A Work Around for Communication Frame's Generation Solver Issue in Constrained Random Verification

Lirong Qiu

***Abstract*—In communication system especially in wireless communication system generating size constrained frames with different format header and different size of packet data carried usually are the most common task handled by the data path stimulus generator in a constrained random verification system. The main challenge for generator is to solve out the well distributed stimulus according to the complex constraint relationship. R99's CCTrCH frame's building set an example on such challenge. This paper presents a work around to handle such challenge. By using a method like truck loading, the direct stimulus solver problem is avoided. As complicated constraints can be changed to the condition to stop loading more, no solver failure is achieved. Leveraging the power of computer resource, this work around can also make a good distribution about the generated stimulus.**

***Index Terms*— CCTrCH, constrained random verification, solver, stimulus generation, complicate constraint**

I. Introduction

With the development of deep sub-micrometer fabrication technology, electronic designs have been growing rapidly in both device count and functionality. Verification complexity grows faster than the design complexity. As one of remedies for verification complexity, constrained random simulation with robust constraint solving capability is proposed as the key to any practical testbench automation tool [1] and become the main workhorse in today's hardware verification flows. The efficiency of the overall flow depends critically on (1) the performance of the constraint solver and (2) the distribution of the generated solutions [2]. There are quite a few general-purpose constraint solvers available both from academia and industry. However sometimes such constraint solver may not be enough due to different design's functionality.[3] In communication systems especially in wireless communication system, integrated circuit solutions have been one of the enabling technologies, contributing to the success of wireless communications [4]-[6]. Due to variety of services are usually beard by radio data link whose rate is stable in specified time slot, generating size constrained frames with different format header and different size of packet data carried usually are the most common task handled by the data path stimulus generator in wireless communication IC verification. The challenging part of such stimulus generation is that it is hard for solver to figure out what a packet data composition can be fit into the known sized frame.

To generate MAC-e PDU [7] is an example. Figure 1 shows the format of MAC-e PDU.

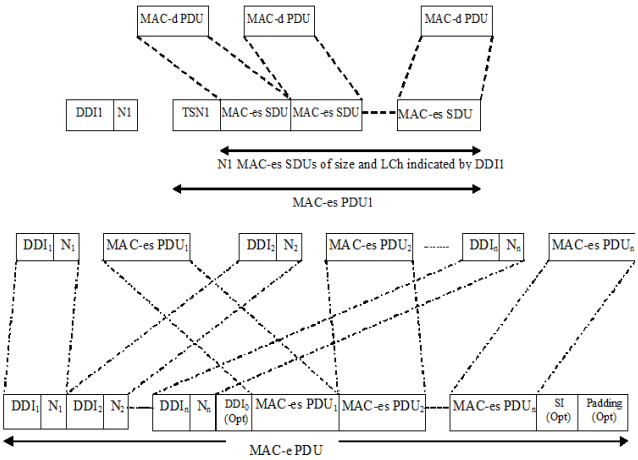

Fig. 1 MAC-e PDU frame structure

The generated MAC-e PDU should meet following constraint:

$$(L_{DDI} + L_N + L_{TSN}) \times n_{esp} + randbit \times L_{DDI} + \sum_{i=0}^{n_{esp}} (n_{ess}^i \times l_{ess}^i) + randbit \times L_{SI} + l_{pad} = l_{ep}$$

In this formula, all capitals are known constant value and all low-case letters represent random variable. Here l_{ep} is the length of MAC-e PDU which is in a known set defined by protocol and can't be randomized freely. n_{esp} is the number of MAC-es PDU and less than a constant value. n_{ess}^i is the number of MAC-es SDU in i^{th} MAC-es PDU and less than a constant. l_{ess}^i is the length of MAC-es SDU in i^{th} MAC-es PDU which is a multiple of 8. $randbit$ is a randomized value in 0 and 1, as DDI_0 and SI are optional. Here the constraints that l_{ep} belongs to a known set and l_{ess}^i is a multiple of 8 bring great difficult to solve the right stimulus. It will be even harder to generate the MAC-e PDU without padding.

