called states in DC.

Duration Calculus [2], introduced by Zhou et al, is a logic for reasoning about the behaviour of the states. This makes it really suited for modelling the communication protocols. For the convenience of readers who

*On leave from the Institute of Information Technology, Nghia Do, Tu Liem, Hanoi, Vietnam.

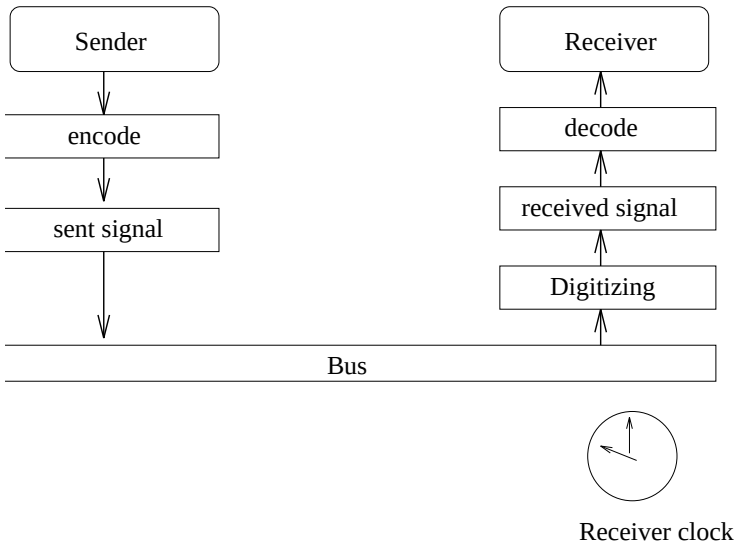

Figure 1: Communication Protocol Model

are not familiar with DC, we give a brief description of the class of DC formulas that are used in this paper. For more details of DC, readers are referred to [2].

Let S denote the set of states (boolean functions of time). For a state $X \in S$ the state we denote $\neg X$ the boolean function $1 - X$. A formula of DC is evaluated over an interval of time. Let ℓ denote the length of the particular interval. In this paper we are interested in the DC formulas generated by the following grammar:

$$D \mathrel{\widehat{=}} \lceil X \rceil \mid \mathit{int\_intv} \mid \ell \sim c \mid D^\frown D \mid D \Rightarrow D \mid D \wedge D$$

where X stands for a state, c stands for a real number and $\sim$ is any operator from the set $\{<, \leq, >, \geq, =\}$. The meaning of the formulas is described as follows. Let an interval $[a, b]$ be given. For a state X, the formula $\lceil X \rceil$ is true for $[a, b]$ iff $X(t) = 1$ for all $t \in (a, b)$. The formula $\mathit{int\_intv}$ is true for $[a, b]$ iff $a, b \in \mathsf{N}$ (the set of natural numbers). The formula $\ell \sim c$ is true for $[a, b]$ iff $b - a \sim c$. The formula $D^\frown D'$ is true for $[a, b]$ iff D is true for $[a, m]$ and D' is true for $[m, b]$ for some $m \in [a, b]$. The meaning of the remaining formulas are the same as in propositional logic. Note that the formula $\mathit{int\_intv}$ was introduced in [6] to express the intervals between ticks of the reference clock, i.e. the intervals with integers as their starting and ending points. We say that a formula A is valid, denoted by $\models A$, iff A is true for all intervals and for all interpretations of the states. The DC formula that cannot be satisfied by an interval is denoted by ff (e.g. $\ell < 0$).

PVS theories for Duration Calculus which encode its syntax, semantics and proof system in PVS, have been developed in [8]. They can be used to prove the validity of DC formulas in PVS. In these theories, a formula of DC is declared to be of the type `Form`, and some denotations of DC are changed to be more convenient for PVS editor (emacs), e.g. $\models$ is typed in as `|-`, $\wedge$ as `/\`, $^\frown$ as `^` and $\lceil X \rceil$ as `cell(X)`. The theories can be imported by the PVS command `IMPORTING int_intv_add_rules[TV]`.

Now, the communication protocols are modelled in DC as follows. The signal sent by the sender is modelled by a state XX. The signal received by the receiver by digitising the signal on the bus is modelled by a state YY. By digitising we mean that when the receiver detects that YY is 1 (or 0), it considers that YY is 1 (or 0) for the whole next cycle of its clock ([3]).Thus, YY can change only at the clock ticks and is right-continuous, and hence has no isolated point. More precisely, YY is a state satisfying the following DC formulas

$$\begin{aligned}
&(\mathit{int_intv} \wedge (\ell = 1)) \Rightarrow (\lceil YY \rceil \vee \lceil \neg YY \rceil)\,, \\
&(\lceil YY \rceil^\frown \lceil YY \rceil) \Rightarrow \lceil YY \rceil\,, \\
&(\lceil \neg YY \rceil^\frown \lceil \neg YY \rceil) \Rightarrow \lceil \neg YY \rceil\,.
\end{aligned}$$

