Formal Verification of Serial Port Module in Robot Control System

Chenhui Lou[1, a *] and Xiaojuan Li[1, b]

[1]Information Engineering College, Capital Normal University, Beijing 100048, China

[a]626617007@qq.com, [b]lixjxxxy@263.net

Keywords: Robot control system; Serial module; Formal verification; Theorem proving

Abstract. As robots used in more and more fields, people are more striated with their safety. As the core of the mobile robot, the reliability of the control system is very important to the whole system. In this paper, a modular design of robot control system architecture is modeled by the xMAS (eXecutable MicroArchitectural Specication) and then verified using ACL2, proving the funtionality correctness. As the formalization of xMAS model in ACL2 is not complete, we first improve the formalization process in ACL2 and then establish xMAS model of the UART serial port module, abstract some key properties and verify them. Combination of the theorem prover ACL2 and xMAS model, which is a great way to solve the verification problem of robot control system, could also provide an effective reference method for the correctness verification of robot control system.

Introduction

Robot control system based on ARM and FPAG architecture is gradually becoming universal design architecture of sized mobile robot because of its advantages such as modular convenience, low power consumption, low cost, small size and good scalability, it has been applied for soccer robot [1], attacking robots, etc. Robot control system provides the basic work platform for mobile robots, reliability of its design is critical to robotic applications, hardware and software failure may cause the robot self-destruct or injury accident. Thus, verification of control system has great practical significance.

As the trend of control system design is becoming modular and hierarchical, paper [2] proposed a kind of modular control system which could be used for multiple robots, it could be customized according to different application environments. In this paper, the author uses simulation method for the functionality test of this system, which shows that the control system can meet the design requirements. But due to the incompleteness of the simulation, the result is not reliable and can‘t fully ensure the reliability of the control system. Formal verification is using mathematical logic to build a mathematical model of system and then prove the necessary attributes to verify the correctness of the design, could be used as a reliable method to verify control system. Theorem proving is one of the methods of formal verification, the basic idea is to use the function to represent the behavior characteristics of the system, and theorem to represent property of the system.

In this paper, we propose using theorem proving method to verify serial port module of robot control system, which is implemented using first-order theorem prover ACL2 [3][4].In addition, we introduce an excellent model tool called xMAS which could be used conveniently to verify other critical modules of the control system.

The first part introduces the definition of xMAS; Second part has a description of serial port module; Formalization of xMAS in ACL2 is placed in third part; In section four, establish xMAS model of serial port module and then verify some key properties of the xMAS model. Section five gives a conclusion and a future research direction.

xMAS Model

xMAS [5] is a new kind of communication paradigm proposed by Intel that could be used for the design and verification of communication fabric, it has a set of elementary components, with all have a well-formed definition in terms of boolean equation. Fig. 1 show the basic components of xMAS.

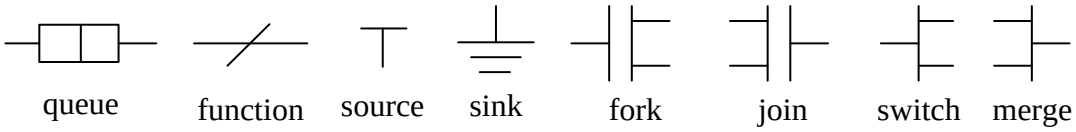

Figure 1. Basic components of xMAS.

Semantic of xMAS components is defined by Boolean equations, due to space limitations, this article only has a summary introduction of components' functionality and features, a detailed definition could refer to reference [5].Fork component has a packet as input and generates two groups, and if and only if one input and two outputs are at the same time in a ready state, should a data transfer, join component finish the reverse functionality, it has two inputs and one output. Merge component imitate the rat arbiter which could have multipul inputs and then select one output packet according to the priority of the input packets, also could choose appropriate arbiter strategy. Switch component is a router which could transfer different packets to different destinations respectively. Source and sink are producers and consumers of data, and the two elements of uncertainty. Function element is a calculation module, you can control the portion of the packet data or make the appropriate changes. Queue element is the only component stores packet, FIFO buffer is the standard sequence. Components are connected through channel, there are three types of channel signal: irdy, trdy and data, these three signals are universal, it means if irdy is true, then it will remain true until trdy is true, then the data transfers. Also, note that both components and channel are typed.