Another example is 3G general CCTrCH frame building [8]-[9]. This multiplexing, channel coding and interleaving procedure has similar constraint like MAC-e PDU: A couple of transport blocks with known size should be mapped into

Manuscript received May 15, 2013; revised November 10, 2013. This work was supported by the National nature science foundation of China (No. 61103161) and the Program for New Century Excellent Talents in University (No.NCET-12-0579).

Lirong Qiu is with the School of Information Engineering, Minzu University of China, and Beijing, China. (E-mail: qiu_lirong@126.com).

one CCTrCH frame; Total bits of one CCTrCH frame can carried are constrained to a value in a known set according to its mapped physical channels; Total bits for all transport blocks is a constant value decided by user equipment capability; Total transport block number is a constant value decided by user equipment capability, etc. One of the toughest constraints is rat matching. Due to rate matching, we can't put each bit in transport blocks CCTrCH frame. Some bits need to be discarded or some redundant bits need to be added. For the rate matching algorithm, the iteration, division and floor operations are all hard to solve. Moreover sometimes we can't let the final puncture rate of transport blocks be bigger than its coding rate as some of the original transmitting bits of transport blocks can't be recovered if puncture rate is too big. That may lead to comparison failure if testbench does a TX-RX transport block check. But the puncture rate is an indirect constraint brought by the complicated rate matching algorithm. So it is even harder for solver to meet such constraint.

As the critical issue of constraint based verification, solver's ability has always been the focus in computer aided verification. A lot of study has been made about the solver algorithm [10]-[11]. Some methods to improve constraint solving technique for specified verification language are also proposed [12]-[13].

However, it is still hard to solve complicated constraint directly. One solution proposed by [1] is to define intermediate sub-formulas upon which the constraint formula is defined. However sometimes to subtract the right intermediate sub-formulas is also a challenging job. For example to get the puncture rate for one transport blocks in above radio frame composition procedure. This paper provides its study on the special class solver issue of generating size constrained frames with different packet and header loaded for communication system. By using a method like loading truck, the direct stimulus solver problem is worked around. As complicated constraints can be changed to the condition of stop loading, randomize engine don't have to analyze the complicated constraint and the system will never have solver failure issue. The following part of this paper includes:

- The method to work around the direct stimulus solver problem.
- Several factors to optimize the work around method.

As the method changes the complicated constraint to the condition of stop loading, the corner case can't be achieved by direct constraint. But as the inputs can be randomized independently and there is no solver issue, they can be constrained to the set which is easier to hit the expected corner case. By leveraging the power of computer resource, it is not hard to cover the corner cases.

II. Work Around for Solver Issue of Generating Size Constrained Frame

In previous chapter, we have pointed out that due to the complicated constraint, it is hard for solver to figure out what a packet data composition can be fit into the known sized frame by analyzing the constraint directly. We can take 3G TDD general downlink CCTrCH frame building as an example and assume UE's capability is 768kbps service supported. According to the table 5.2.2.1 in [14], Constraint which is relative to CCTrCH frame building can be subtracted as following items. Here Number of bits in a radio frame before rate matching on TrCH i with transport format combination j is represented by $N_{ij;}$ Number of bits to be punctured or repeated in each radio frame on TrCH i with transport format combination j is represented by ΔN_{ij} ; Transport block number of TrCH i is represented by M_i; Transport block size of TrCH i is represented by $A_{i;}$ Attached CRC bit size is represented by L_i ; Coding scheme is represented by $Coding_i$; TTI length is represented by TTI_i; Number of TrCHs in a CCTrCH is represented by I; Total maximum number of bits of one CCTrCH frame is represented by $Ndata_{max}$.

a) Maximum sum of number of bits of all transport blocks being received at an arbitrary time instant:
$\sum_{i=1}^{I}(A_i \times M_i) \leq 10240$.