Now, we model the relationship between XX and YY. Without loss of generality, assume that the delay between the sender and the receiver is 0. Due to the fact that it takes a significant amount of time to change the signal on the bus from high to low or vice-versa, the signal on the bus cannot be represented by a boolean function. When the signal on the bus is neither high nor low, the receiver will choose an arbitrary

value from $\{0,1\}$ for the value of YY [7]. The phenomenon is depicted in Fig. 2. Assume that it takes RR (RR is a natural number) receiver-clock cycles for the sender to change the signal on the bus from high to low or vice-versa. Then if the sender changes the signal from low to high or from high to low, the receiver's signal will be unreliable for RR cycles starting from the last tick of the receiver clock and during this period it can be any value chosen nondeterministically from 0 and 1. Otherwise, the signal received by the receiver is the same as the signal sent by the sender (see Figure 2). To write a DC formula for this assumption, we observe the state XX in an interval with length longer than $RR+1$ cycles. Assume that XX is stable for the whole interval. Then we can guarantee that after the first $(RR-1)$ ticks inside the interval YY is the same as XX until the last tick in the interval since there may be a state change of XX just before or just after the interval. Hence, the assumption can be written precisely as

$$\begin{array}{l}(\lceil XX\rceil \wedge (\ell \geq RR+1)) \Rightarrow \\ (\ell \leq RR)^\frown(\lceil YY\rceil \wedge int\_intv)^\frown(\ell < 1)\,, \\ (\lceil \neg XX\rceil \wedge (\ell \geq RR+1)) \Rightarrow \\ (\ell \leq RR)^\frown(\lceil \neg YY\rceil \wedge int\_intv)^\frown(\ell < 1)\,.\end{array}$$

The assumption on the behaviour of XX and YY is now written as the PVS theory `protocolmodel` in Fig. 3. In the theory, `SEXP` is the data type consisting of all boolean functions (mappings from *time* to $\{0,1\}$).

Since the behaviour of a state can be specified by a DC formula, a communication protocol can be modelled as consisting of a coding function f, which maps a sequence of bits to a DC formula expressing the behaviour of XX, and a decoding function g, which maps a DC formula expressing the behaviour of YY to a sequence of bits. The protocol is correct iff for any sequence w of bits, if the sender put the signal represented by $f(w)$ on the bus then by digitising the receiver must receive and receives only the signals represented by a DC formula D for which $g(D) = w$. It is not difficult to write down a general theory in PVS/DC to specify the general protocol. We do not write it here; instead, we write a theory to specify the Biphase Mark protocols in the next section.

3 Biphase Mark Protocols

In the Biphase Mark Protocols (BMP) the sender encodes a bit as a cell consisting of a mark subcell of length RB and a code subcell of length RA. The sender keeps the signal stable in each subcell (hence either $\lceil XX\rceil$ or $\lceil \neg XX\rceil$ holds for the interval representing a subcell). For a cell, if the signal in the mark subcell is the same as the signal in the code subcell, the information carried by the cell is 0; otherwise, the information carried by the cell is 1. There is a phase reverse between two consecutive cells. This means that for a cell the signal of the mark subcell of the following cell is held as the negation of the signal of the code subcell of the cell. The receiver, on detecting a state change (of YY), knows that it is the beginning of a cell, and skips RD cycles (called the *sampling distance*) and samples the signal. If the sampled signal is the same as the signal at the beginning of the cell, it decodes the cell as 0; otherwise it decodes the cell as 1. Fig. 4 illustrates sending and receiving the sequence $w = 01$.

At the beginning of the transmission, the signal is low for RA cycles (this means, $\lceil \neg XX\rceil$ holds for the interval of length RA starting from the beginning). When the sender finishes sending, it keeps the signal stable for RC time units which is longer than the code subcell. We use HLS, LHS to denote the formulas representing intervals consisting of the code subcell of a cell and the mark subcell of the next one for the sender, and use $HLR^\frown(\ell = RD)$, $LHR^\frown(\ell = RD)$ to denote the formulas representing the intervals between the two consecutive sampling points (from the time the receiver samples the signal of a code subcell to the next one, see Fig. 4). Formally,

$$\begin{array}{lcl} HLS & \widehat{=} & (\lceil X\rceil \wedge \ell = RA)^\frown(\lceil \neg X\rceil \wedge \ell = RB)\,, \\ LHS & \widehat{=} & (\lceil \neg X\rceil \wedge \ell = RA)^\frown(\lceil X\rceil \wedge \ell = RB)\,, \\ HLR & \widehat{=} & (\lceil Y\rceil \wedge int \wedge 1 \leq \ell \leq CC)^\frown \\ & & (\lceil \neg Y\rceil \wedge \ell = 1)\,, \\ LHR & \widehat{=} & (\lceil \neg Y\rceil \wedge int \wedge 1 \leq \ell \leq CC)^\frown \\ & & (\lceil Y\rceil \wedge \ell = 1)\,. \end{array}$$

Now, we are ready to formalise the BMP in DC. What we have to do is write down the encoding function f and the decoding function g. From the informal description of the protocol, we can define f inductively as follows.

1. $f(\epsilon) \widehat{=} (\lceil \neg X\rceil \wedge \ell = RC)$
2. If $f(w) = D^\frown(\lceil X\rceil \wedge \ell = RC)$, then
$$\begin{array}{lcl} f(w0) & \widehat{=} & D^\frown HLS^\frown(\lceil \neg X\rceil \wedge \ell = RC) \\ f(w1) & \widehat{=} & D^\frown HLS^\frown(\lceil X\rceil \wedge \ell = RC) \end{array}$$
3. If $f(w) = D^\frown(\lceil \neg X\rceil \wedge \ell = RC)$, then
$$\begin{array}{lcl} f(w0) & \widehat{=} & D^\frown LHS^\frown(\lceil X\rceil \wedge \ell = RC) \\ f(w1) & \widehat{=} & D^\frown LHS^\frown(\lceil \neg X\rceil \wedge \ell = RC) \end{array}$$

For example, $f(1) = LHS^\frown(\lceil \neg X\rceil \wedge \ell = RC)$, $f(10) = LHS^\frown LHS^\frown(\lceil X\rceil \wedge \ell = RC)$, and $f(101) = LHS^\frown LHS^\frown HLS^\frown(\lceil X\rceil \wedge \ell = RC)$.