For the system to be verified, we would encode it as a single-clock xMAS network, and then extract the properties of the network and verify.

Serial Port Module

In the robot control system, ARM microprocessor, as the central control center, is responsible for implementing complex control algorithms and interacting with the upper layer host computer. FPGA, as a coprocessor, is responsible for collecting external sensor information, and controlling motor drive. In this article, we would verify PTZ control system which is directly connected to ARM, its responsibility is collecting image information around the robot, Fig. 2 shows a block diagram of the serial port control module:

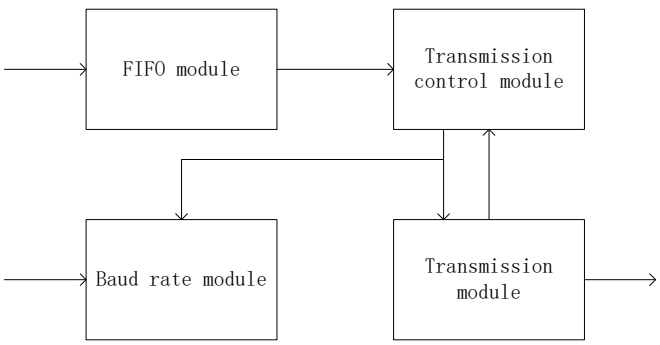

Figure 2. Structure of serial module

From the above chart we can see that the serial module contains four sub-modules, data flow between sub-modules is represented by the directed edge between each other. The format of data transferred is frame contains 11 bits, including one start bit, eleven data bits, one check bit and one end bit.

Baud rate of serial port indicates how many bits of data can be transmitted per second, the system takes the baud rate of 9600bps, generates a high pulse and triggers control module transmitting data one by one every 1/9600 seconds.

First, ARM writes data to be transmitted into FIFO, which has a length of 8 bytes. If the FIFO is full of 8 bytes, the overflow signal is asserted high, notifying upper module could no longer write data. As long as there are data in FIFO, empty signal is set to be low notifying that sending control module could read data from FIFO, if the FIFO is empty, empty signal is high.

Secondly, if the sending module detects empty signal is low, and the tx_done signal denotes whether last transmission has been completed is high, it sends a read signal read_req to read data from FIFO, then sets data transmission enable signal to high and notifies the sending module and baud rate generator module beginning to send data. Only when empty and tx_done are both meeted should data transmission carry out.

Finally, the baud rate generator module detects tx_en signal is high, then generates a BPS_clk signal of 9600bps by a clock divider and sends to the transmitting module, at the same time the sending module reads data temporarily stored in sending control module, and then BPS_clk issues a high level to transfer frame data bit by bit frame data every 1/9600 seconds. As the transmission is finished, the tx_done signal will be setted to high, notifying sending control module that data transmission of last data frame has been completed.

Formalization of xMAS in ACL2

xMAS is higher-order micro-architecture model, because of its high level of abstraction, it could reduce the state space of system effectively, with this advantage, its combination with model checking method could resolve many problems which failed by reason of state explosion. Nowadays, xMAS has been used for deadlock detecting, invariant generation, etc. But as the limitation of model checking, state exploring problem appears when handle extra large networks, so we have a new thought that combining xMAS with theorem proving. Theorem proving would express properties as mathematical formula, deduction by mathematical derivation rules, so it has no problem of state exploring.

```
(defun apply-field-join-and-merge (component param)
    (let ((field (component-field component)))
        (if (endp field)
            nil
            (if (xmas-join-equal (car field) param)
                (cadr field)
                (apply-field-join (cdr field) param)))))

(defun xmas-join-equal (field-item param)
    (if (and (equal (nth 0 field-item) (nth 0 param))
             (equal (nth 1 field-item) (nth 1 param)))
        t
        nil))

(defun apply-field-fork (component param)
    (let ((func (component-field component)))
        (cdr (assoc param func))))
```