b) Maximum sum of number of bits of all convolutionally coded transport blocks being received at an arbitrary time instant:
$\sum_{i=1}^{I}(A_i \times M_i \times Fisconv(Coding_i)) \leq 640$;
$Fisconv(Coding_i)$ is a function to check if coding scheme is convolutionally coding for TrCH i. If it is, it will return 1; or return 0.

c) Maximum sum of number of bits of all turbo coded transport blocks being received at an arbitrary time instant:
$\sum_{i=1}^{I}(A_i \times M_i \times Fisturbo(Coding_i)) \leq 10240$;
$Fisturbo(Coding_i)$ is a function to check if coding scheme is Turbo coding for TrCH i. If it is, it will return 1; or return 0.

d) Maximum number of simultaneous transport channels:
$I \leq 8$.

e) Maximum total number of transport blocks received within TTIs that end at the same time:
$\sum_{i=1}^{I}(M_i) \leq 64$.

f) Maximum number of physical channels per subframe is less than 64.

g) Total maximum number of bits of one CCTrCH frame is $Ndata_{max}$ decided by its mapped physical channels and timeslot format of each physical channel.

h) Number of bits in a radio frame before rate matching on TrCH i with transport format combination N_{ij} is decided by transport block number, transport block, attached CRC bit size, coding scheme and radio frame segmentation. We use a simple function symbol $Fcrc_code_seg()$to represent the calculation ofN_{ij}:
$N_{ij} = Fcrc\_code\_seg(A_i, M_i, L_i, Coding_i TTI_i)$.

i) If there is comparison between the decoded transport blocks and the original transmitted transport blocks, the puncture rate after rate matching should not be bigger than the coding rate:
$\frac{\Delta N_{ij}}{N_{ij}} \leq code\_rate$.

In fact 768kbps TDD can have maximum 4 of simultaneous CCTrCHs. However, 3 of them are usually used to carry BCH, FACH and/or PCH, DSCH respectively. The constraints for CCTrCHs carrying BCH, FACH and/or PCH are relatively easy as less physical channels are used and TrCH is limited. CCTrCH for DSCH has been replaced

by HSDPA service. So this paper just focuses on the solver issue of the most complicated CCTrCH's building with dedicated type TrCH contained.

A. Constraint Analysis and Work around Method

To analyze the constraint lists above, we can see following points:

- Item f and item g are quite independent constraints. According to timeslot format of one physical channel, total number of data bits in one physical channel belongs to the set of {88, 86, 84, 80, 72, 84, 82, 80, 76, 68}. We randomize the assigned physical channel first within 1 to 64 and then randomize the timeslot format of each physical channel to work out $Ndata_{max}$.
- it is a little hard to solve constraints from item a to item e as the constraint variables are relative with each other and so many constraint items narrow the variables' random space.
- Item i is a complicated constraint as it depends on the connection between TrCH transport blocks and total number of bits of CCTrCH and such connection is not a straightforward and clear one.

Strictly speaking item h and rate matching are not constraints. But they build the connections between TrCH transport blocks and CCTrCH frame. Item h is easier to use intermediate variables in constraint expression becauseN_{ij}'s calculation is single direction and only linear operation is used. But rate matching algorithm like Figure 2 shows are quite complicated and not straightforward and clear. Constraint item i is buried in the connection. Normally we can randomize some variables which are constrained to more compact random space and infer other variables random space to work around the hard solver issue. Here CCTrCH frame bits and real rate matching rate are two variables need to randomize in advance. But the iteration, division and floor operations in rate matching algorithm prevent the inferring of TrCH blocks with a known puncture rate and CCTrCH frame bits.