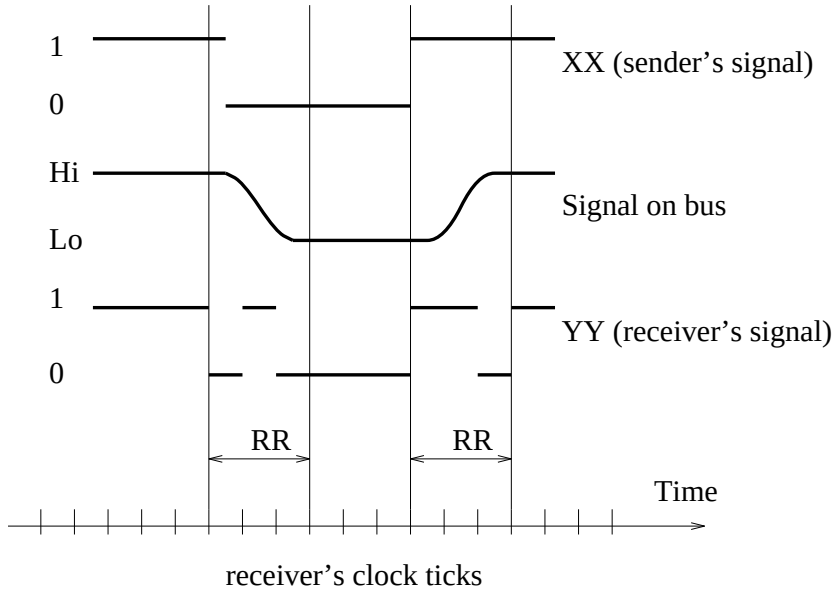

Figure 2: Signal patterns

```
protocolmodel[TV: TYPE+]: THEORY
BEGIN
     IMPORTING int_intv_add_rules[TV]

     XX, YY                   : SEXP;
     RR                       : nat;

     receive0: AXIOM |- ((cell (YY) ^ cell (YY)) => cell (YY)) AND
                     |- ((cell (NOT YY) ^ cell (NOT YY)) => cell (NOT YY));
     receive1: AXIOM |- ((int_intv /\ (l = 1)) => (cell (YY) \/ cell (NOT YY)));
     receive2: AXIOM RR >= 1 IMPLIES |- (((l >= RR + 1) /\ cell(XX)) =>
                              ((l <= RR) ^ (int_intv /\ cell(YY)) ^ (l < 1)));
     receive3: AXIOM RR >= 1 IMPLIES |- (((l >= RR + 1) /\ cell(NOT XX)) =>
                              ((l <= RR) ^ (int_intv /\ cell(NOT YY)) ^ (l < 1)));
END protocolmodel
```

Figure 3: Theory protocolmodel

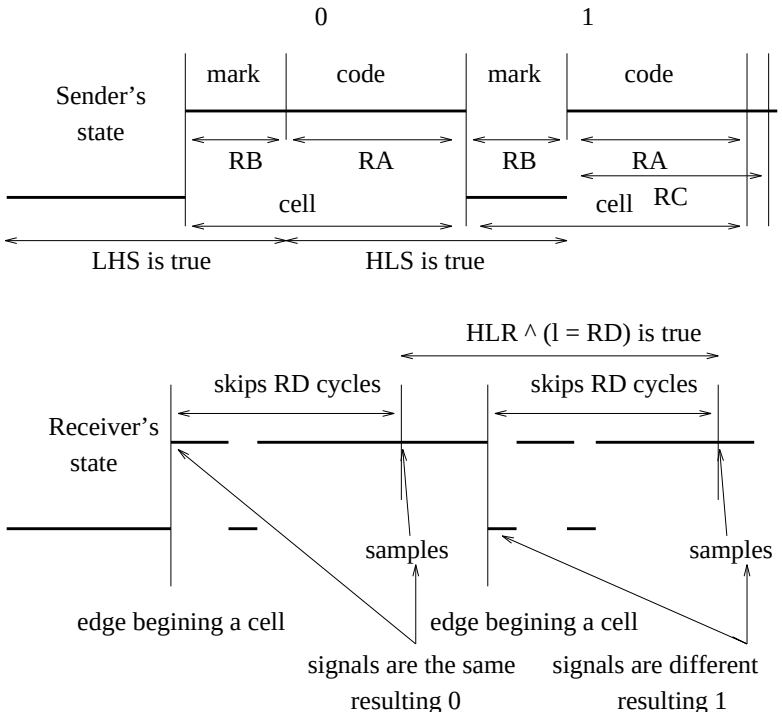

Figure 4: Biphase Mark Protocol

Because the decoding function g is a partial function, we have to describe its domain first, i.e. what kind of DC formulas on the state YY are detected (received) by the receiver. According to the behaviour of the receiver, first it skips RR cycles. Then it begins to scan for an edge (HLR or LHR). When an edge is detected, it skips RD cycles and repeats this procedure until it detects that the transmission has completed (YY is stable for more than CC cycles). Thus, a DC formula D is received by the receiver iff D is of the form $A_0{}^\frown A_1{}^\frown \ldots {}^\frown A_n$, $n \geq 1$, where

- $A_0 = (1 \geq \ell \wedge \ell > 0)^\frown(int\_intv \wedge (\ell = RR - 1)))$
- and either
  $A_n = (int\_intv \wedge \lceil \neg YY \rceil \wedge (\ell > CC))^\frown(\ell < 1))$,
  or $A_n = (int\_intv \wedge \lceil \neg YY \rceil \wedge (\ell > CC))^\frown(\ell < 1))$
- and for $j = 1, \ldots, n-1$ either $A_j = LHR^\frown(\ell = RD)$ or $A_j = HLR^\frown(\ell = RD)$
- and if $n = 1$ then $A_n = (int\_intv \wedge \lceil \neg YY \rceil \wedge (\ell > CC))^\frown(\ell < 1))$ and if $n > 1$ then $A_1 = LHR^\frown(\ell = RD)$ (since at the beginning the signal is low).