Figure 3. Component function

```
((equal flg 'irdy)
    (let*  ((cpt (get-init-component channel ntk))
            (type (component-type cpt))
            (index-out (if (equal (get-in-channel cpt 0 ntk) channel) 0 1))
            (next-unvisited (remove1 (cons channel flg) unvisited)))
    (cond
        ((equal type 'merge)
            (or
                (xmas-transfer-calculate 'irdy (get-in-channel cpt 0 ntk) ntk next-unvisited ntkstate)
                (xmas-transfer-calculate 'irdy (get-in-channel cpt 1 ntk) ntk next-unvisited ntkstate)))
        ((equal type 'join)
            (and
                (xmas-transfer-calculate 'irdy (get-in-channel cpt 0 ntk) ntk next-unvisited ntkstate)
                (xmas-transfer-calculate 'irdy (get-in-channel cpt 1 ntk) ntk next-unvisited ntkstate)))
        ((equal type 'fork)
            (and
                (xmas-transfer-calculate 'irdy (get-in-channel cpt 0 ntk) ntk next-unvisited ntkstate)
                (xmas-transfer-calculate 'irdy (get-out-channel cpt index-out ntk) ntk next-unvisited ntkstate))))))
```

Figure 4. Irdy signal computing code

Paper [6] has some research about formalization of xMAS in ACL2, but it only formalizes five components of eight, this paper would improve this process of formalization, focus on the three remaining components.

First, define a common structure using defstructure macro to denote components and channels.

(defstructure component type ins outs field); (4)

(defstructure channel init target); (5)

Formula (4) and (5) defines component and channel respectively, component has four domains, type indicates component type, ins indicates the input channels, outs represents the output channels, field indicates the function element. Channel has two domains, init and target represent initial component and target component respectively.

In ACL2, field of component is represented by alist, like '(' (* * *) '(* * *)), for merge and join, each sub-sequence representing two parameters corresponding to a respective function values, for fork, it has one parameters and two return values, the function defined to get result from field is shown in Fig. 3.

Function xmas-transfer-calculate [6] is the main function computing xMAS network, including signal of channel and the data flow, it has five parameters: xmas-transfer-calculate (flg channel ntk unvisited ntkstate)

Flg represents values to be calculated: irdy, trdy and data; Channel is currently selected channels, ntk denotes current network, ntkstate is current network status, unvisited is combination of channel with the signal of the current network used to ensure termination of function; Its main structure is calculating current channel and signal and traverse the entire network. To add the three components into this formalization, we need to add new branchs selection structure in this function, Figure. 4, shows the example code of signal irdy.

Formal Verification and Analysis

In order to verify the functionality correctness of the serial port, first should establish its formal model[7-9]. Based on the semantic of xMAS and the function of serial port module, xMAS model of serial port module is shown in Fig. 5.