$$Z_{0,j} = 0$$

$$Z_{i,j} = \left\lfloor \frac{\left(\left(\sum_{m=1}^{i} RM_m \times N_{m,j} \right) \times N_{data,j} \right)}{\sum_{m=1}^{I} RM_m \times N_{m,j}} \right\rfloor \quad \text{for all i = 1 ... } I$$

$$\Delta N_{i,j} = Z_{i,j} - Z_{i-1,j} - N_{i,j} \quad \text{for all i = 1 ... } I$$

Fig. 2 Rate matching algorithm

To work around this hard solver issue, let's think about how physical channel and transport channel are defined in 3G system. A colorful metaphor is used to help with the understanding of both different channels: physical frame transferred on physical channel is a truck while transport blocks in channel are packages with different size and carried by this truck. Like figure 3 shows: when a truck is loading at goods yard, the truck can always been fully loaded because workers can check the space of the truck left and find the right packages to fill it from thousand ones piled in goods yard. If the left space is quite limited, workers can try to find a small one.

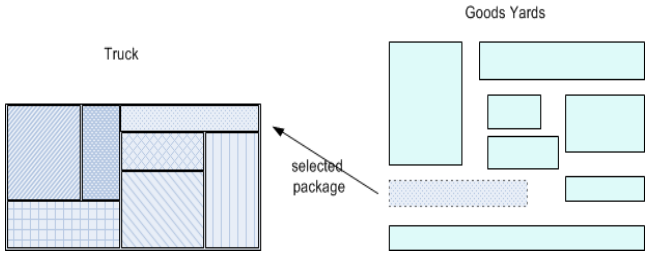

Fig. 3 Diagram of Loading Last space of a truck

The main reason that the truck can be loaded compactly is that there are so many of different size of package which can be selected during truck loading that a fitful size package can always be found to fill the left truck's space. This truck loading method brings a hint to the solver issue: we can randomize abundant of all kinds of TrCH blocks and try to fill them into CCTrCH frame. When one constraint is broken by a new TrCH block, this block will be discarded.

B. Implementation about Work Around

The detail implementation about the work around mentioned in previous chapter is shown in the flow chart of Figure 4.

From the flow chart we can see that the assigned physical channels and the timeslot format of each physical channel for CCTrCH is randomized first. $Ndata_{max}$ is figured out. $Ndata_{max}$ is also the initial "left space of Truck". With this space, a rough "maximum package size", maximum total block bits from one TrCH, can be inferred. We can set a little larger than the inferred value because TrCH blocks are just randomized in this range. If some TrCH blocks are too big for CCTrCH frame, they will be discarded and new TrCH blocks can be randomized. After blocks from TrCH are evaluated to be fitful for CCTrCH frame, the maximum total block bits can be re-figured based on left space. That can lead to a quick convergence to the working around because new randomized TrCH blocks can have less possibility to be discarded due to a good randomization range adjustment based on the feedback of left space. steps①②③④ are functions to check if TrCH blocks are fitful for CCTrCH frame. Step⑤⑥⑦ are functions to check if current filling process is done. Obviously randomization for CCTrCH mapped physical channel and their time slot are quite easy. So are TrCH blocks. Complicated constraints are not considered any more during their randomization because they are changed to checking functions in step①②③④ and terminating function in step ⑤⑥⑦. We set the maximum TrCH blocks number, maximum TrCH channels etc according to UE ability in the flow chart. In fact they can be randomized in advance for expected test scenarios.