Now, the decoding function g can be written as follows. Let D be a formula received by the receiver.

- If $D = (\ell \leq 1 \wedge \ell > 0)^\frown(int\_intv \wedge \ell = RR - 1)^\frown(\lceil \neg YY \rceil \wedge \ell > CC \wedge int\_intv)^\frown\ell < 1$ then $g(D) = \epsilon$.
- Let $g(D)$ be defined.

  If $D = D'^\frown(\lceil Y \rceil \wedge int\_intv \wedge \ell \geq CC)^\frown\ell < 1$ then
  $g(D'^\frown HLR^\frown(\ell = RD)^\frown(\lceil Y \rceil \wedge int\_intv \wedge \ell \geq CC)^\frown\ell < 1)$
  $= g(D)1$, and
  $g(D'^\frown HLR^\frown(\ell = RD)^\frown(\lceil \neg Y \rceil \wedge int\_intv \wedge \ell \geq CC)^\frown\ell < 1)$
  $= g(D)0$.

  If $D = D'^\frown(\lceil \neg Y \rceil \wedge int\_intv \wedge \ell \geq CC)^\frown\ell < 1$, then
  $g(D'^\frown LHR^\frown(\ell = RD)^\frown(\lceil Y \rceil \wedge int\_intv \wedge \ell \geq CC)^\frown\ell < 1)$
  $= g(D)0$, and
  $g(D'^\frown LHR^\frown(\ell = RD)^\frown(\lceil \neg Y \rceil \wedge int \wedge \ell \geq CC)^\frown\ell < 1)$
  $= g(D)1$.

For example, let D be $(\ell \leq 1 \wedge \ell > 0)^\frown(int\_intv \wedge \ell = RR - 1)^\frown LHR^\frown(\ell = RD)^\frown LHR^\frown(\ell = RD)^\frown HLR^\frown(\ell = RD)^\frown(\lceil YY \rceil \wedge \ell > CC \wedge$

$int\_intv)^\frown(\ell < 1))$. Then,

$$\begin{aligned} g(D) = \; & g((\ell \leq 1 \wedge \ell > 0)^\frown(int\_intv \wedge \ell = RR - 1) \\ & ^\frown LHR^\frown(\ell = RD)^\frown LHR^\frown(\ell = RD)^\frown \\ & (\lceil YY \rceil \wedge \ell > CC \wedge int\_intv)^\frown(\ell < 1))1 \\ = \; & g((\ell \leq 1 \wedge \ell > 0)^\frown(int\_intv \wedge \ell = RR - 1) \\ & ^\frown LHR^\frown(\ell = RD)^\frown \\ & (\lceil \neg YY \rceil \wedge \ell > CC \wedge int\_intv)^\frown(\ell < 1)) \\ & 01 \\ = \; & g((\ell \leq 1 \wedge \ell > 0)^\frown(int\_intv \wedge \ell = RR - 1) \\ & ^\frown(\lceil \neg YY \rceil \wedge \ell > CC \wedge int\_intv)^\frown \\ & (\ell < 1)) \\ & 101 \\ = \; & \epsilon 101\,. \end{aligned}$$

Now we write a PVS theory to specify the BMP. A finite sequence wa of bits is represented by the pair of its length $length(wa)$ and an array (`[nat -> bit]` in PVS) $content(wa)$. For example, the sequence 101 is represented by the pair $(3, a)$, where 3 is the length of the sequence, and a is an array such that $a(0) = 1$, $a(1) = 0$ and $a(2) = 1$. In order to describe the behaviour of the encoding and decoding algorithm more intuitively, it is convenient to represent the DC formulas resulting from encoding a sequence of bits and from digitising the signal as a sequence of DC formulas connected by the chop operator for which each formula in the sequence corresponds to one step of the algorithms. For example, $f(101)$ can be represented by the sequence of formulas LHS, LHS, HLS, $(\lceil X \rceil \wedge \ell = RC)$, from which $f(101)$ results by connecting them in this order using the operator $^\frown$. Similarly, a formula D received by the receiver can be represented by the sequence $A_0, A_1, \ldots, A_n$, where $D = A_0{}^\frown A_1{}^\frown \ldots {}^\frown A_n$ as in the definition of the receivable formulas. Thus, instead of writing f we write the function `encode` that maps a sequence of bits wa to the sequence of formulas representing $f(wa)$, and instead of writing g we write the function `decode` that maps the sequence of formulas representing a received formula D to the sequence of bits $g(D)$. Since f and g are defined inductively, the functions `encode` and `decode` are defined as recursive functions of sequences. For example, `encode(wa)` is defined via `encode(w)`, where `w` is the greatest prefix of `wa`. Note that `wa` is represented by the pair `(length(wa),content(wa))`. Thus, `w` is represented by the pair `(length(wa) - 1,content(wa))`. In order to represent the domain of the function `decode`, we introduce the predicate `received?` which is true for a sequence of DC formulas iff the sequence represents a receivable formula.

Now, a theory to expressing the BMP in PVS/DC$^-$ is written as the theory `biphase` in Fig. 5 and Fig. 6.