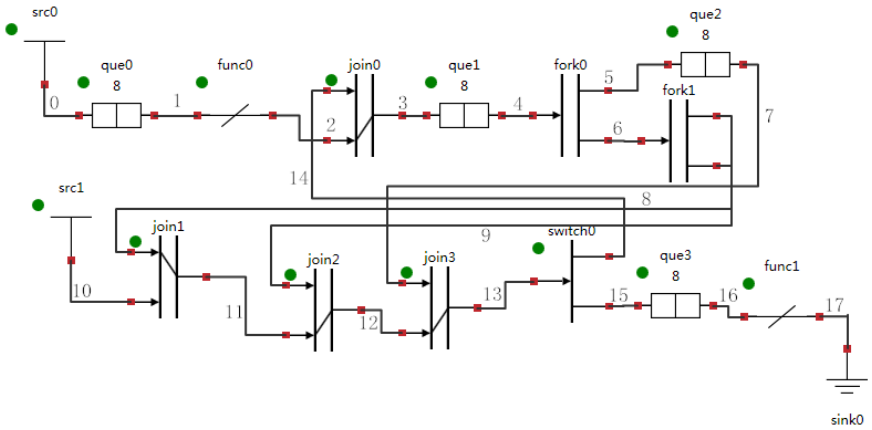

Figure 5. xMAS model of serial module

After establishing xMAS model, we need to define components and channels in ACL2. According to the formal description of component and the channel, Fig. 6 gives the code of component definition in the xMAS model.

```
(defconst *src0*  (component 'source nil '(0) '(msg1 nil nil msg2 nil)))
(defconst *que0*  (component 'queue '(0) '(1) '()))
(defconst *func0* (component 'function '(1) '(2) '('(num (frame num)))))
(defconst *join0* (component 'join '(2) '(14) '('(1 num num) '(0 num nil))))
(defconst *que1*  (component 'queue '(3) '(4) '()))
(defconst *fork0* (component 'fork '(4) '(5 6) '('(num num 1))))
(defconst *que2*  (component 'queue '(5) '(7) '()))
(defconst *fork1* (component 'fork '(6) '(8 9) '(1 1  1)))
(defconst *src1*  (component 'source nil '(10) '(1 2 3 4 5 6 7 8 9 10)))
(defconst *join1* (component 'join '(8 10) '(11) '('(1 num num) '(0 num nil))))
(defconst *join2* (component 'join '(9 11) '(12) '( '(num1 num2 (fractional-frequency num1)))))
(defconst *join3* (component 'join '(7 12) '(13) '('(num 0.01 num) )))
(defconst *switch0* (component 'switch '(12) '(13 14) '('(num num) '(nil tx_done))))
(defconst *que3*  (component 'queue '(15) '(16) '()))
(defconst *func1* (component 'function '(16) '(17) '(frame-num (dis-frame frame-num))))
(defconst *sink0* (component 'sink '(17) nil '()))
```

Figure 6. Definition of xMAS components in ACL2

As seen in Fig. 6, components are defined using an ACL2 function called defconst, in this way could quote these components convinently. In join2 component, by loop assignment on cycle variable to simulate the frequency dividing operation. Code for part of channels definition is shown below, each channel connects two components.

(Defconst * channel0 * *src0 * * que0 *)
(Defconst * channel1 * * que0 * * func0 *)
(Defconst * channel2 * * func0 * * join0 *)
(Defconst * channel3 * * join0 * * que1 *)
(Defconst * channel4 * * que1 * * fork0 *)
(Defconst * channel5 * * fork0 * * que2 *)

During working process of the robot, the camera head is an important way to collect data, which has a significant impact on subsequent decisions, path planning. Therefore, it's an important step to send instruction to PTZ control accurately and real-timely. For the xMAS model above, extract two key properties [10] and verify it in ACL2. Fig. 7, and Fig. 8 shows codes for these two properties.

```
(defthm message-consistency
    (let ((result (xmas-transfer-calculate 'data channel ntk unvisited ntkstate)))
        (implies
                (and (xmasnetworkp ntk)
                     (member-equal channel (xmasnetwork-channels ntk))
                     (msg-invariant src0))
            (msg-invariant sink0))))
```

Figure 7. Verification code of property1

```
(defthm xmas-transfer-implies-available-space
    (let ((result (xmas-transfer-calculate 'trdy channel ntk unvisited ntkstate)))
        (implies
                (and (xmasnetworkp ntk)
                     (member-equal channel (xmasnetwork-channels ntk)))
            (xmas-can-receive resource ntkstate))))
```

Figure 8. Verification code of property2

Property 1. Message consistency, i. e, message could transfer through the network and arrive the final destination node without modification. ACL2 code is shown in Fig. 7.

The code used in the fifth row from src0 inject message function representation of the network, the network will eventually discharged from sink0, and consistent message. Within the limits of the above theorem, to ensure consistency of the message sent.

Property 2. There is enough space in the queue to receive data packets to be sent. During deliver the message will be forwarded several times and temporarily stored in the queue, the theorem guarantees every forwarding destination queue has enough space to receive the current message.