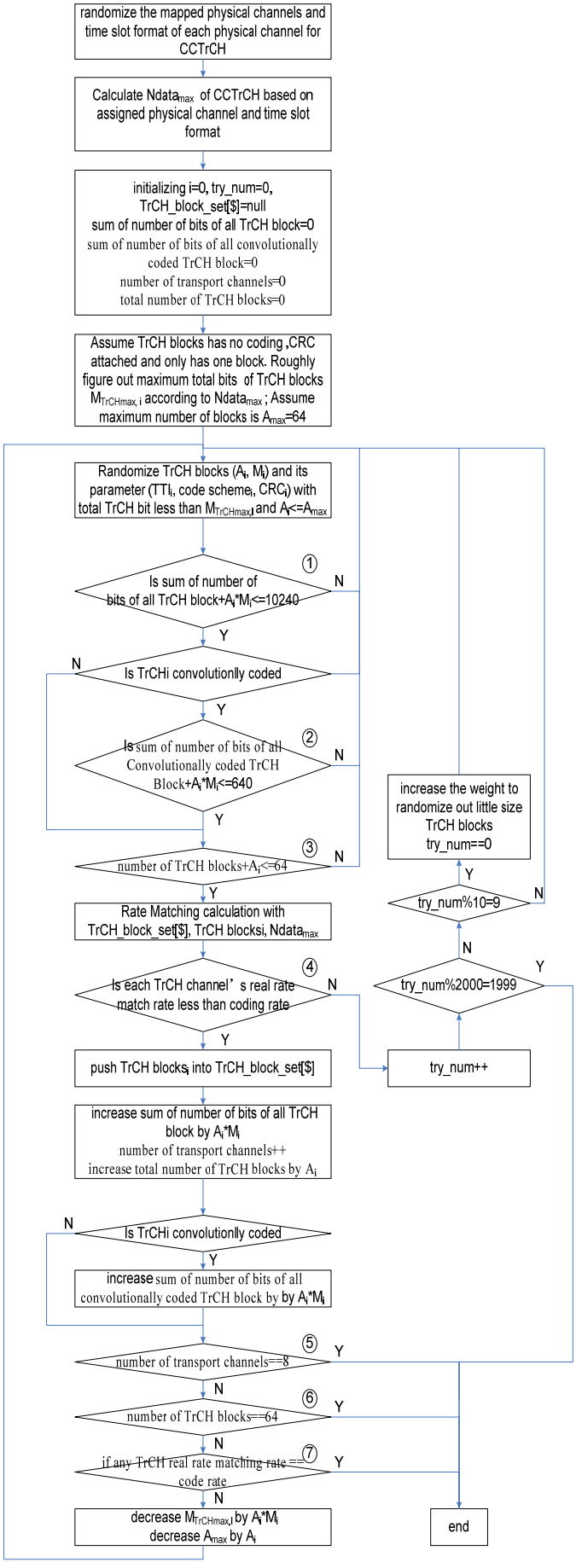

Fig. 4 Flow chart to randomize TrCH blocks for CCTrCH

However one thing we should worry about is that as the constraints are not used anymore, we may face the issue of creating test case with specified constraints such as the case of total transport blocks number is 64, total bits of all convolutional coded transport blocks is 640 and rate match rate is coding rate. In fact getting the specify constrained case hit directly is impossible for this work around. But by slicing the random value range and adjusting the distribution, the specify cases can be hit by leveraging the power of computer resource. For example, if Trch block has high weight to randomize into small size one, the case of total transport blocks number is 64 can have high possibility to hit.

Summary for Variable punclimit

CATEGORY	EXPECTED	UNCOVERED	COVERED	PERCEI
Automatically Generated Bins	2	0	2	100.0

Automatically Generated Bins for punclimit

Bins

NAME	COUNT	AT LEAST
auto[0]	2998	1
auto[1]	2	1

Summary for Variable total_conv

CATEGORY	EXPECTED	UNCOVERED	COVERED	PERCENT
User Defined Bins	21	2	19	90.48

left[451:480]	5	1
left[481:510]	5	1
left[511:540]	6	1
left[541:570]	2	1
left[571:600]	7	1
top	16	1

User Defined Bins for channel_num

Bins

NAME	COUNT	AT LEAST
left_1	24	1
left_2	62	1
left_3	109	1
left_4	205	1
left_5	276	1
left_6	268	1
left_7	314	1
top	321	1