4 Verification of BMP

According to our model of protocols described in Section 2, we have to verify that for any sequence of bits wa, if the sender puts on the bus the signal represented by DC formula $f(wa)$, then the receiver must receive and receives only the signals represented by a DC formula D for which $g(D) = wa$. Naturally, we can only prove this requirement with some condition on the values of the parameter RR, RA, RB, RC, CC, RD and RR. The requirement is formalised as:

For all sequence of bits wa,

- there exists a DC formula D received by the receiver such that $f(wa) \Rightarrow D$, and
- for all D receivable by the receiver, if $f(wa) \Rightarrow D$ then $g(D) = wa$.

Since in BMP g is a deterministic function, for any sequence of bits wa there is no more than one receivable formula D for which $f(wa) \Rightarrow D$. Thus we can have a stronger requirement which is formalised as:

For all sequences of bits wa there exists uniquely a receivable formula D such that $f(wa) \Rightarrow D$ and $g(D) = wa$.

Our verification is done by proving the following two theorems.

Theorem 1 *For any receivable formulas D and D', if D is different from D' syntactically then $\models ((D \wedge D') \Rightarrow ff)$.*

Theorem 2 *Assume that $RR \geq 1$, $RB \geq RR + 1$, $RA \geq RR + 1$, $RC \geq CC + RA$, $RD \geq RB + RR$, $RD \leq RA + RB - 3 - RR$, and $CC \geq RA + 1$. Then for any sequence of bits wa there exists a receivable formula D for which $f(wa) \Rightarrow D$ and $g(D) = wa$.*

In order to prove these theorems using PVS, we specify them as a theory in PVS.

Remember that in the theory `biphase` we have represented a receivable formula D as a sequence of DC formulas `cc` for which `received?(cc)` is true. Note that we have also defined in the theory the function `send_signal`, which is exactly the function f, and the function `receive_signal`, which for a given sequence of DC formulas representing a receivable formula returns the formula itself. With these in mind, we write a theory to formalise our requirements as the theory `correctness` given in Fig. 7.

In the sequel, we give a brief description of how the proof has been done by using the PVS proof checker.

The theorem `correct_1` is proved through several steps.

```
biphase[TV: TYPE+] : THEORY
 BEGIN
  IMPORTING protocolmodel[TV]

     RA, RB, RC     : Value;
     m, k, j        : VAR nat;
     CC, RD         : nat;
     bit            : TYPE = {x : nat | 0 <= x AND x <= 1} CONTAINING 0;
     info           : TYPE = [ nat, ARRAY[nat -> bit] ];
     send_receive   : TYPE = [ nat, ARRAY[nat -> Form] ];
     word           : VAR info;
     empty_word     : info = (0, LAMBDA j: 0)
     codding        : VAR send_receive;
     n(codding)     : nat = proj_1(codding);
     a(codding)     : ARRAY[nat -> Form] = proj_2(codding);
     length(word)   : nat = proj_1(word);
     content(word)  : ARRAY[nat -> bit] = proj_2(word);
  % abreviations of the formulas expressing state changes
     LHS    : Form = (cell(NOT XX) /\ (l = RA)) ^ (cell(XX) /\ (l = RB));
     HLS    : Form = (cell(XX) /\ (l = RA)) ^ (cell(NOT XX) /\ (l = RB));
     TT     : ARRAY[nat -> Form] = (LAMBDA j: (l = 0))
     LHR    : Form = (int_intv /\ cell(NOT YY) /\ (l >= 1) /\ (l <= CC)) ^
                      (int_intv /\ cell(YY) /\ (l = 1));
     HLR    : Form = (int_intv /\ cell(YY) /\ (l >= 1) /\ (l <= CC)) ^
                      (int_intv /\ cell(NOT YY) /\ (l = 1));
  % encoding function: the sender encodes sequences of bits into a
  % sequence of formulas expressing the sent signal. Note that for
  % word: info, (length(word)-1,content(word)) is the greatest proper
  % prefix of word.
  encode(word): RECURSIVE send_receive =
    (IF length(word) = 0 THEN (0, (TT WITH [(0) := (cell(NOT XX) /\ (l = RC))]))
     ELSIF a(encode(length(word) - 1, content(word)))
          (n(encode(length(word) - 1, content(word)))) =  (cell(NOT XX) /\ (l = RC))
     THEN IF content(word)(length(word) - 1) = 0
          THEN (n(encode(length(word) - 1, content(word))) + 1,
               (a(encode(length(word) - 1, content(word))) WITH
                [(n(encode(length(word) - 1, content(word)))) := LHS] WITH
                [(n(encode(length(word) - 1, content(word))) + 1) :=
                                              (cell(XX) /\ (l = RC))]))
          ELSE (n(encode(length(word) - 1, content(word))) + 1,
               (a(encode(length(word) - 1, content(word))) WITH
                [(n(encode(length(word) - 1, content(word)))) := LHS] WITH
                [(n(encode(length(word) - 1, content(word))) + 1) :=
                                             (cell(NOT XX) /\ (l = RC))]))
          ENDIF
     ELSIF content(word)(length(word) - 1) = 0
     THEN (n(encode(length(word) - 1, content(word))) + 1,
          (a(encode(length(word) - 1, content(word))) WITH
           [(n(encode(length(word) - 1, content(word)))) := HLS] WITH
           [(n(encode(length(word) - 1, content(word))) + 1) :=
                                            (cell(NOT XX) /\ (l = RC))]))
     ELSE (n(encode(length(word) - 1, content(word))) + 1,
          (a(encode(length(word) - 1, content(word))) WITH
           [(n(encode(length(word) - 1, content(word)))) := HLS] WITH
           [(n(encode(length(word) - 1, content(word))) + 1) :=
                                              (cell(XX) /\ (l = RC))]))
     ENDIF)
     MEASURE (LAMBDA (word: info): length(word));
```