All the codes above run successfully in ACL2, Figures 9 shows verification results of property 1 and property 2 in ACL2, Table 1, shows verification data, time estimates based on Intel Core i5 -3230M processor.

```
Q.E.D.

Summary
Form:  (DEFTHM XMAS-TRANSFER-IMPLIES-AVAILABLE-SPACE ...)
Rules: ((:DEFINITION XMASP)
        (:DEFINITION XMAS-NETWORKP)
        (:EXECUTABLE-COUNTPART CONSP)
        (:EXECUTABLE-COUNTPART XMAS-CAN-RECEIVE)
        (:REWRITE CDR-CONS)
TIME:  0.18 seconds (prove: 0.18, print: 0.00, other: 0.00)
Prover steps counted:  1357
XMAS-TRANSFER-IMPLIES-AVAILABLE-SPACE
```

```
Q.E.D.

Summary
Form:  (DEFTHM MESSAGE-CONSISTENCY ...)
Rules: ((:COMPOUND-RECOGNIZER NATP-COMPOUND-RECOGNIZER)
        (:DEFINITION XMASP)
        (:DEFINITION XMAS-NETWORKP)
        (:DEFINITION MSG-INVARIANT)
        (:EXECUTABLE-COUNTERPART MESSAGEP)
        (:EXECUTABLE-COUNTERPART EQUAL)
        (:FAKE-RUNE-FOR-TYPE-SET NIL)
        (:REWRITE XMAS-RESOURCES-GEN))
Time: 0.27 seconds (prove: 0.27, print:0.00, other:0.00)
Prover steps counted: 1836
MESSAGE-CONSISTENCY
```

Figure 9. Verification results

Table 1 Verification data

	loc	prove steps	time[second]
Property 1	6	1836	0.27
Property 2	5	1357	0.18

Summary

This is the first time that xMAS is fully formalized in ACL2, and then use it for the functional correctness verification of serial port module. Verifying results could guarantee the reliability of the

serial module in robot control system. In addition, method described here provides a new reference method for verification of key module in robot control system. In future work, xMAS support for asynchronous communication will be researched, hope that it could play strengths in the field of asynchronous communication.

Acknowledgements

National Natural Science Foundation of China (61373034).
Natural Science Foundation of China(4122017).

Refserences

[1] Willows, Cheang Chi Keong 87C196KC based soccer robot control system design [J] Computer Measurement & Control, 2002, 10 (7): 449-451.

[2] Chen Jianbin. ARM and FPGA-based mobile robot control system design [D]. South China University of Technology, 2011.

[3] ACL2 proof [EB / OL]. Http://zh.wikipedia.org/wiki/ACL2.

[4] Kaufmann M, Manolios P, Moore J S, et al. Computer-Aided reasoning: ACL2 case studies[M]. Kluwer Academic Publishers, 2000.

[5] Chatterjee S, Kishinevsky M. Automatic generation of inductive invariants from high-level microarchitectural models of communication fabrics [J] Formal Methods in System Design, 2012, 40 (2):. 147-169.

[6] Gastel B V, Schmaltz J. A formalisation of XMAS [J]. Eprint Arxiv, 2013.

[7] Burns F, Sokolov D, Yakovlev A. GALS Synthesis and Verification for xMAS Models[C]// Design, Automation & Test in Europe Conference & Exhibition (DATE), 2015. IEEE, 2015:1419-1424.

[8] Chatterjee S, Kishinevsky M, Ogras U Y. Quick formal modeling of communication fabrics to enable verification [C] .High Level Design Validation and Test Workshop (HLDVT), 2010 IEEE International IEEE, 2010:. 42-49.

[9] Verbeek F, Schmaltz J. Formal specification of networks-on-chips: deadlock and evacuation [C] .Proceedings of the Conference on Design, Automation and Test in Europe European Design and Automation Association, 2010:1701-1706.

[10] Gao Ya, Li Xiao-juan, Guan Yong, et, al. Theorem Prover ACL2-based Verification of Node Communication in the Robot Operating System ROS [J]. Journal of Chinese Computer System, Vol.35, No.9 2014: 2126-2130s