Fig. 5 a coverage report snapshot generated with 3000 test sets

With this method, we make a test program and run at VCS simulation environment. According to the coverage report showed in Fig 5, with 3000 test sets, 2 puncture limit cases are hit. We also get maximum TrCH channel numbers and maximum sum of all convolutionally coded transport blocks bits. Following data is one TrCH block sets. We can see the distribution is quite random and not restricted to one or two blocks.

blknum[1] blksize[27] coding[2] crc[0] tti[1] punc_weight[143] coderate[3]
blknum[1] blksize[17] coding[0] crc[16] tti[1] punc_weight[258] coderate[1]
blknum[2] blksize[262] coding[1] crc[16] tti[2] punc_weight[124] coderate[2]
blknum[2] blksize[131] coding[0] crc[8] tti[1] punc_weight[282] coderate[1]
blknum[3] blksize[461] coding[2] crc[16] tti[4] punc_weight[145] coderate[3]
blknum[43]blksize[28] coding[2] crc[24] tti[1] punc_weight[127] coderate[3]
blknum[4] blksize[602] coding[2] crc[24] tti[2] punc_weight[125] coderate[3]

blknum[1] blksize[293] coding[0] crc[0] tti[1] punc_weight[263] coderate[1]
blknum[2] blksize[28] coding[1] crc[8] tti[1] punc_weight[139] coderate[2]
blknum[3] blksize[37] coding[2] crc[24] tti[4] punc_weight[139] coderate[3]
blknum[1] blksize[44] coding[1] crc[0] tti[4] punc_weight[100] coderate[3]
blknum[1] blksize[5] coding[1] crc[24] tti[2] punc_weight[109] coderate[2]
blknum[40] blksize[113] coding[2] crc[0] tti[4] punc_weight[131] coderate[3]
blknum[11] blksize[2] coding[2] crc[0] tti[1] punc_weight[150] coderate[3]

III. Factors of Optimizing the Working Around

As complicated constraints are changed to checking functions and terminating functions in the work around in this paper, the solver failure issue will be gone. However the convergence time for stimuli generation and the stimuli's distribution are important concern about such solver method. Here we will discuss several factors which affect the performance of work around.

A. Priority of Loading Package at Different Position

Like trucking loading, sometime we need put some packages in special position. Such packages should be placed in high priority. Loading packet data into a frame has similar issue. For example: when packet data has variable size and the frame which has packets loaded has fix size, packet data need to be assembled or segmented to get fit into destiny frame and special header or padding may be attached to the frame. Such variety of headers or paddings are quite important the assembling/segmentation scenario. If they are loading first, frame usually has enough space to carry them. So they have less possibility to break the checking rule and good stimuli distribution can be achieved.

B. Resize Packet's Random Range According to Frame's Left Space

As the checking function can guarantee that packet with too large size can be discarded, packet's random range can be set to a very large one. However this is not good for random convergence time for stimuli generation because if current packet is too large to be fit for the left space of frame, this packet will be discarded and a new one needs to be randomized out for next around try. More time has to be taken to get the frame fully loaded. So it is better to check the left space left in frame and figure out the maximum packet size the left space can carry. Let new packet randomize in this range. This method has been taken in the example discussed in chapter 2.2.

C. Get the Generated Frame Refined

After the frame is generated, we may want to some post processing. So it is better to have a callback after frame is generated. Little turning on frame can be done at this stage such as shrink or enlarge some fields, inject some exceptions, information printing etc. Sometimes frame may have some space left and padding is required. So we can pad the frame with post processing. An example is MAC-e PDU's generation. MAC-e PDU may not load any more MAC-es SDU due to the limitation of maximum number of MAC-es SDU, but it still has space left. In such case, DDI0, SI and padding can be selected to fill the rest of space. Although we can regard DDI0, SI and padding as special field of MAC-e PDU and randomize them during loading stage, it is better to isolate them be processed at post processing stage. In this way checking function and terminating function are concise in functionality and easy to maintain.