Figure 5: Theory biphase

```
% Decoding function: the receiver decodes the received signals into a sequence of bits.
% The receiver can receive only the signals expressing by the predicate received?
received?(codding): bool = ((n(codding) >= 1)
    AND (a(codding)(0) = (1 >= l /\ l > 0) ^  (int_intv /\ (l = RR - 1)))
    AND ((a(codding)(n(codding)) = (int_intv /\ cell(NOT YY) /\ (l > CC)) ^ (l < 1))
        OR (a(codding)(n(codding)) = (int_intv /\ cell(YY) /\ (l > CC)) ^ (l < 1)))
    AND (FORALL j: ((j > 1 AND j < n(codding)) IMPLIES
         (a(codding)(j) = LHR ^ (l = RD) OR a(codding)(j) = HLR ^ (l = RD))))
    AND (FORALL j: (j > n(codding) IMPLIES (a(codding)(j) = (l = 0))))
    AND (n(codding) = 1 IMPLIES a(codding)(n(codding)) =
                               (int_intv /\ cell(NOT YY) /\ (l > CC)) ^ (l < 1))
    AND (n(codding) > 1 IMPLIES a(codding)(1) =  LHR ^ (l = RD)));

decode(cc: (received?)): RECURSIVE info =
    (IF n(cc) = 1 THEN empty_word
     ELSIF a(cc)(n(cc) - 1) = LHR ^ (l = RD)
     THEN IF (a(cc)(n(cc)) = (int_intv /\ cell(YY) /\ (l > CC)) ^ (l < 1))
          THEN  (n(cc) - 1, (content(decode(n(cc) - 1, a(cc) WITH
                 [(n(cc) - 1) := (int_intv /\ cell(NOT YY) /\ (l > CC)) ^
                                  (l < 1)] WITH [(n(cc)) := (l = 0)]))
                 WITH [(n(cc) - 2) := 0]))
          ELSE (n(cc) - 1, (content(decode(n(cc) - 1, a(cc) WITH
                 [(n(cc) - 1) := (int_intv /\ cell(NOT YY) /\ (l > CC)) ^
                                  (l < 1)] WITH [(n(cc)) := (l = 0)]))
                 WITH [(n(cc) - 2) := 1]))
          ENDIF
     ELSIF (a(cc)(n(cc)) = (int_intv /\ cell(YY) /\ (l > CC)) ^ (l < 1))
     THEN (n(cc) - 1, (content(decode(n(cc) - 1, a(cc) WITH
            [(n(cc) - 1) := (int_intv /\ cell(YY) /\ (l > CC)) ^
                             (l < 1)] WITH [(n(cc)) := (l = 0)]))
           WITH [(n(cc) - 2) := 1]))
     ELSE (n(cc) - 1, (content(decode(n(cc) - 1, a(cc) WITH
            [(n(cc) - 1) := (int_intv /\ cell(YY) /\ (l > CC)) ^
                            (l < 1)] WITH [(n(cc)) := (l = 0)]))
           WITH [(n(cc) - 2) := 0]))
     ENDIF)
     MEASURE (LAMBDA (cc: (received?)): n(cc));

   firstpart(codding,k): send_receive =  (IF k = 0 THEN (0,TT)
                ELSIF k > n(codding) THEN codding ELSE (k - 1, a(codding)) ENDIF);
   lastpart(codding,k): send_receive =  (IF k = 0 THEN codding
                ELSIF k > n(codding) THEN (0,TT)
                ELSE (n(codding) - k, (LAMBDA j: a(codding)(j + k))) ENDIF);
   tail(codding): send_receive = lastpart(codding,1);
   init(codding): send_receive = IF n(codding) = 0 THEN (0,TT)
          ELSE (n(firstpart(codding,n(codding))), a(firstpart(codding,n(codding)))
                WITH [(n(codding)) := (l = 0)]) ENDIF;

   signal_suffix(codding): RECURSIVE Form =
           (IF n(codding) = 0 THEN a(codding)(n(codding))
           ELSE signal_suffix(init(codding)) ^ a(codding)(n(codding)) ENDIF)
          MEASURE (LAMBDA codding: n(codding));
   signal_prefix(codding): RECURSIVE Form =
          (IF n(codding) = 0 THEN a(codding)(n(codding))
           ELSE a(codding)(0) ^ signal_prefix(tail(codding)) ENDIF)
          MEASURE (LAMBDA codding: n(codding));

% send_signal/receive_signal: formulas express the state for the sent/received signal
   send_signal(word): Form = signal_suffix(encode(word));
   receive_signal(cc:(received?)): Form = signal_suffix(cc);
END biphase
```