IV. Summary and Future Work

The method presented by this paper to work around the complicated constraints' solver issue in communication frame's generation is an emulation of truck loading process. The complicated constraints are changed to checking rules or terminating rules. If source packets or fields are not fitful the frame to be loaded according to the checking rules and terminating rules, the packets or fields will be discarded and another one will be randomized out for next try. So there is no solver failure issue in this process as complicated constraints are not considered during randomization.

This work around avoids to solve the complicated constraints and is quite fitful for the case of communication frame's generation. But it should be fitful for other cases. Future work can be done to model the constraint expressions and subtract the common feature of these constraint expressions. Then this work around can be applied to these constraint expressions automatically and become one part of the solver algorithm.

ACKNOWLEDGEMENT

Our work is supported by the National nature science foundation of China (No. 61103161) and the Program for New Century Excellent Talents in University (NCET-12-0579).

References

[1] J. Yuan, C. Pixley, A. Aziz. "Constraint-Based verification", *Springer*, 2006
[2] N Kitchen, A Kuehlmann, "Stimulus generation for constrained random simulation", Proceedings of IEEE/ACM international conference on Computer-aided design, 2007
[3] .Naveh, M.Rimon, I.Jaeger, Y.Katz, M.Vinov, "Constraint-Based Random Stimuli Generation for Hardware Verification," *AI magazine*, vol. 28, no. 3, pp.13-29. 2007
[4] Soerensen, J., Birk, P., & Zvonar, Z. "New Challenges for Integrated Circuit Solutions," *Wireless Personal Communications*, vol. 17, no.2-3, pp.291-302. 2001.
[5] Kim, J., Ha, D. S., & Reed, J. H. "A new reconfigurable modem architecture for 3G multi-standard wireless communication systems," *In Circuits and Systems, 2005. ISCAS 2005.* pp. 1051-1054.
[6] Martelli, C., Reutemann, R., Benkeser, C., & Huang, Q. "A 50mW HSDPA Baseband Receiver ASIC with Multimode Digital Front-End," *In Solid-State Circuits Conference, 2007. ISSCC 2007.* Digest of Technical Papers. San Francis, CA, 2007, pp. 260-601
[7] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Radio Access Network; MAC protocol specification (Release 7)", TS25.321 version 7.h.0 (Jun. 2010)
[8] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Multiplexing and channel coding (FDD) (Release 1999)", TS25.212 version 3.4.0 (Sep. 2000)
[9] 3GPP, "3rd Generation Partnership Project;Technical Specification Group Radio Access Network;Multiplexing and channel coding (TDD)(Release 1999)", TS25.222 version 3.4.0 (Sep. 2000)
[10] Chai, D., & Kuehlmann, A. "A fast pseudo-boolean constraint solver," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on,* vol. 24, no. 3, pp.305-317. 2005

[11] Zeng, Zhihong, Maciej Ciesielski, and Bruno Rouzeyre. "Functional test generation using constraint logic programming," *SOC Design Methodologies. Springer US*, pp. 375-387. 2002.
[12] Ferrandi, F., Rendine, M., & Sciuto, D. "Functional verification for SystemC descriptions using constraint solving," *In Proceedings of the conference on Design, automation and test in Europe.* pp. 744. IEEE Computer Society.
[13] Große, D., Ebendt, R., & Drechsler, R. "Improvements for constraint solving in the SystemC verification library," *In Proceedings of the 17th ACM Great Lakes symposium on VLSI. 2007*, pp. 493-496. ACM.
[14] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Radio Access Network; UE Radio Access Capabilities (Release 1999)", TS25.306 V3.0.0 (Dec. 2000)

Lirong Qiu. She was born in Jinan City, Shandong Province, People's Republic in 1978. She received his M.Sc. in Computer Sciences (2004) and PhD in Information Sciences (2007) from Chinese Academy of Science. Her current research interests include different aspects of Artificial Intelligence and Distributed Systems.

Now she is full professor of computer sciences at Information Engineering Department, Minzu University of China.

Prof. Qiu is the member of artificial intelligence community of China.