Figure 6: Theory biphase (contd)

```
correctness[TV: TYPE+] : THEORY
  BEGIN
   IMPORTING biphase[TV]

     m,k,j,i : VAR nat
     wa, wb : VAR info
     cc1, cc2, cc: VAR (received?)

     same?(wa,wb): bool = ((length(wa) = length(wb) AND
                     FORALL j: (j < length(wa)
                       IMPLIES content(wa)(j) = content(wb)(j))));


     break1: LEMMA (RR >= 1 AND RB >= RR + 1 AND RA >= RR + 1 AND
                    RC >= CC + RA AND RD >= RB + RR AND RD <= RA + RB - 3 - RR
                    AND CC >= RA + 1)
             IMPLIES
               (length(wa) = m IMPLIES
                 (((a(encode(wa))(n(encode(wa))) = (cell(XX) /\ (l = RC)))
                   IMPLIES (EXISTS cc :
                     (same?(decode(cc),wa) AND
                       (a(cc)(n(cc)) =
                       (int_intv /\ cell(YY) /\ (l > CC)) ^ (l < 1)) AND
                       |- (send_signal(wa) =>
                           receive_signal(cc)) AND
                       |- ((signal_suffix(init(encode(wa))) ^
                                     (cell(XX) /\ (l = RA))) =>
                          (signal_suffix(init(cc)) ^
                           (int_intv /\ cell(YY) /\
                                        (l > RA + RB - 3 - RR - RD)
                                      /\ (l <= RA - RR + 1)) ^ (l < 1))))))
                  AND
                  ((a(encode(wa))(n(encode(wa))) =
                                              (cell(NOT XX) /\ (l = RC)))
                   IMPLIES (EXISTS cc :
                     (same?(decode(cc),wa) AND
                       a(cc)(n(cc)) =
                       (int_intv /\ cell(NOT YY) /\ (l > CC)) ^ (l < 1) AND
                       |- (send_signal(wa) =>
                           receive_signal(cc)) AND
                       |- ((signal_suffix(init(encode(wa))) ^
                                     (cell(NOT XX) /\ (l = RA))) =>
                          (signal_suffix(init(cc)) ^
                           (int_intv /\ cell(NOT YY) /\
                                       (l > RA + RB - 3 - RR - RD) /\
                                       (l <= RA - RR + 1)) ^ (l < 1))))))));


     correct_1: THEOREM (cc1 /= cc2 AND CC >= RA + RR + 1) IMPLIES
                 |- ((receive_signal(cc1) /\ receive_signal(cc2)) => ff);

     correct_2: THEOREM   (RR >= 1 AND RB >= RR + 1 AND RA >= RR + 1
                           AND RC >= CC + RA + 1  AND RD >= RB + RR AND
                           RD <= RA + RB - 3 - RR AND CC >= RA + RR + 1)
                         IMPLIES
                           (EXISTS cc: same?(decode(cc),wa) AND
                           |- (send_signal(wa) => receive_signal(cc)));
 END correctness
```

Figure 7: Theory correctness

At the first step, we prove that for finite sequences $cc1$ and $cc2$ of DC formulas, $cc1 \neq cc2$, there exists a position k such that the kth element of $cc1$ and the kth element of $cc2$ are different, and $cc1$ and $cc2$ are the same at all positions before k.

At the second step, we prove that

$$(receive\_signal(cc1) \wedge receive\_signal(cc2)) \Rightarrow \\ signal\_prefix(firstpart(cc1, k))^\frown \\ (signal\_suffix(lastpart(cc1, k)) \wedge \\ signal\_suffix(lastpart(cc2, k)))\,,$$

where for a sequence $cc = A_0, A_1, \ldots, A_n$ of DC formulas $signal\_prefix(cc)$ is the chop of the formulas in the sequence in the order associated to the right and $signal\_suffix(cc)$ is the chop of the formulas in the sequence in the order associated to the left (the two are equivalent but the current version of PVS/DC does not recognise this automatically, so we have to prove the equivalence by induction on the length of the sequence), and $firstpart(cc, k)$ is the sequence $A_0, A_1, \ldots, A_{k-1}$ and $lastpart(cc, k)$ is the sequence $A_k, , \ldots, A_n$.

At the last step, we prove that if $cc1$ and $cc2$ are receivable, and if A and B are, respectively, the formulas at the kth position of $cc1$ and $cc2$, and $A \neq B$ then $(A \wedge B) \Rightarrow ff$.

The theorem `correct_2` is a weak version of the lemma `break1`. Hence, its proof is carried out easily by using the lemma `break1` with a suitable instantiation. The proof of the lemma `break1` is by induction on the length of wa using different instantiations of some auxiliary lemmas. In order to reason about inequalities, we use the definition of DC semantics to convert a DC formula into its semantics in high order logics to take advantage of the distinct features of PVS. But this method works only for simple DC formulas. If the formulas are rather complicated (with more than about 4 occurrences of the modality $^\frown$), we cannot translate them into higher order logic formulas since in this case, the resulting formulas are so big that the computer gets stuck. Thus, we have to use the DC proof rules to simplify them or split them into smaller ones before using the method. One of the difficulties in doing proof in the current version of PVS/DC$^-$ is that the tool cannot recognise automatically the equivalence of two formulas that are different only because of the different associations of the $^\frown$ operator.

In spite of these difficulties, we successfully verified the correctness of BMP with the PVS/DC$^-$ tool. The total run time of the proof when rerunning them is about five hours. Particularly, when rerunning the proof of the lemma `break1`, the runtime is almost one hour and the real time is almost two hours.

5 Reasoning about the Clock-Rate Difference and Optimal Designs

In this section, we use the results of the previous section to reason about the difference between the rates of the receiver clock and the sender clock. We also discuss the optimal value of the parameters for a design. This shows the advantages of our approach in comparison to others.

We have proved that given $RR \geq 1$, the protocol is correct if

$$\begin{aligned} &RB \geq RR + 1, \\ &RA \geq RR + 1, \\ &RC \geq CC + RA, \\ &RD \geq RB + RR, \\ &RD \leq RA + RB - 3 - RR, \\ &CC \geq RA + 1, \end{aligned}$$

where RR, RD, CC are natural numbers, and RB, RC, RA are non negative real numbers. All of them represent periods of time referring to the receiver clock. Since the sender uses its own clock, RB, RC, RA should be given according to the sender clock (we recall that RB, RC, RA are the parameters for decoding). Let ρ be the ratio between the clock rate of the receiver and the sender, and RB, RC, RA be given according to the sender clock. Then, their values using the receiver clock as the time reference are $RB * \rho, RC * \rho, RA * \rho$. For the protocol to be correct, it should be the case that

$$\begin{aligned} &RB * \rho \geq RR + 1, \\ &RA * \rho \geq RR + 1, \\ &RC * \rho \geq CC + RA, \\ &RD \geq RB * \rho + RR, \\ &RD \leq (RA + RB) * \rho - 3 - RR, \\ &CC \geq RA * \rho + 1\,. \end{aligned}$$

Thus, ρ should satisfy

$$\begin{aligned} &\rho \geq \max\{(RR + 1)/RB, (RR + 1)/RA, \\ &(CC + RA)/RC, \\ &(RD + 3 + RR)/(RA + RB)\}\,, \\ &\rho \leq \min\{(RD - RR)/RB, (CC - 1)/RA\}\,. \end{aligned}$$

Hence, given a value of RR, RA, RB, RC, CC, RD, we can decide the allowed range for the ratio of the clock rates of the receiver and the sender. We calculate the range for the ratio of the rates of the receiver and sender clocks for the conventional and unconventional choices of the parameters mentioned in [9].

For the conventional choice, $RR = 1$, $RA = RB = 16$, and $RD = 23$. Note that RC and CC do not play an important role in the choice: given RA, RR, RD and ρ that satisfy the above inequalities, we can always calculate RC and CC such that all of them satisfy the above inequalities. Thus, we remove the expressions in which there is an occurrence of CC and RC in the constraints for ρ. Thus,

$$\rho \geq \max\{2/16, 27/32\} = 27/32$$
$$\rho \leq \min\{22/16\} = 22/16\,.$$

For the unconventional choice, $RR = 1$, $RA = 13$, $RB = 5$, and $RD = 10$. Similarly to the previous case,

$$\rho \geq \max\{2/5, 2/13, 14/18\} = 14/18$$
$$\rho \leq \min\{9/5\} = 9/5\,.$$

Thus, the range for the ratio between the clock rates for the unconventional case is larger than for the conventional one, which has been remarked in [9].

For the transmission to be fast, RA and RB should be as small as possible. Let us choose $RB = RR + 2$ (we could not choose $RB = RR + 1$ since that would require that the clocks should run at the same rate). Therefore, $RD \geq 2(RR + 1)$ and $RD \leq RA - 1$. Let us choose $RD = 2RR + 3$ and $RA = 2RR + 5$. Thus, if $RR = 2$ then $RB = 4$, $RD = 7$ and $RA = 9$. In this case, ρ is allowed to be between $12/13$ and $5/4$. We can think of this as an optimal option since it allows a large range of the ratio of the clock rates and also allows the transmission to be fast.

6 Conclusion

We have presented our approach to the specification and verification of the Biphase Mark Protocols using Duration Calculus and its associated tool. We have also shown that, the fact that DC is based on boolean state functions and time intervals makes it particularly convenient for modelling and verifying the communication protocols since a signal can be written easily as a DC formula. To our knowledge, the model of BMP presented in this paper is more general and more intuitive than the others. We compare our method and the method in [9] for example. The method in [9] works only on a fixed value of the length of the code subcell and mark subcell, on a fixed value of sampling distance (in fact, they verify for the unconventional choice only) and on the assumption that it takes one cycle of the sender clock for the sender to change the signal on bus. In contrast, our method works for general parametters. Apart from this, we are also able to reason about the different clock rates of different components in the system, which is extremely useful in distributed systems. In our model of communication protocols, the verification can be done using classical and familiar techniques like induction, which does not require a deep knowledge about the calculus.

The modelling and verification technique used in this paper can similarly be applied to other protocols such as Audio Control Protocols as well.

References

[1] D. Bosscher et al. "Verification of an Audio Control Protocol", *LNCS 863*, 1994, pp. 170-192.

[2] Z. Chaochen, C. A. R. Hoare, A. P. Ravn, "A calculus of durations", *Information Processing Letters*, 40, 1992, 269-276.

[3] T. Henzinger, Z. Manna, and A. Pnueli, "What Good are Digital Clocks?", Springer-Verlag, LNCS 623, 1992, pp.

[4] D.V. Hung and W. Ji, "On the design of hybrid control systems using I/O automata", V. Chandru and V. Vinay (Eds.) *Foundations of Software Technology and Theoretical Computer Science* (FST&TCS16), LNCS 1180, Springer-Verlag, Dec 1996, pp. 156–167.

[5] D.V. Hung and P.H. Giang, "A Sampling Semantics of Duration Calculus", Bengt Jonsson and Joachim Parrow (Eds), *Formal Techniques for Real-Time and Fault Tolerant Systems*, LNCS 1135, Spriger-Verlag, pp. 188–207, 1996.

[6] D.V. Hung and Ko Kwang Il, "Verification via Digitized Models of Real-Time Hybrid Systems", *Asia-Pacific Software Engineering Conference 1996* (APSEC'96), IEEE Computer Society Press, 1996, pp 4–15.

[7] B.P. Mahony and I.J. Hayes, "Using continuous real functions to model timed histories", in *Proc. 6th Australian Software Engineering Conf. (ASWEC91)*, 1991.

[8] Mao Xiaoguang, Xu Qiwen and Wang Ji, "Towards a proof Assistant for Interval Based Logics", UNU/IIST report 77, July 1996.

[9] J. S. Moore "A formal model of asynchronous communication and its use in mechanically verifying a Biphase Mark Protocol", *Formal Aspects of Computing*, 6, 1994, pp. 60-91.

[10] B. Moszkowski, "A temporal Logic for Multilevel Reasoning about Hardware", *IEEE Computer*, Vol. 18, No. 2, 1985, pp. 10-19.